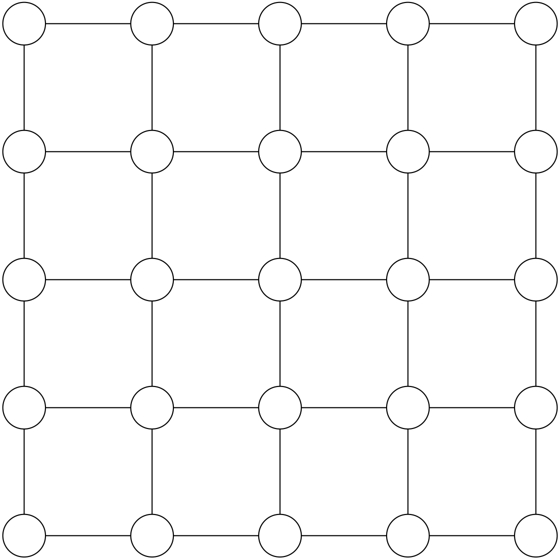
Space-filling curves for 3D mesh traversals

Michael Bader, TU München

Herman Haverkort, TU Eindhoven

Tobias Weinzierl, Durham University



FINITE ELEMENT METHOD:

repeat

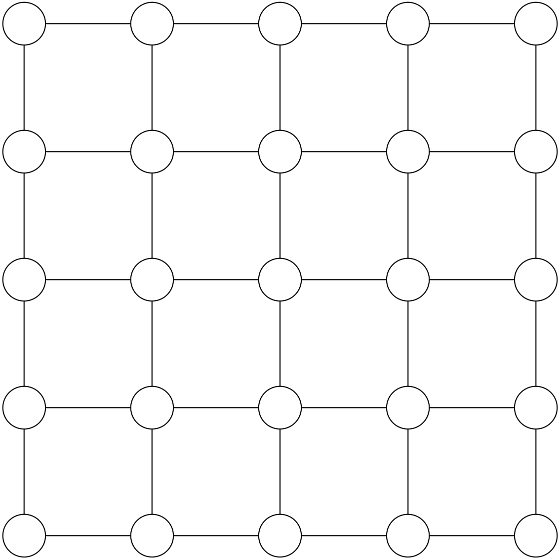
for each square cell

retrieve current values of the four vertices

compute and store new values for the four vertices

until happy

For example: plate with heat sources and sinks at subset of vertices; compute steady-state heat distribution and flow.



In which order? Row by row?

FINITE ELEMENT METHOD:

repeat

for each square cell

retrieve current values of the four vertices

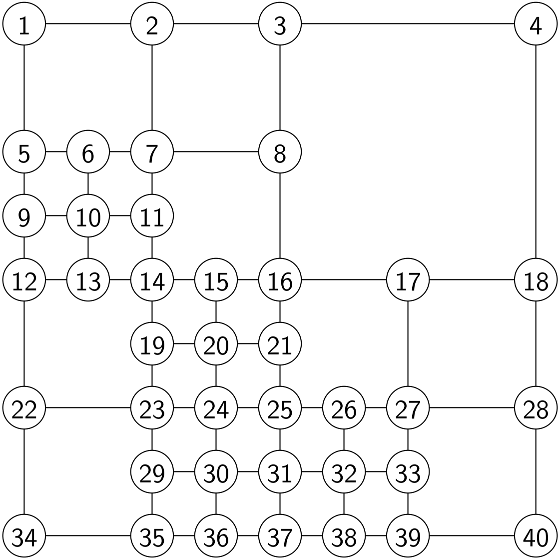
compute and store new values for the four vertices

until happy

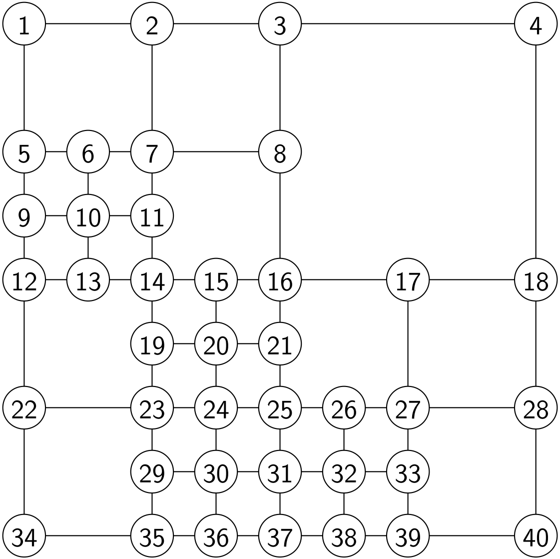
In what data structure? Matrix?

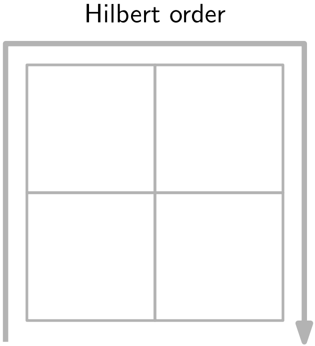
For example: plate with heat sources and sinks at subset of vertices; compute steady-state heat distribution and flow.

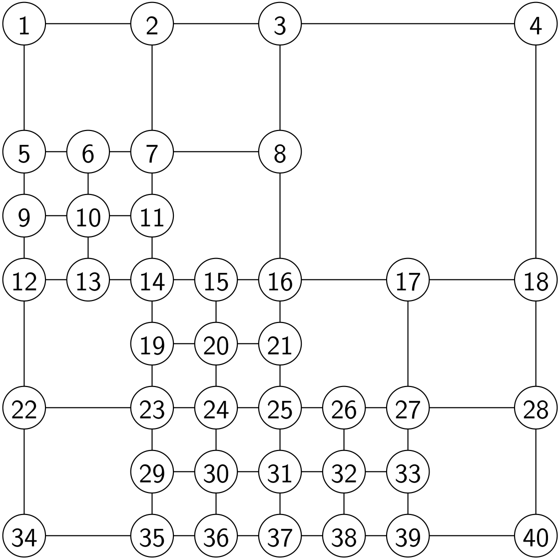
Traversing an irregular grid

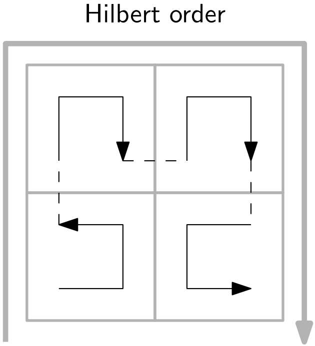


Traversing an irregular grid

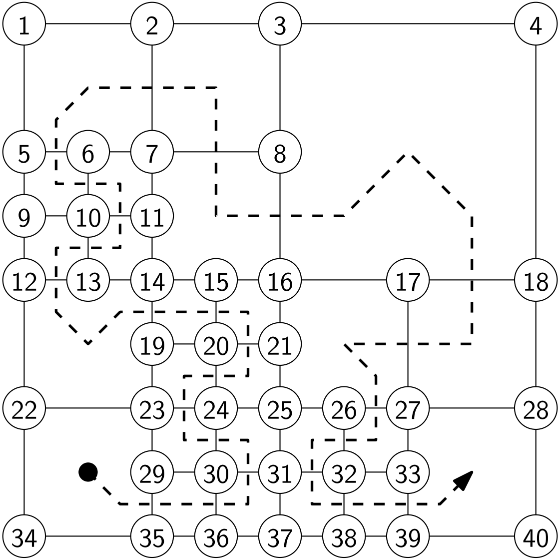


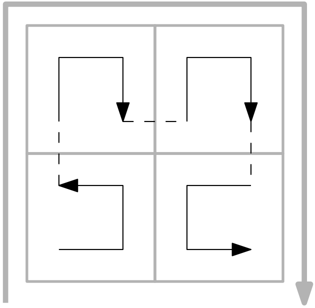



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40

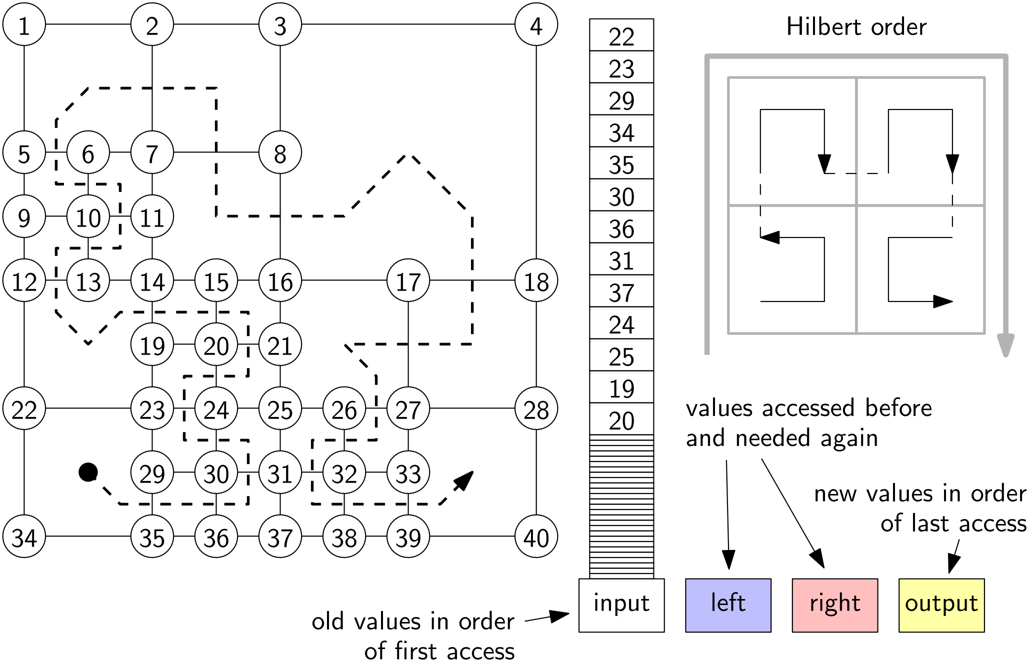

Hilbert order

Traversing an irregular grid

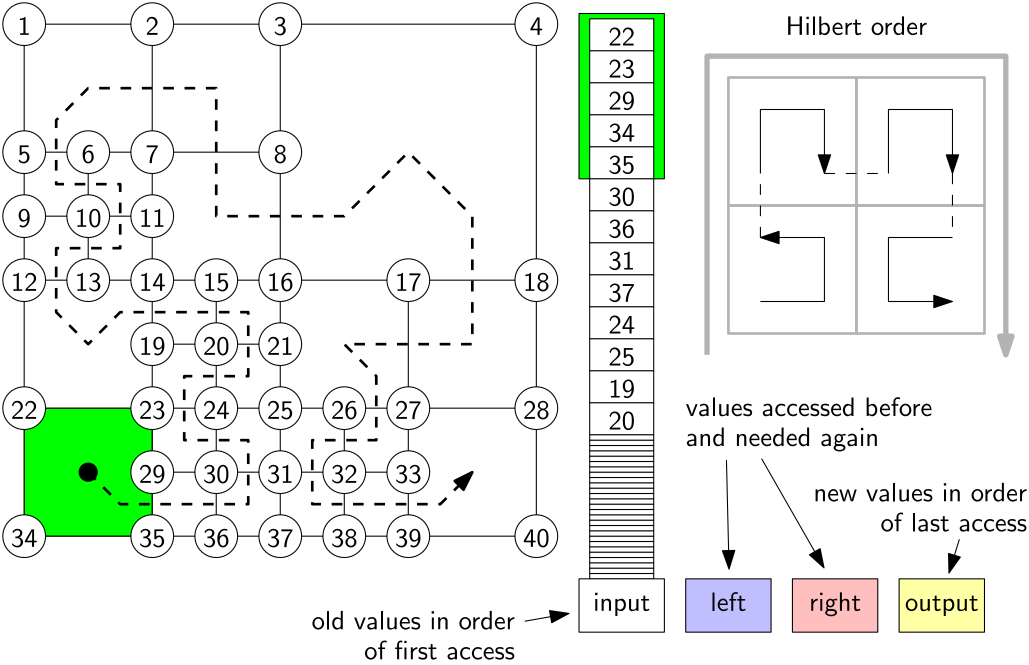




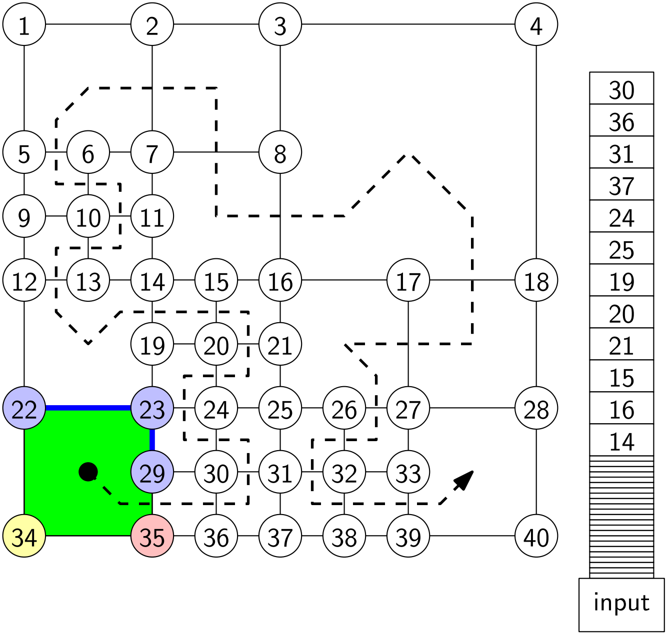
Traversing an irregular grid

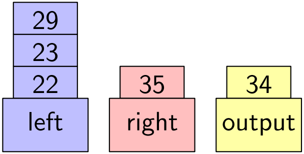


Traversing an irregular grid

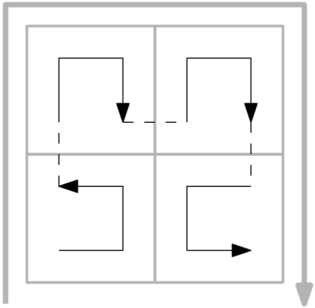


Traversing an irregular grid

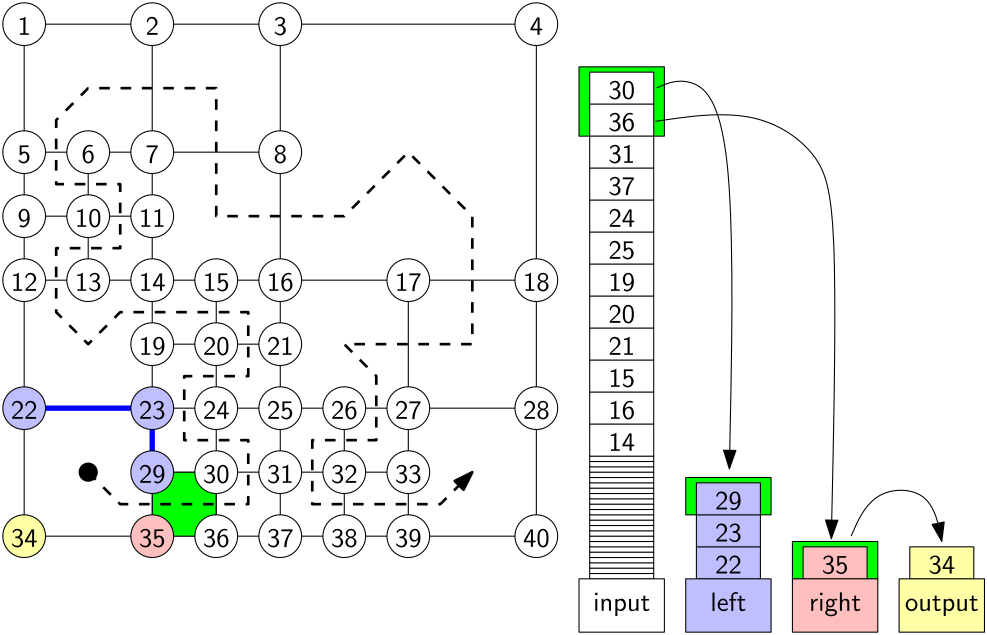




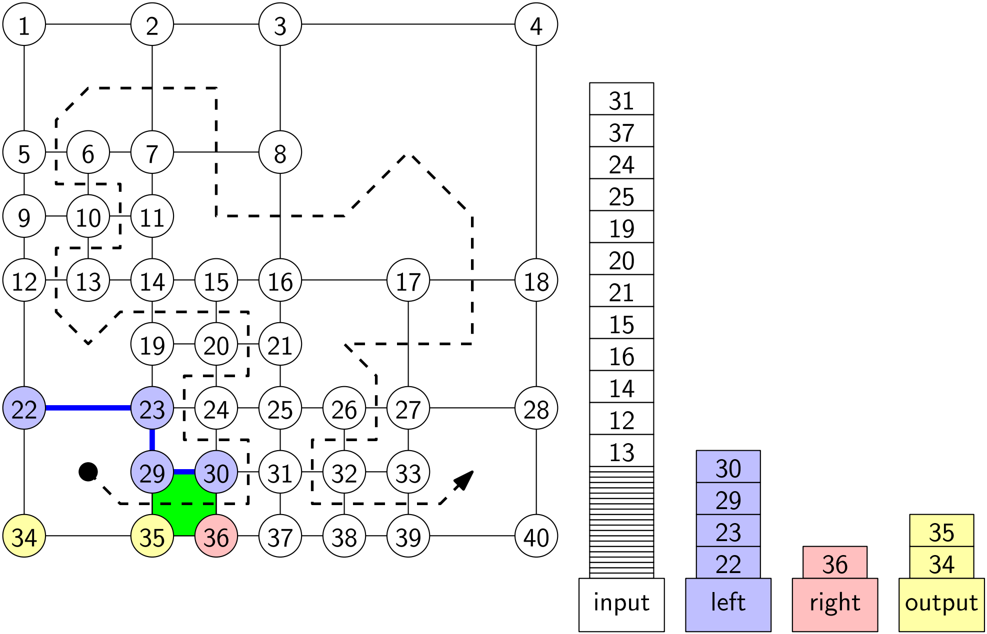
Hilbert order



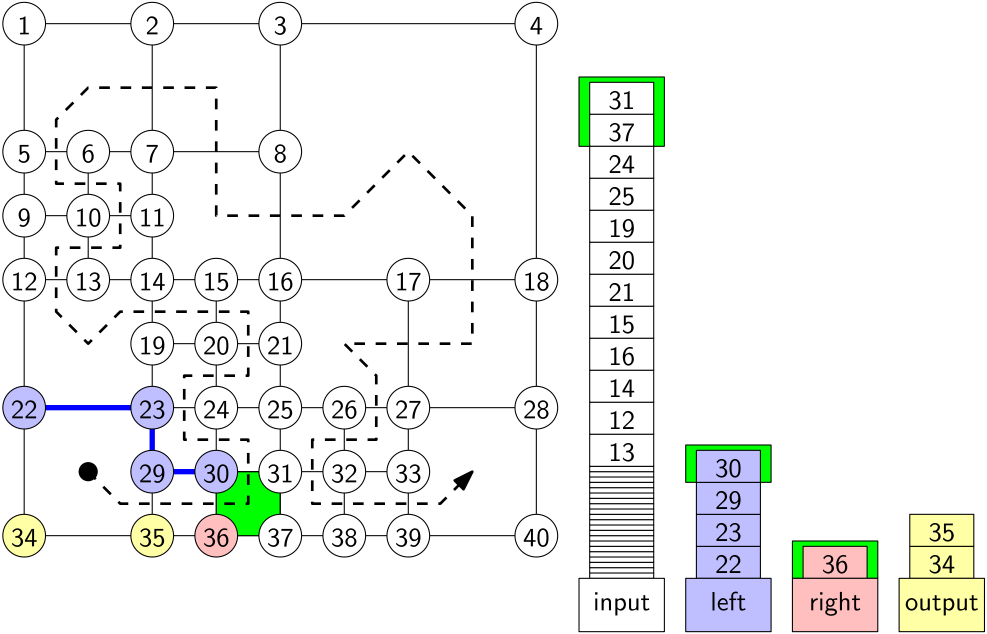
Traversing an irregular grid



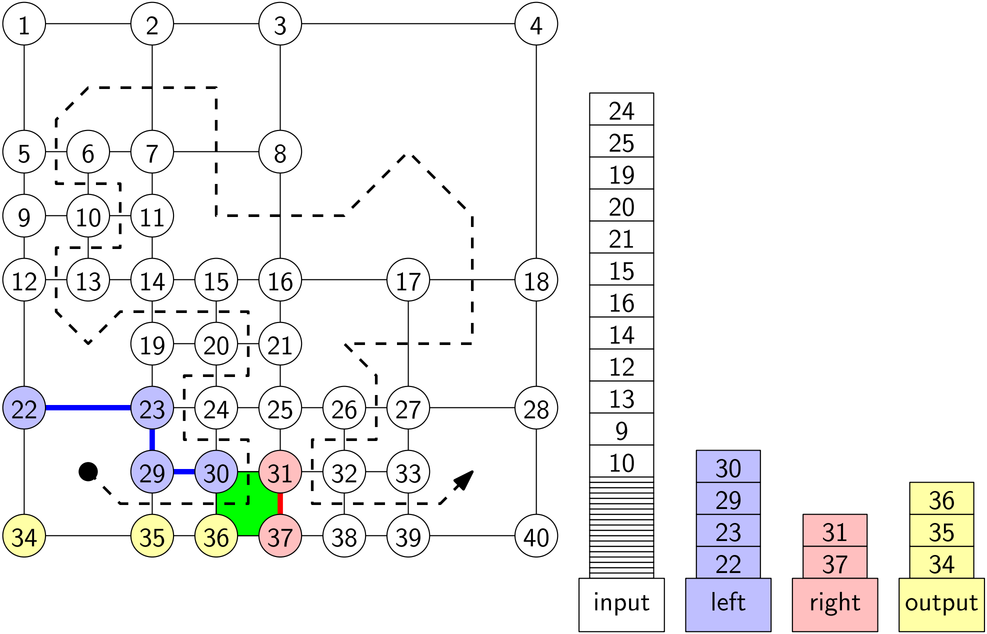
Traversing an irregular grid



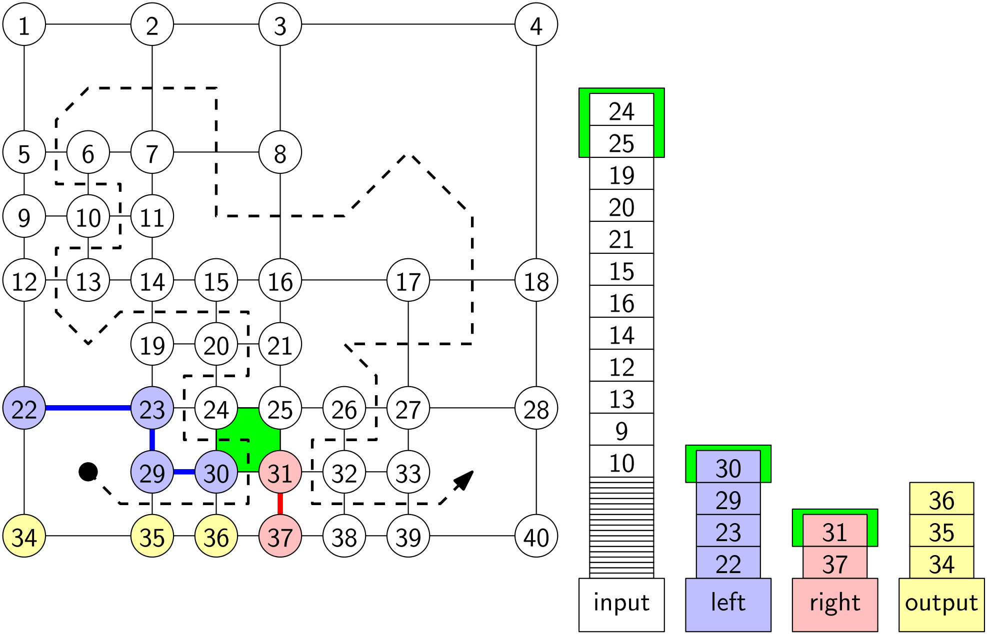
Traversing an irregular grid



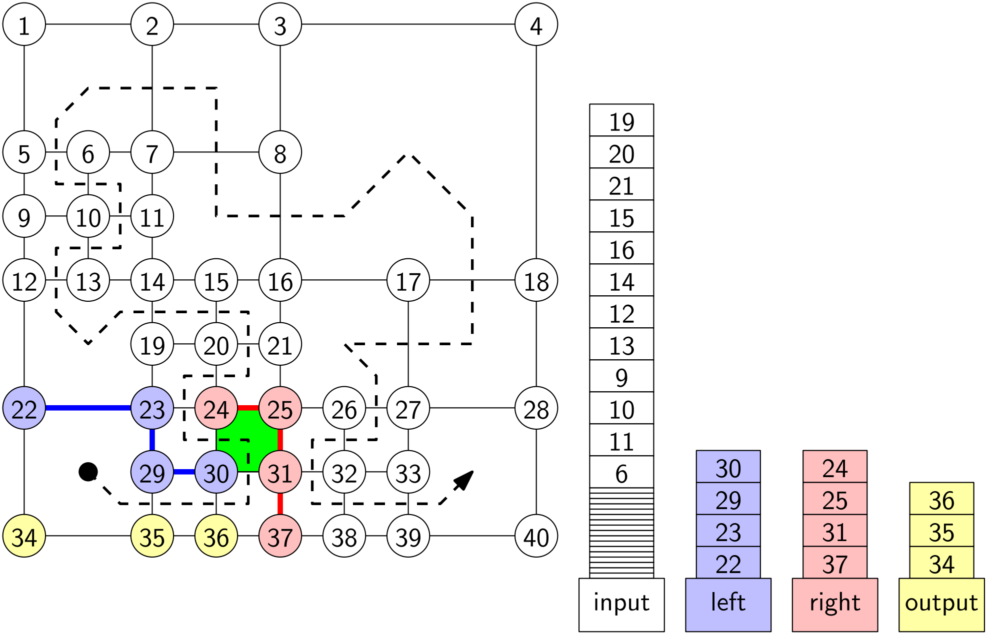
Traversing an irregular grid



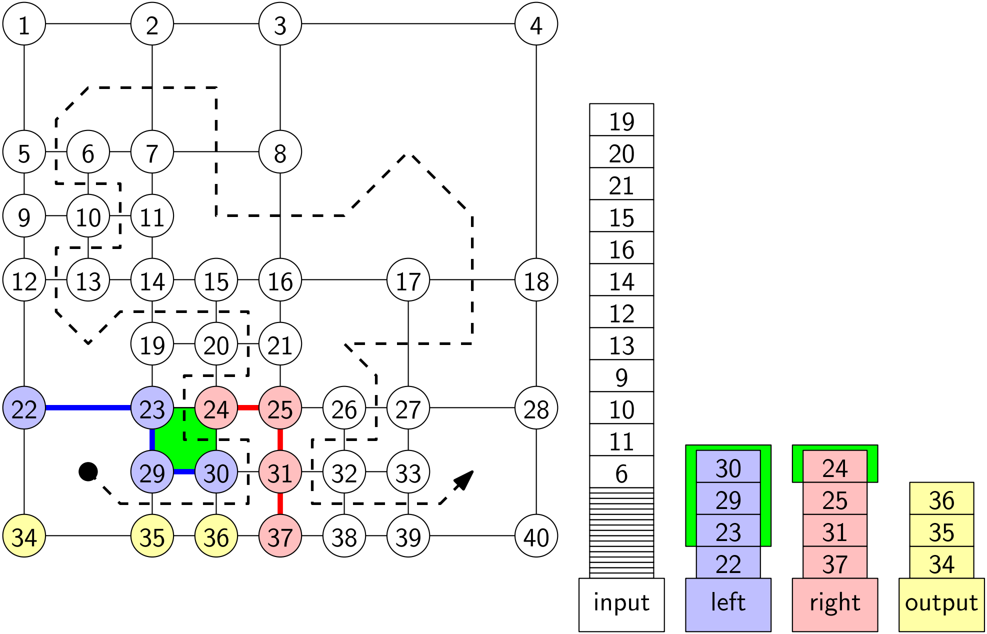
Traversing an irregular grid



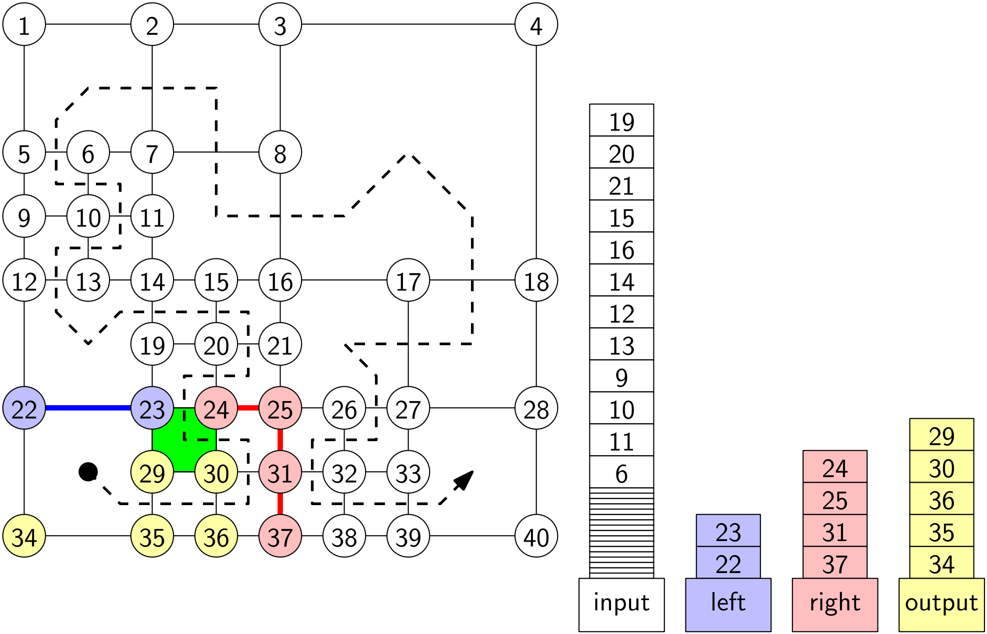
Traversing an irregular grid



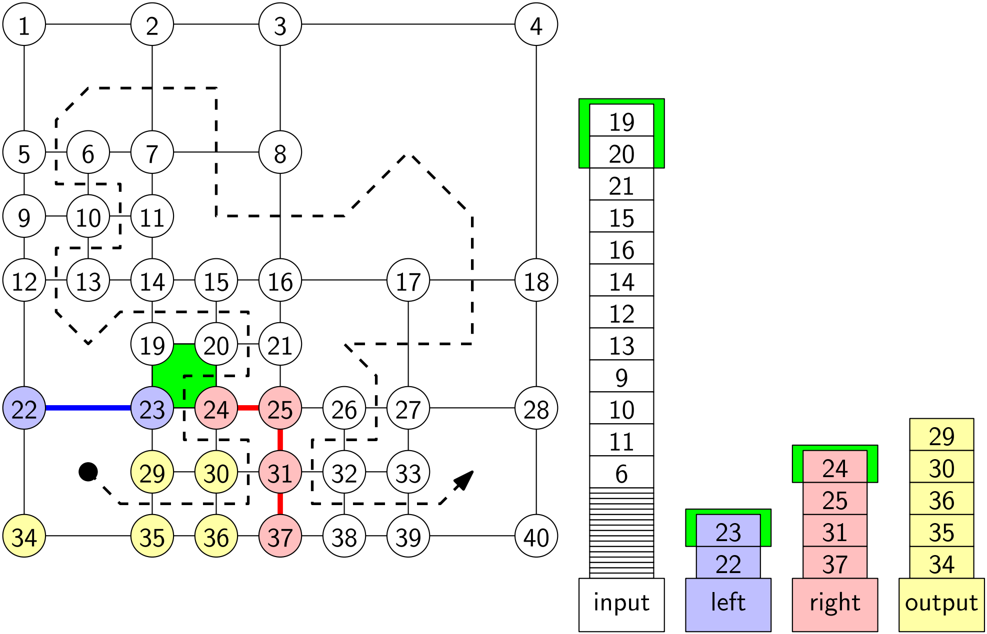
Traversing an irregular grid



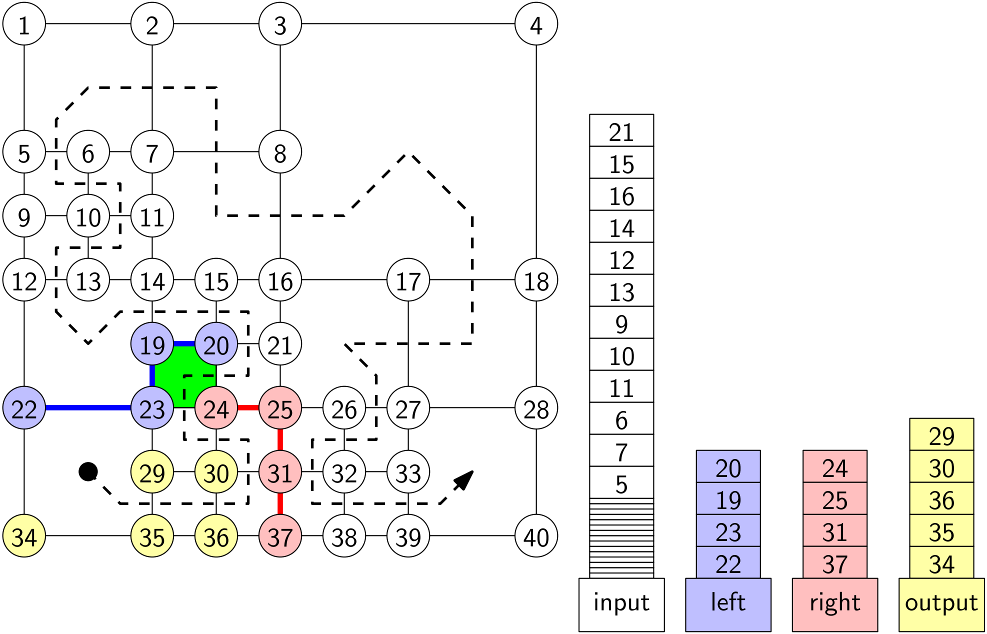
Traversing an irregular grid



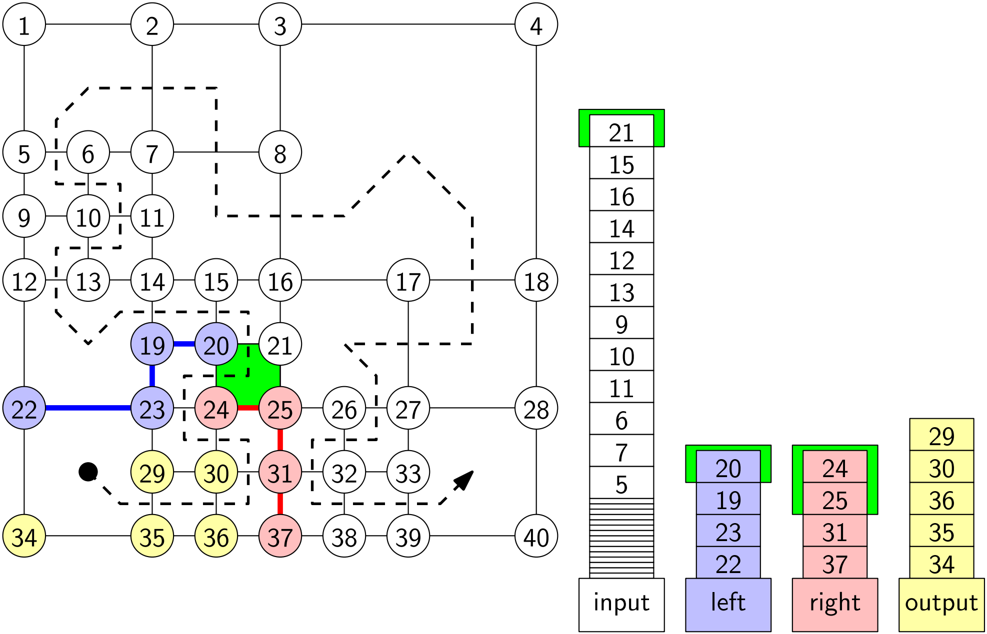
Traversing an irregular grid



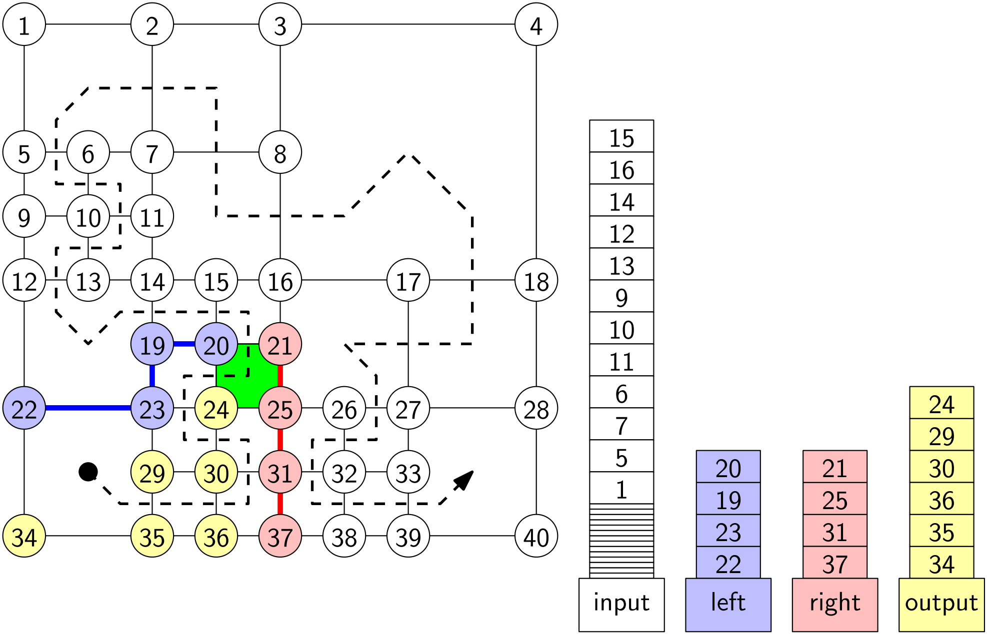
Traversing an irregular grid



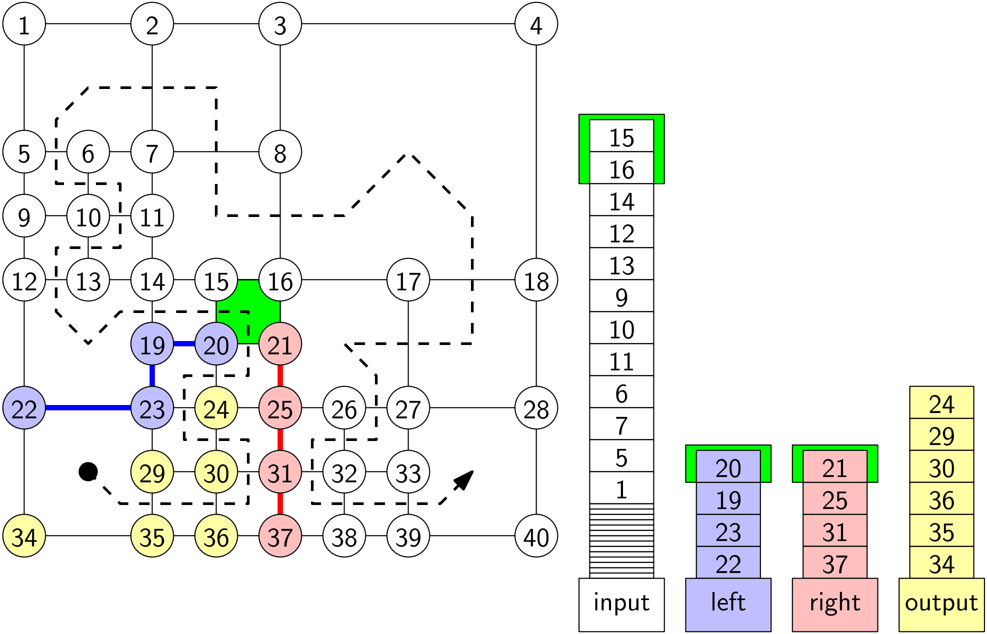
Traversing an irregular grid



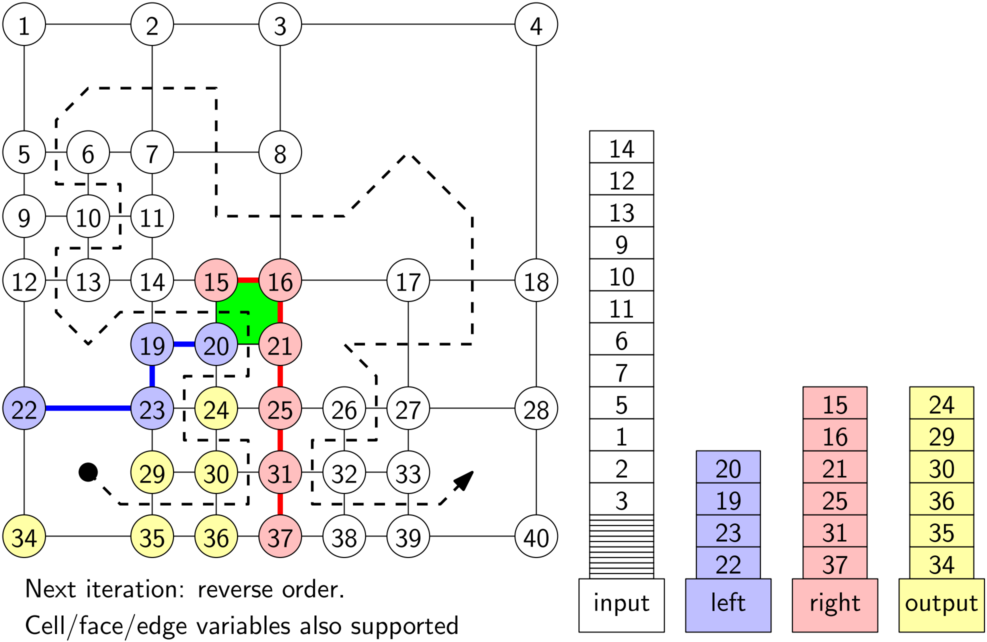
Traversing an irregular grid



Traversing an irregular grid

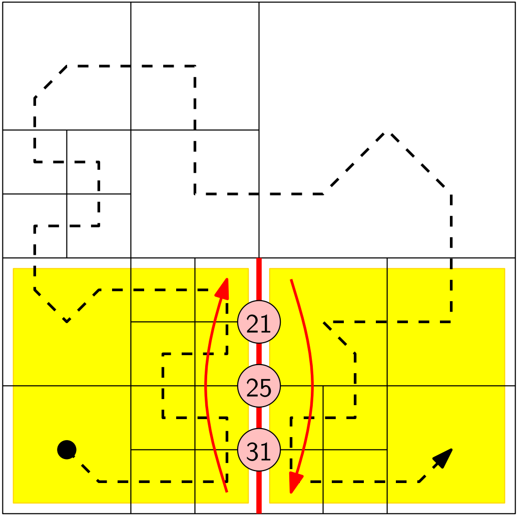


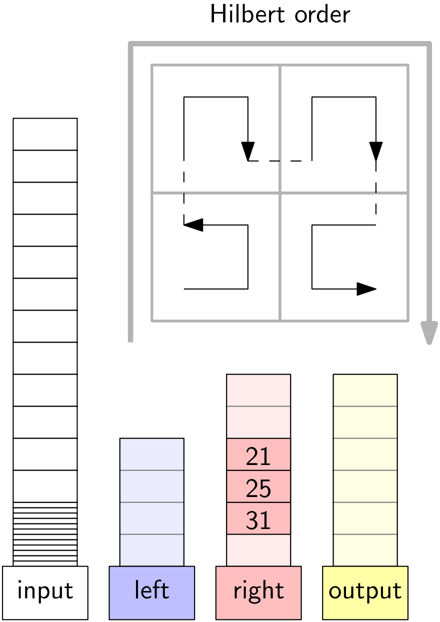
Traversing an irregular grid



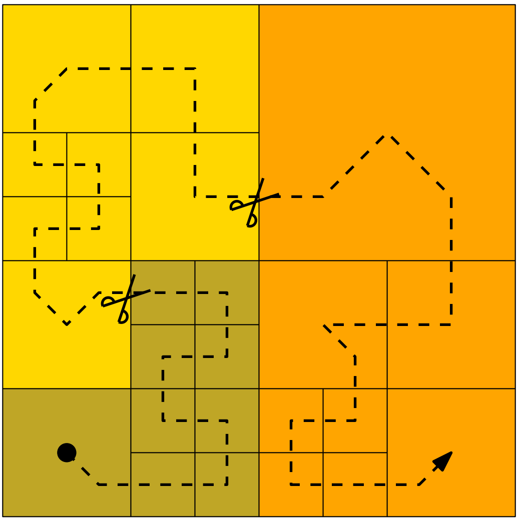
Next iteration: reverse order.

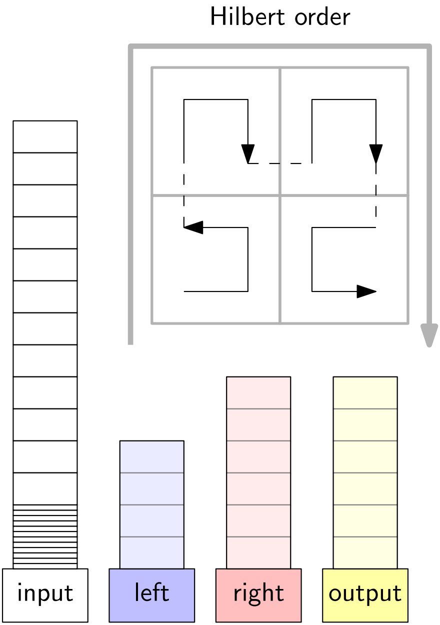
Cell/face/edge variables also supported



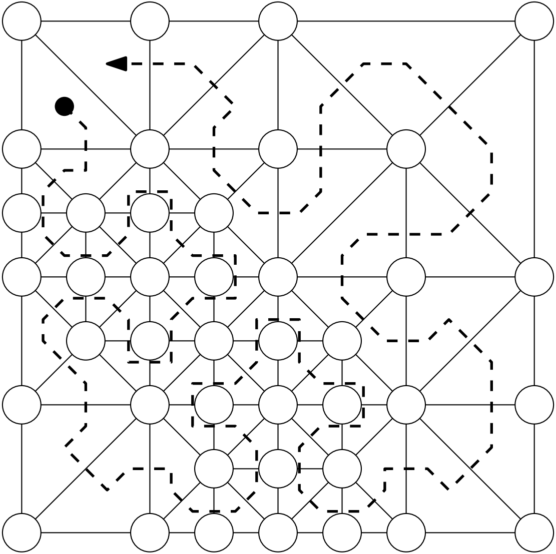


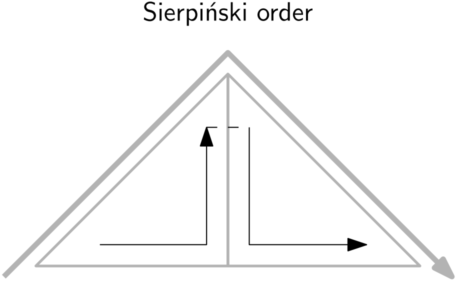
On any edge between two quadrants
order 2nd time = reverse order 1st time



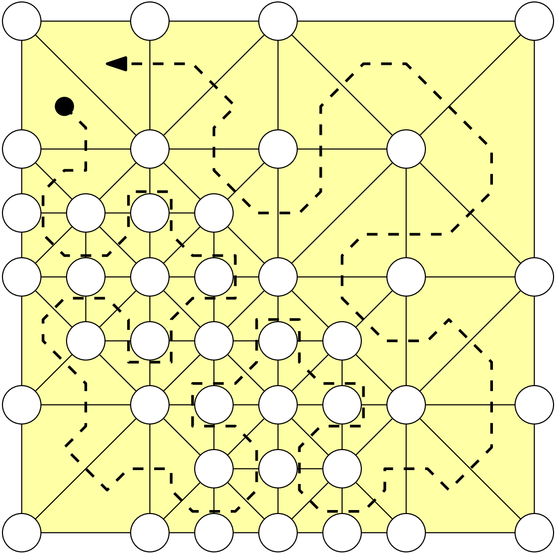


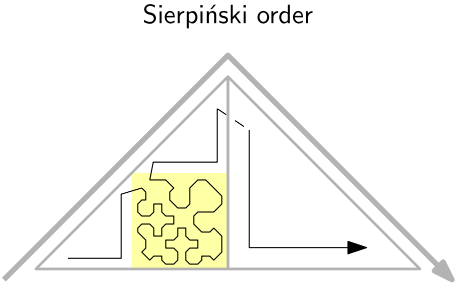
Consecutive squares always share an edge
→(?) well-shaped partitions for load balancing




Sierpiński order

Traversing an irregular grid

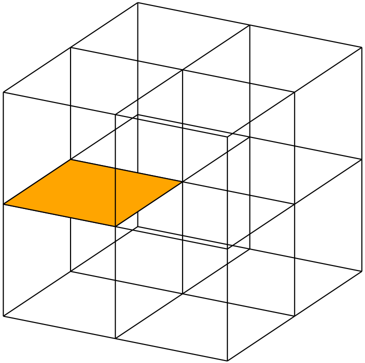




What about 3D?

Desiderata:

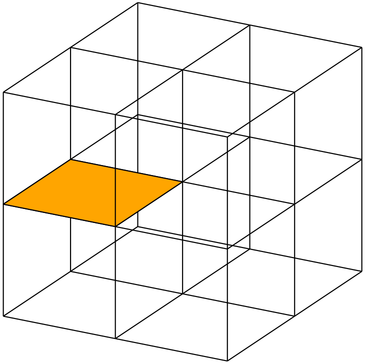
- *octree traversal* (cubes/simplices recursively subdivided into 8 parts)
- *face-continuous*: consecutive cells share a face
- *palindromic*: for each pair of adjacent cubes/simplices C_1 and C_2,
  second traversal of common face is reverse of first traversal



What about 3D?

Desiderata:

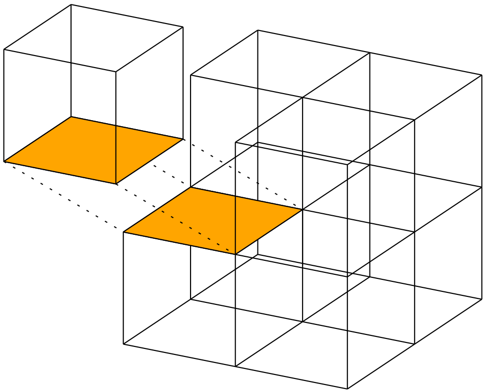
- *octree traversal* (cubes/simplices recursively subdivided into 8 parts)
- *face-continuous*: consecutive cells share a face
- *palindromic*: for every **set** of cubes/simplices sharing a common face/**edge**,
  every subseq. traversal of common face/**edge** is reverse of **previous** traversal



What about 3D?

Desiderata:

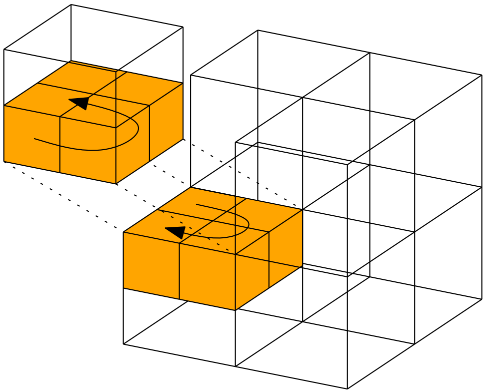
- *octree traversal* (cubes/simplices recursively subdivided into 8 parts)
- *face-continuous*: consecutive cells share a face
- *palindromic*: for every **set** of cubes/simplices sharing a common face/**edge**,
  every subseq. traversal of common face/**edge** is reverse of **previous** traversal



What about 3D?

Desiderata:

- *octree traversal* (cubes/simplices recursively subdivided into 8 parts)
- *face-continuous*: consecutive cells share a face
- *palindromic*: for every **set** of cubes/simplices sharing a common face/**edge**,
  every subseq. traversal of common face/**edge** is reverse of **previous** traversal



What about cubes?

Desired: palindromic, face-continuous octree traversal

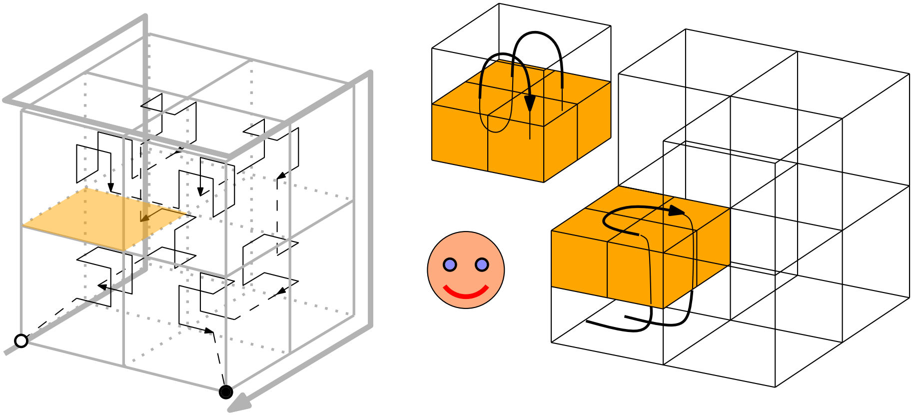
Bader 2013: finds no solution within a restricted framework

What about cubes?

Desired: palindromic, face-continuous octree traversal

Bader 2013: finds no solution within a restricted framework

Weinzierl (pers. conf.):

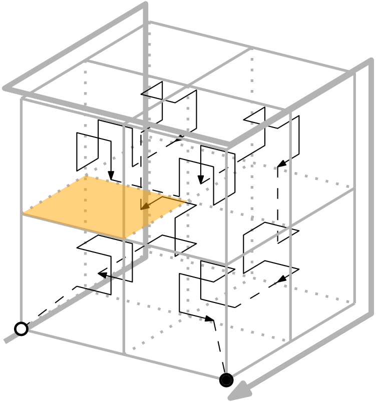


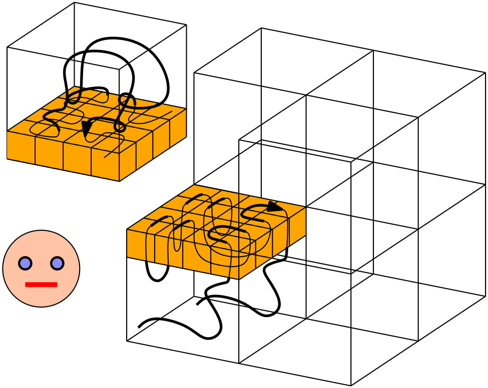
What about cubes?

Desired: palindromic, face-continuous octree traversal

BADER 2013: finds no solution within a restricted framework

WEINZIERL (PERS. CONF.):



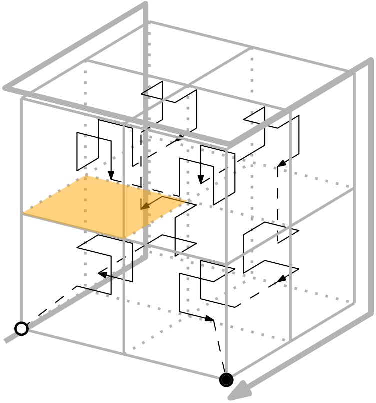


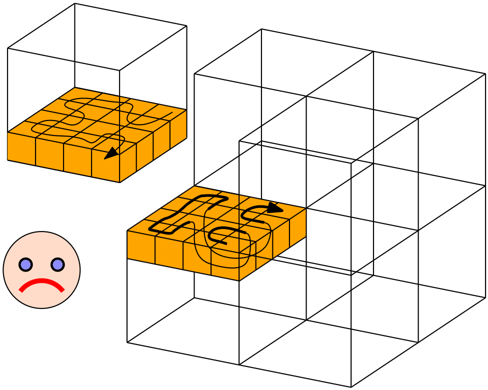
What about cubes?

Desired: palindromic, face-continuous octree traversal

BADER 2013: finds no solution within a restricted framework

WEINZIERL (PERS. CONF.):

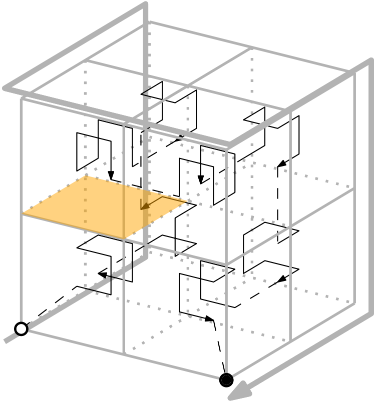


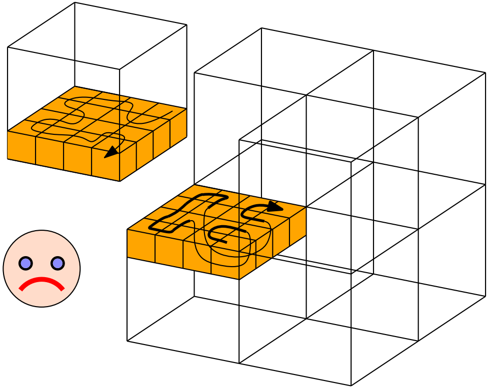


Desired: palindromic, face-continuous octree traversal

Bader 2013: finds no solution within a restricted framework

Weinzierl (pers. conf.):





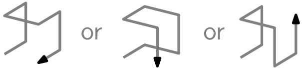
Brute-force search $4 \times 4 \times 4$ finds solutions, but all self-similar recursive expansions fail.

Brute-force search $8 \times 8 \times 8$ is infeasible (and would still generate false positives).

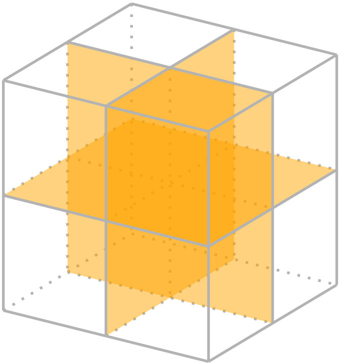
What about cubes?

Generate $S =$ all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one;



- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.

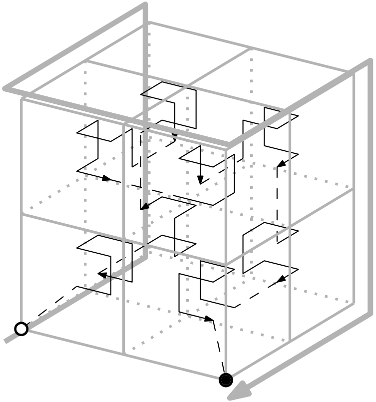


What about cubes?

Generate $S =$ all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one;
- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.

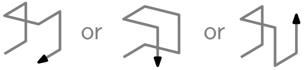
Finds 8384, for example:



What about cubes?

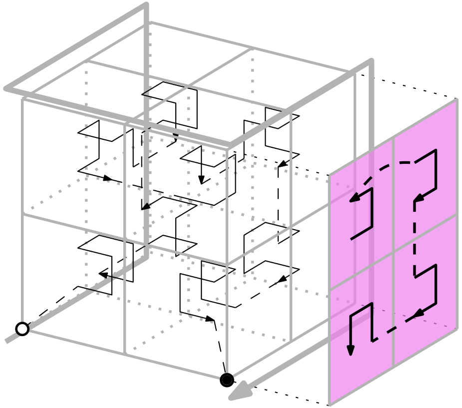
Generate S = all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one; ←



- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.

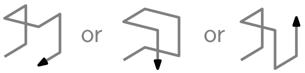
Finds 8384, for example:



What about cubes?

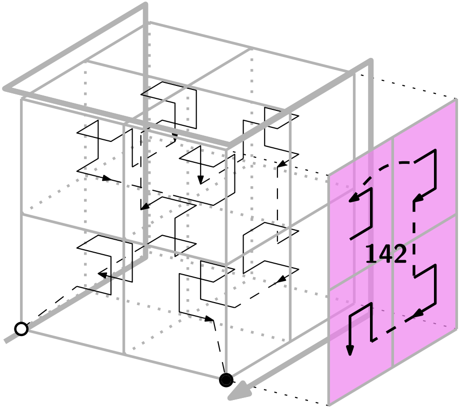
Generate S = all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one;



- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.

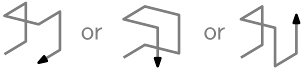
Finds 8384, for example:



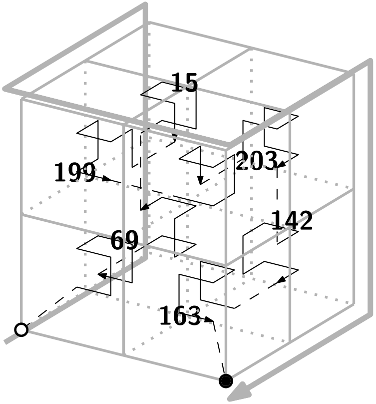
Identify face patterns by id $\in \{0, ..., 255\}$

Generate $S =$ all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one;
- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.



Finds 8384, for example:

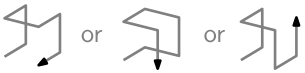


Identify face patterns by id $\in \{0, ..., 255\}$

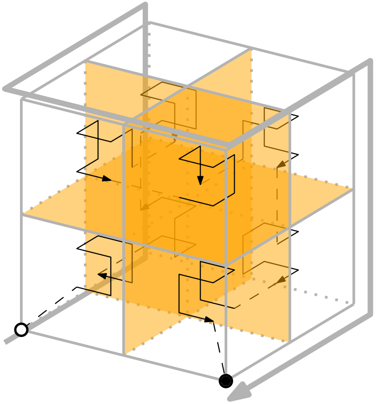
What about cubes?

Generate $S =$ all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one;
- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.



Finds 8384, for example:



$S' =$

$48 \times 8384 = 402\,432$

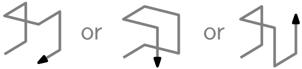
traversals obtained by reflections and rotations

Identify face patterns by id $\in \{0, ..., 255\}$

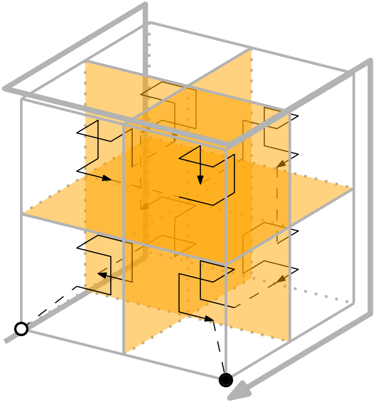
What about cubes?

Generate S = all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one;
- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.



Finds 8384, for example:



$S' =$

$48 \times 8384 = 402\,432$

traversals obtained by reflections and rotations

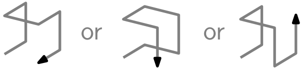
Identify face patterns by id $\in \{0, \ldots, 255\}$

Try to refine $4 \times 4 \times 4$ traversals from S into $8 \times 8 \times 8$ traversals by replacing $2 \times 2 \times 2$ traversals in octants by $4 \times 4 \times 4$ traversals from S'

What about cubes?

Generate S = all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one;
- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.

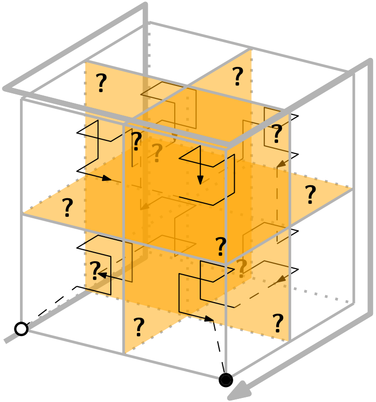


Finds 8384, for example:

$S' =$

$48 \times 8384 = 402\,432$

traversals obtained by reflections and rotations



Identify face patterns by id $\in \{0, ..., 255\}$

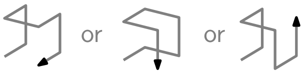
For each traversal from S:

exhaustive search of all choices of patterns $\in \{0, ..., 255\}$ for 12 interior faces:

What about cubes?

Generate S = all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one;



- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.

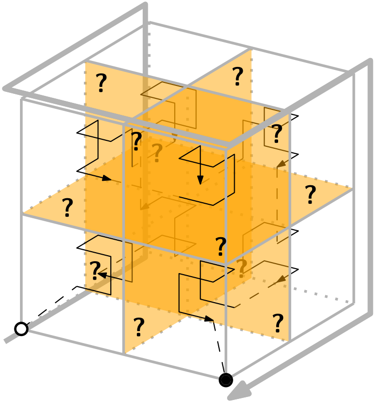
Finds 8384, for example:

$S' =$

$48 \times 8384 = 402\,432$

traversals obtained by

reflections and rotations



Identify face patterns by id $\in \{0, ..., 255\}$

For each traversal from S:

exhaustive search of all choices of patterns $\in \{0, ..., 255\}$ for 12 interior faces:

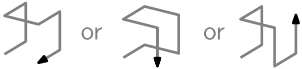
verify for each octant:

S' contains $4 \times 4 \times 4$ traversal with matching octant order and face patterns

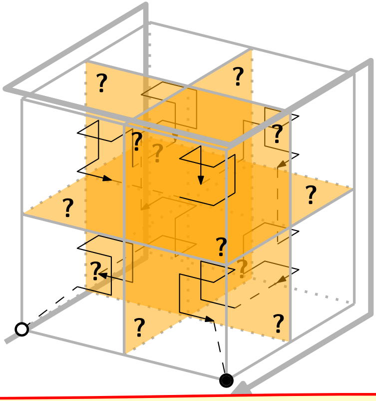
What about cubes?

Generate S = all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one;
- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.



Finds 8384, for example:



$S' =$

$48 \times 8384 = 402\,432$

traversals obtained by reflections and rotations

Identify face patterns by id $\in \{0, ..., 255\}$

unsuccessful for all but 410 traversals from S
(no $8 \times 8 \times 8$-refinement exists)

For each traversal from S:

exhaustive search of all choices of patterns $\in \{0, ..., 255\}$ for 12 interior faces:

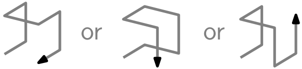
verify for each octant:

S' contains $4 \times 4 \times 4$ traversal with matching octant order and face patterns

What about cubes?

Generate S = all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one;
- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.

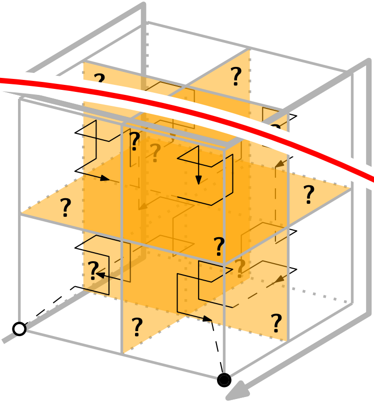


Finds ~~8384~~ **410**, for example:

S' = **410** **19 680**

48 × ~~8384~~ = ~~402 432~~

traversals obtained by reflections and rotations



Identify face patterns by id $\in \{0, ..., 255\}$

unsuccessful for all but 410 traversals from S
(no $8 \times 8 \times 8$-refinement exists)

For each traversal from S:

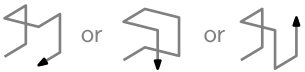
exhaustive search of all choices of patterns $\in \{0, ..., 255\}$ for 12 interior faces:

verify for each octant:

S' contains $4 \times 4 \times 4$ traversal with matching octant order and face patterns

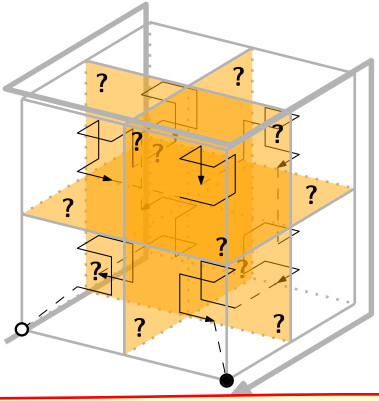
Generate S = all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one;



- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.

Finds ~~8384~~ **410**, for example:



$S' =$ **410** **19 680**
$48 \times \cancel{8384} = \cancel{402\,432}$
traversals obtained by reflections and rotations

Identify face patterns by id $\in \{0, ..., 255\}$

unsuccessful for all but 1 traversal from S
(no 16×16×16-refinement)

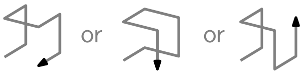
For each traversal from S:

exhaustive search of all choices of patterns $\in \{0, ..., 255\}$ for 12 interior faces:
  verify for each octant:
    S' contains $4 \times 4 \times 4$ traversal with matching octant order and face patterns

What about cubes?

Generate S = all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one;
- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.

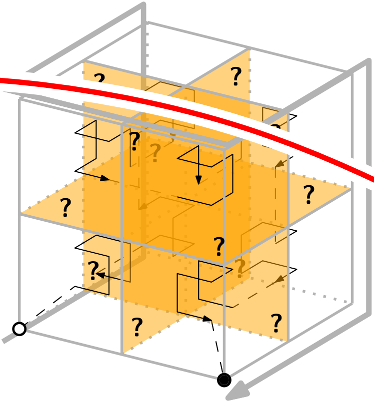


Finds ~~8384~~ ~~410~~ 1, for example:

$S' =$ ~~410~~ 48 ~~19 680~~

48 × ~~8384~~ = ~~402 432~~

traversals obtained by reflections and rotations



Identify face patterns by id $\in \{0, ..., 255\}$

unsuccessful for all but 1 traversal from S
(no 16×16×16-refinement)

For each traversal from S:

exhaustive search of all choices of patterns $\in \{0, ..., 255\}$ for 12 interior faces:

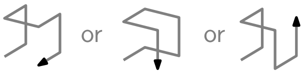
verify for each octant:

S' contains $4 \times 4 \times 4$ traversal with matching octant order and face patterns

What about cubes?

Generate S = all $4 \times 4 \times 4$ traversals that:

- traverse octants one by one;
- are face-continuous;
- have matching patterns on opposite sides of each of the **twelve interior 2×2-faces**.



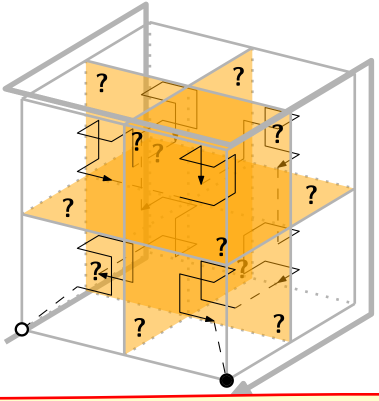
Finds ~~8384~~ ~~410~~ 1, for example:

48

$S' = \cancel{410}$ $\cancel{19\,680}$

$48 \times \cancel{8384} = \cancel{402\,432}$

traversals obtained by reflections and rotations



Identify face patterns by id $\in \{0, ..., 255\}$

no solution found →

no palindr., face-contin. octree traversal exists

For each traversal from S:

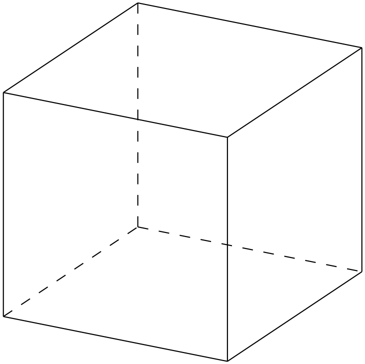
exhaustive search of all choices of patterns $\in \{0, ..., 255\}$ for 12 interior faces:

verify for each octant:

S' contains $4 \times 4 \times 4$ traversal with matching octant order and face patterns

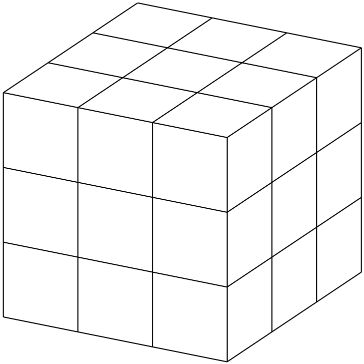
What about cubes?

3D Peano curve (applied by WEINZIERL 2009):



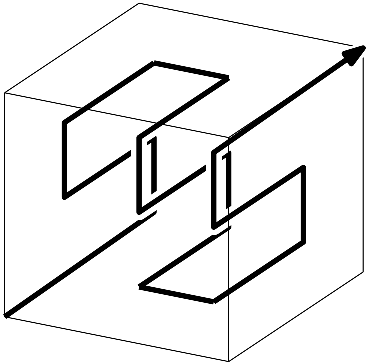
What about cubes?

3D Peano curve (applied by WEINZIERL 2009):

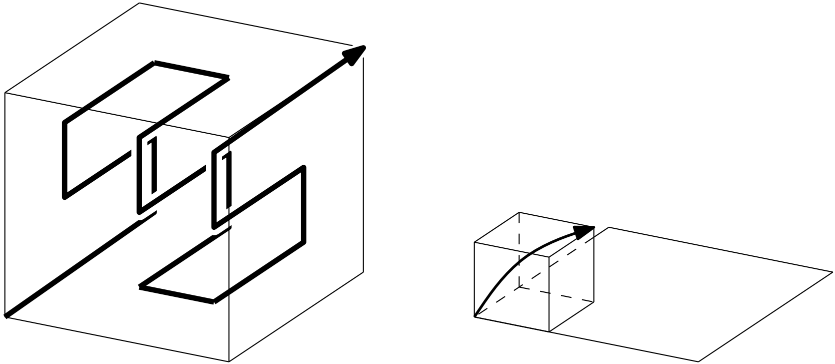


What about cubes?

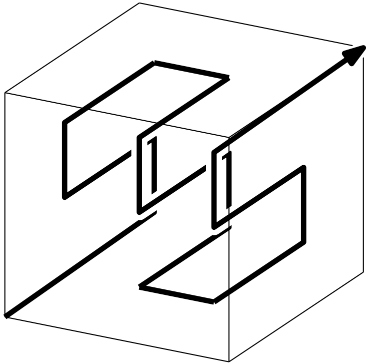
3D Peano curve (applied by WEINZIERL 2009):

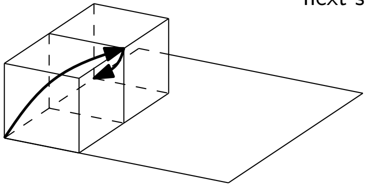


3D Peano curve (applied by WEINZIERL 2009):



3D Peano curve (applied by WEINZIERL 2009):

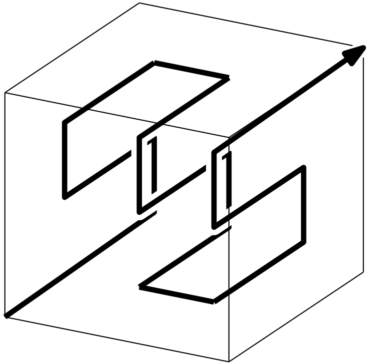


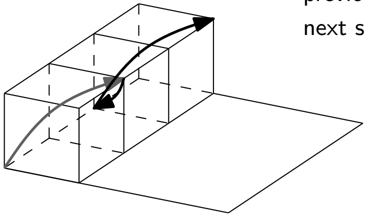


in each subcube, reflect pattern to connect to previous and next subcube

What about cubes?

3D Peano curve (applied by WEINZIERL 2009):

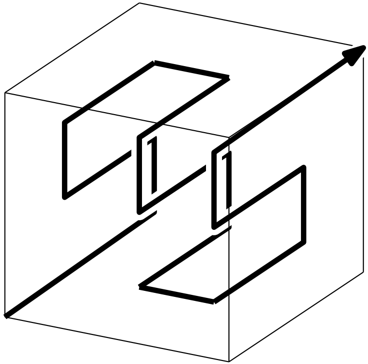


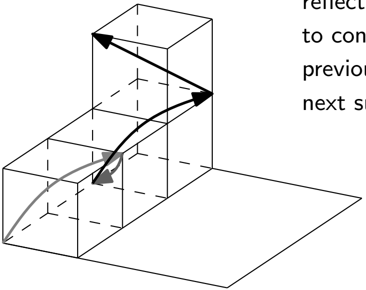


in each subcube, reflect pattern to connect to previous and next subcube

What about cubes?

3D Peano curve (applied by WEINZIERL 2009):

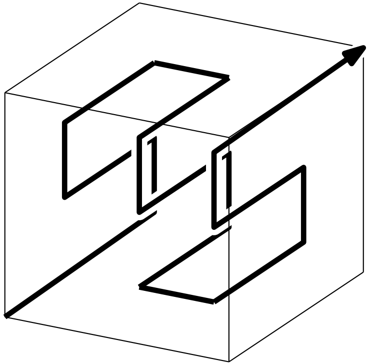


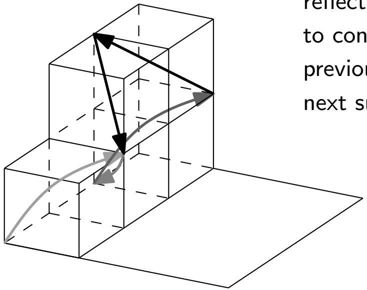


in each subcube, reflect pattern to connect to previous and next subcube

What about cubes?

3D Peano curve (applied by WEINZIERL 2009):

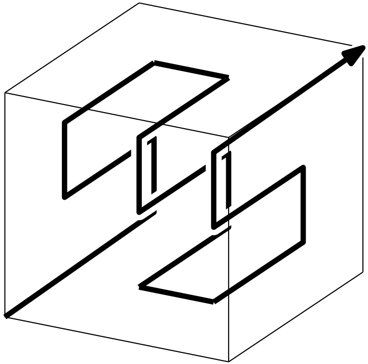


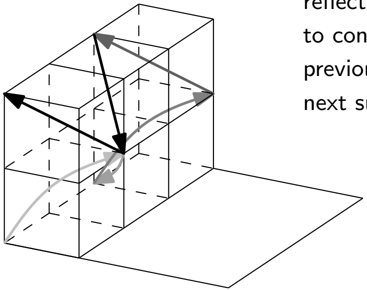


in each subcube, reflect pattern to connect to previous and next subcube

What about cubes?

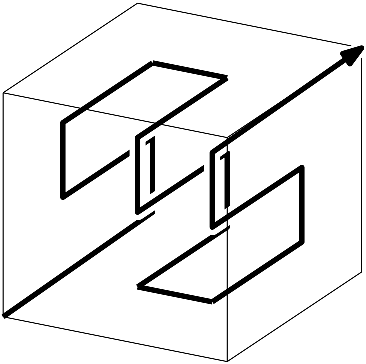
3D Peano curve (applied by WEINZIERL 2009):

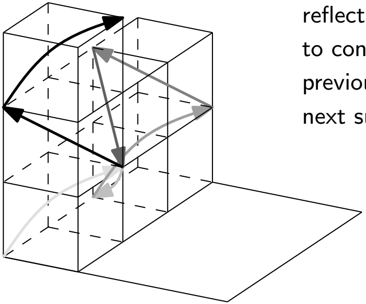




in each subcube, reflect pattern to connect to previous and next subcube

3D Peano curve (applied by WEINZIERL 2009):

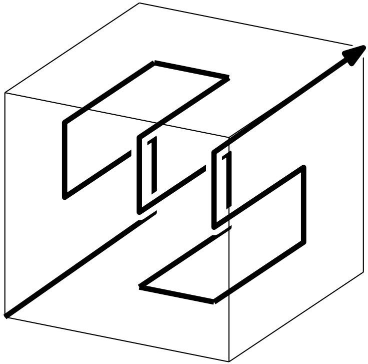


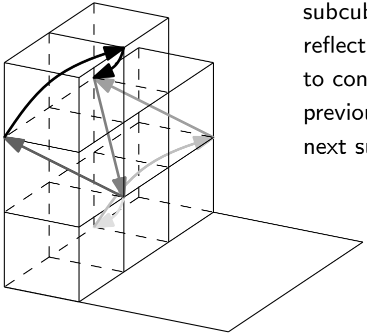


in each subcube, reflect pattern to connect to previous and next subcube

What about cubes?

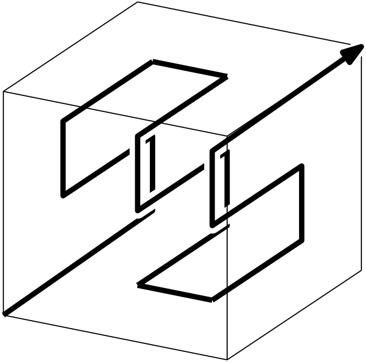
3D Peano curve (applied by WEINZIERL 2009):

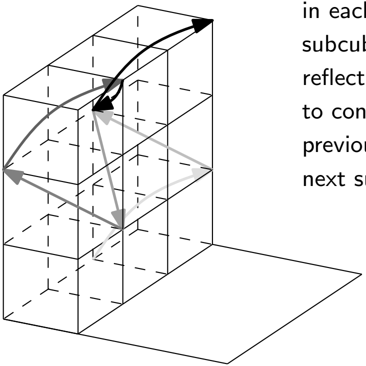




in each subcube, reflect pattern to connect to previous and next subcube

3D Peano curve (applied by WEINZIERL 2009):

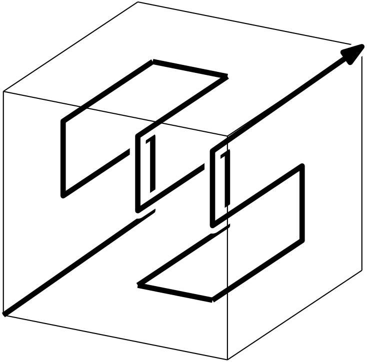


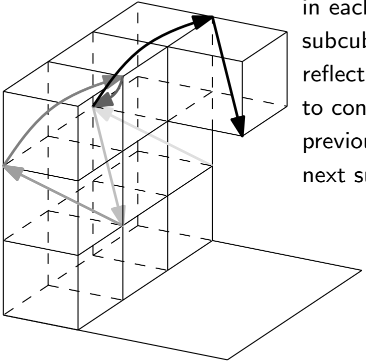


in each subcube, reflect pattern to connect to previous and next subcube

What about cubes?

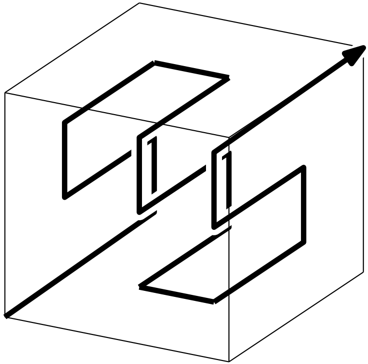
3D Peano curve (applied by WEINZIERL 2009):

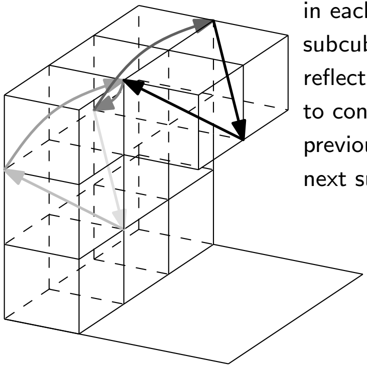




in each subcube, reflect pattern to connect to previous and next subcube

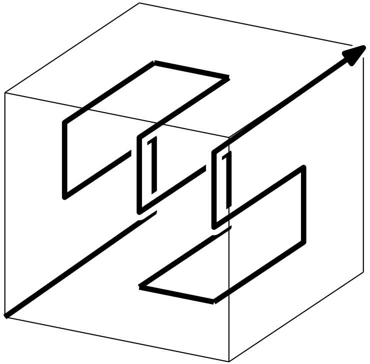
3D Peano curve (applied by WEINZIERL 2009):

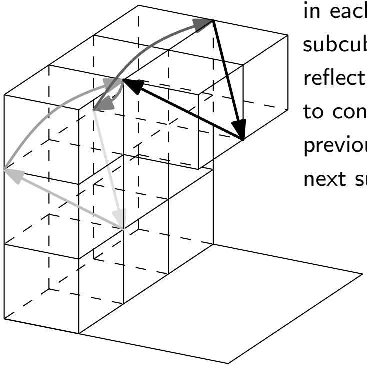




in each subcube, reflect pattern to connect to previous and next subcube

3D Peano curve (applied by WEINZIERL 2009):





in each subcube, reflect pattern to connect to previous and next subcube

Desired: palindromic, face-continuous ~~oct~~ree traversal

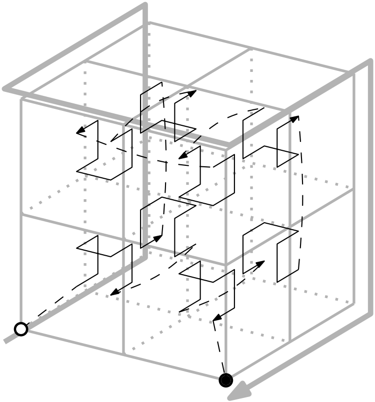
Cons:

- recursive subdivision into 27 subcubes → adaptive refinement less adaptive;
- partitions have larger surface-for-volume than with octree traversals (SASBURG 2011).

What about cubes?

Faloutsos's traversal (generalized from Faloutsos 1986)

Octants A and B share face f $\rightarrow$ traversal in B is reversed image of A under reflection in f

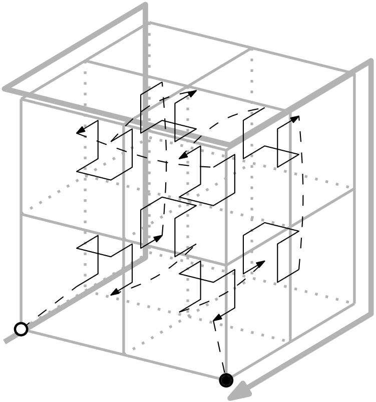


Desired: palindromic, ~~face-continuous~~ octree traversal

What about cubes?

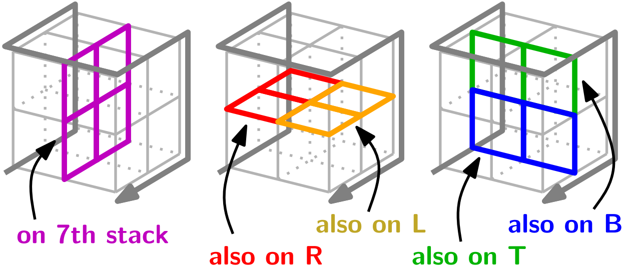
Faloutsos's traversal (generalized from FALOUTSOS 1986)

Octants A and B share face f $\rightarrow$ traversal in B is reversed image of A under reflection in f



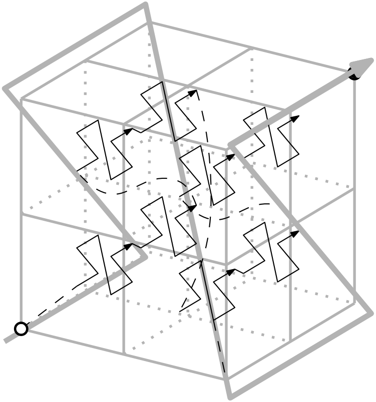
L: left
R: right
B: bottom
T: top
F: front
H: hind

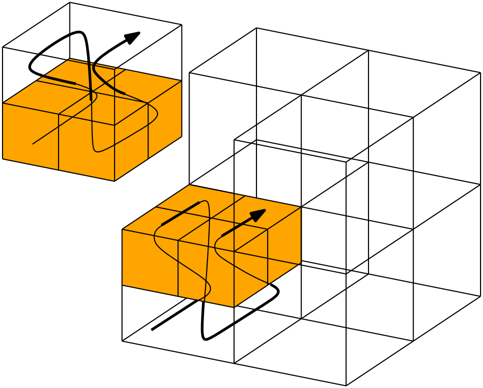
stack for $\frac{\text{pushing}}{\text{popping}}$ vertices on (left / right / bottom / top / front / hind) face



Desired: palindromic, ~~face-continuous~~ octree traversal

Morton order (Morton 1966): no rotations or reflections

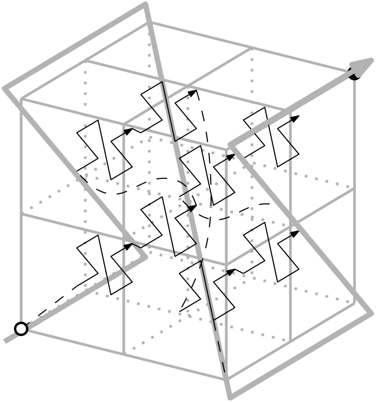


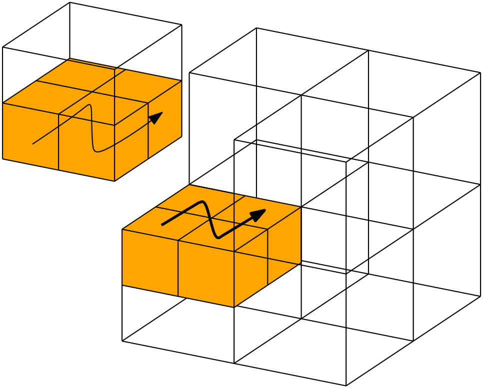


Desired: ~~palindromic~~, ~~face-continuous~~ octree traversal

What about cubes?

Morton order (Morton 1966): no rotations or reflections

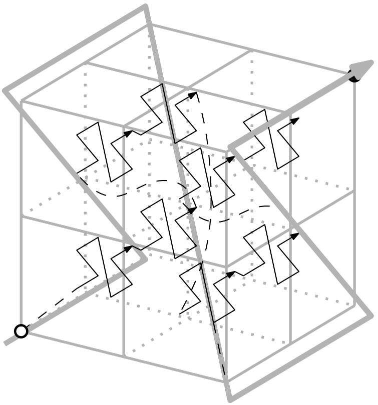


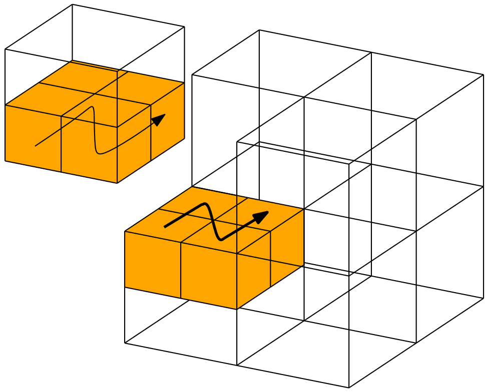


Desired: ~~palindromic~~, ~~face-continuous~~ octree traversal

What about cubes?

Morton order (Morton 1966): no rotations or reflections

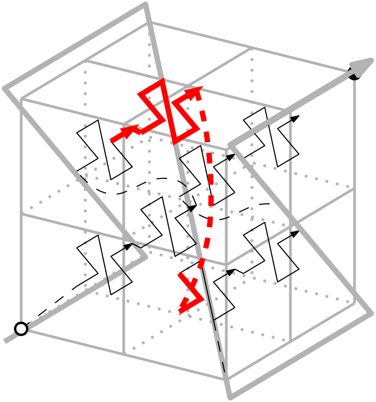


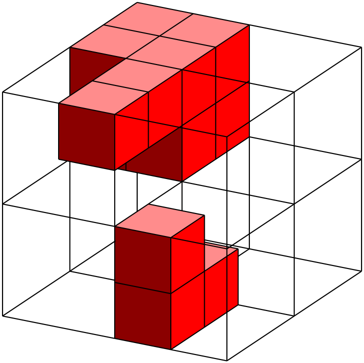


homodromic: 2nd traversal of face
= reverse of 1st *or the same*

Desired: ~~palindromic~~, ~~face-continuous~~ octree traversal

Morton order (MORTON 1966): no rotations or reflections





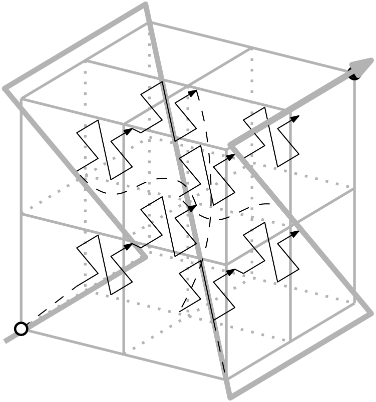
homodromic: 2nd traversal of face
= reverse of 1st *or the same*

quasi-face-continuous: interior of union of consecutive set of cells has $O(1)$ connected components.

Desired: ~~palindromic~~, ~~face-continuous~~ octree traversal

What about cubes?

Morton order (Morton 1966): no rotations or reflections

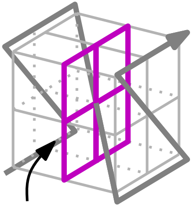


L:
R:
B:
T:
F:
H:

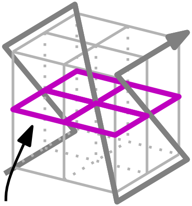
stack for $\frac{\text{pushing}}{\text{popping}}$ vertices on

left
right
bottom
top
front
hind

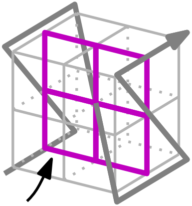
face



pushed onto R;
turned onto L
after 4th octant



pushed onto T;
turned onto B
aft. 2nd/6th oct.



pushed onto H;
turned onto F
aft. 1/3/5/7th

homodromic: 2nd traversal of face
= reverse of 1st *or the same*

quasi-face-continuous: interior of union of consecutive set of cells
has $O(1)$ connected components.

Desired: ~~palindromic~~, ~~face-continuous~~ octree traversal

What about cubes?

Desiderata:

- *octree traversal* (cubes/simplices recursively subdivided into 8 parts)
- *face-continuous*: consecutive cells share a face

  altern.: *quasi-face-cont.*: interior of union of set of consecutive cells has $O(1)$ components
- *palindromic*: for every set of adjacent cubes/simplices sharing a common face/edge, every subseq. traversal of common face/edge is reverse of previous traversal

  altern.: *homodromic*: subsequent traversals are reverse or same (more stack operations required)

octree trav.	non-face-continuous	quasi-face-continuous	face-continuous
non-homodromic			1000s of generalized Hilbert curves
homodromic		Morton traversal	
palindromic	Faloutsos's traversal		**computer says no**

What about cubes?

Desiderata:

- *octree traversal* (cubes/simplices recursively subdivided into 8 parts)
- *face-continuous*: consecutive cells share a face

  altern.: *quasi-face-cont.*: interior of union of set of consecutive cells has $O(1)$ components
- *palindromic*: for every set of adjacent cubes/simplices sharing a common face/edge, every subseq. traversal of common face/edge is reverse of previous traversal

  altern.: *homodromic*: subsequent traversals are reverse or same (more stack operations required)

octree trav.	non-face-continuous	quasi-face-continuous	face-continuous
non-homodromic			1000s of generalized Hilbert curves
homodromic		Morton traversal	**computer says no**
palindromic	Faloutsos's traversal		**computer says no**

What about cubes?

Desiderata:

- *octree traversal* (cubes/simplices recursively subdivided into 8 parts)
- *face-continuous*: consecutive cells share a face

  altern.: *quasi-face-cont.*: interior of union of set of consecutive cells has $O(1)$ components
- *palindromic*: for every set of adjacent cubes/simplices sharing a common face/edge, every subseq. traversal of common face/edge is reverse of previous traversal

  altern.: *homodromic*: subsequent traversals are reverse or same (more stack operations required)

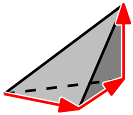
octree trav.	non-face-continuous	quasi-face-continuous	face-continuous
non-homodromic			1000s of generalized Hilbert curves
homodromic		Morton traversal	**computer says no**
palindromic	Faloutsos's traversal	open How to recognize quasi-face-continuous traversals?	**computer says no**

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (MATOUSEK & SAVERNOVA 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after HILL 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:

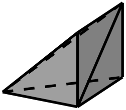


2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:

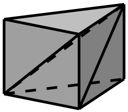


2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (MATOUSEK & SAVERNOVA 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after HILL 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:

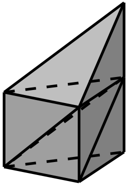


2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:



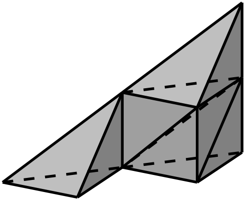
Tetrahedral meshes

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:

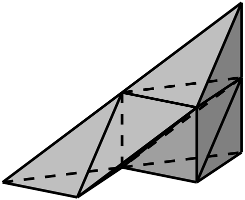


2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:

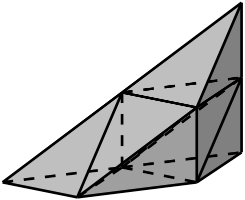


2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (MATOUSEK & SAVERNOVA 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after HILL 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:

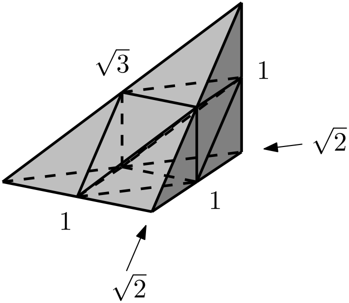


2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:



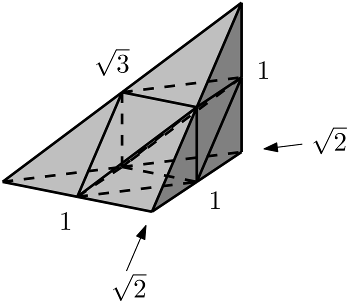
Tetrahedral meshes

2D: any △ can be dissected into four similar (but smaller) triangles.

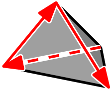
3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (MATOUSEK & SAVERNOVA 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after HILL 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:



$\alpha = \arccos(-1/3)$:



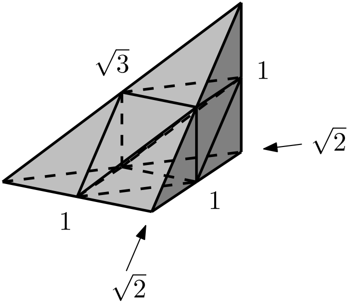
(special cases of Hill tetrahedra)

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

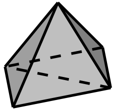
3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (MATOUSEK & SAVERNOVA 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after HILL 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:



$\alpha = \arccos(-1/3)$:



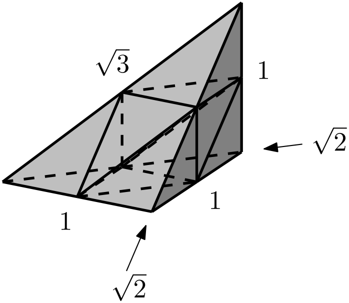
(special cases of Hill tetrahedra)

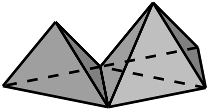
2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (MATOUSEK & SAVERNOVA 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after HILL 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$: $\alpha = \arccos(-1/3)$:





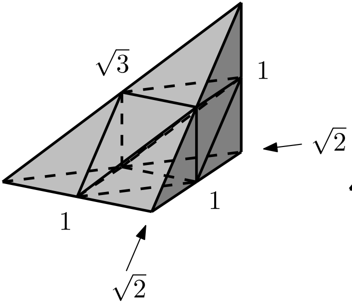
(special cases of Hill tetrahedra)

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

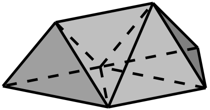
3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:



$\alpha = \arccos(-1/3)$:



(special cases of Hill tetrahedra)

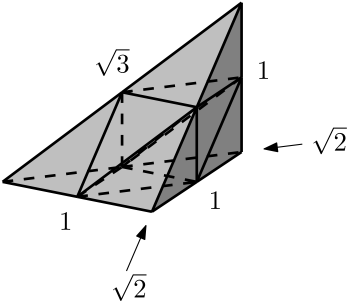
Tetrahedral meshes

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

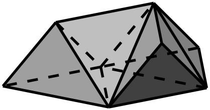
3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:



$\alpha = \arccos(-1/3)$:



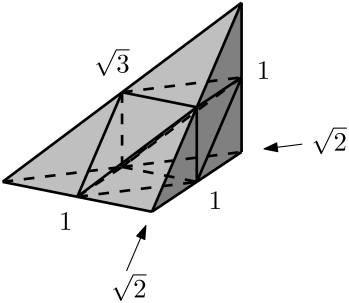
(special cases of Hill tetrahedra)

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

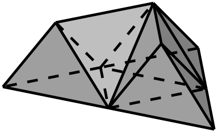
3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:



$\alpha = \arccos(-1/3)$:



(special cases of Hill tetrahedra)

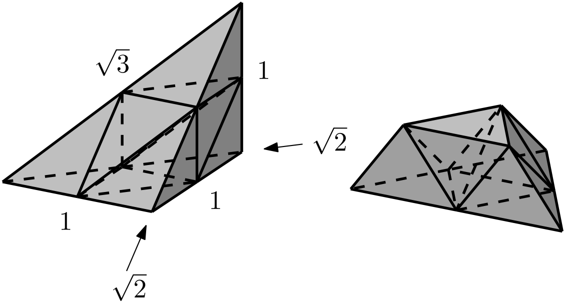
Tetrahedral meshes

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$: $\alpha = \arccos(-1/3)$:

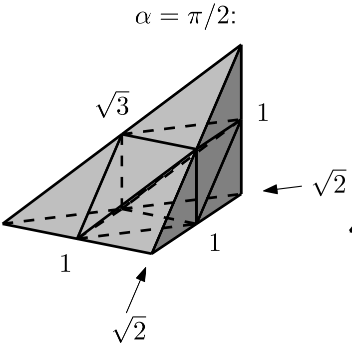


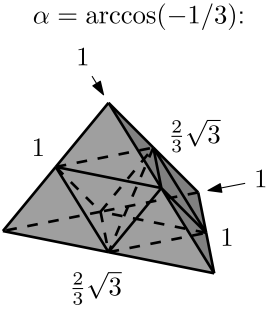
(special cases of Hill tetrahedra)

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair





(special cases of Hill tetrahedra)

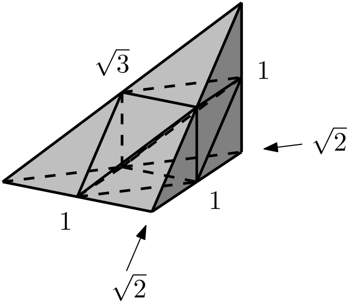
Tetrahedral meshes

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

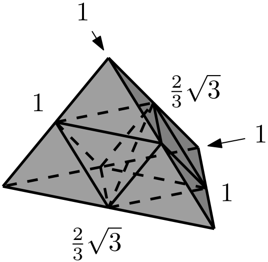
3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:

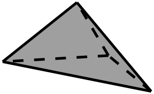


$\alpha = \arccos(-1/3)$:



(special cases of Hill tetrahedra)

non-Hill:



(Liu & Joe 1994)

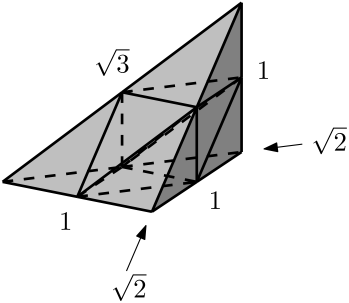
Tetrahedral meshes

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

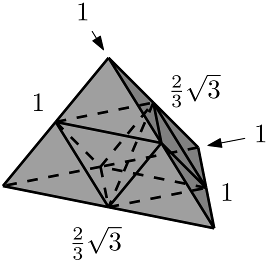
3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:

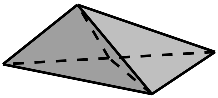


$\alpha = \arccos(-1/3)$:



(special cases of Hill tetrahedra)

non-Hill:



(Liu & Joe 1994)

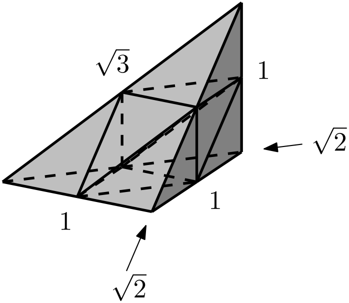
Tetrahedral meshes

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

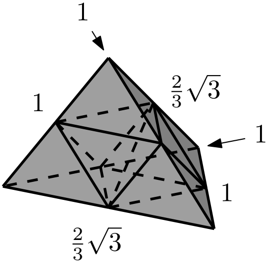
3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (MATOUSEK & SAVERNOVA 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after HILL 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

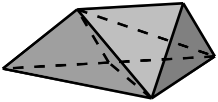
$\alpha = \pi/2$:



$\alpha = \arccos(-1/3)$:



non-Hill:



(special cases of Hill tetrahedra)

(LIU & JOE 1994)

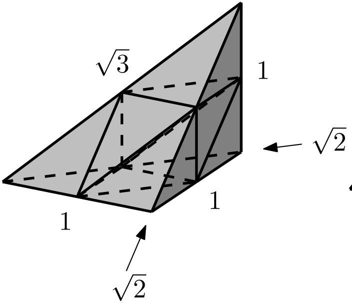
Tetrahedral meshes

2D: any △ can be dissected into four similar (but smaller) triangles.

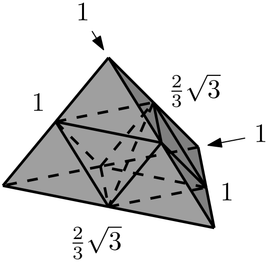
3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

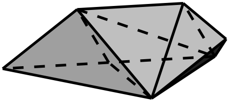
$\alpha = \pi/2$:



$\alpha = \arccos(-1/3)$:



non-Hill:



(special cases of Hill tetrahedra)

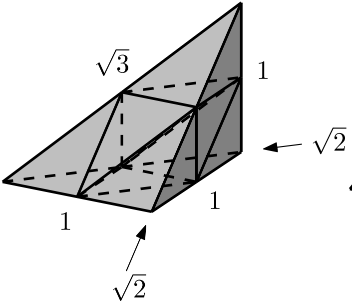
(Liu & Joe 1994)

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

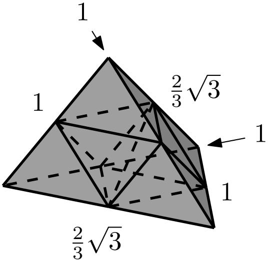
3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

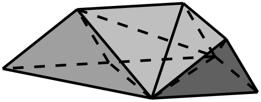
$\alpha = \pi/2$:



$\alpha = \arccos(-1/3)$:



non-Hill:



(special cases of Hill tetrahedra)

(Liu & Joe 1994)

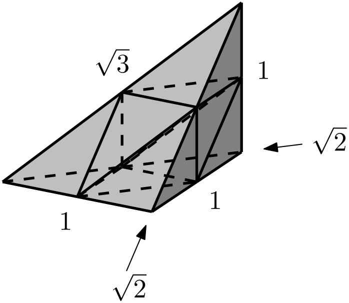
Tetrahedral meshes

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

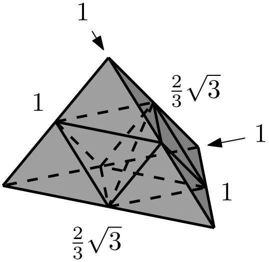
3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:

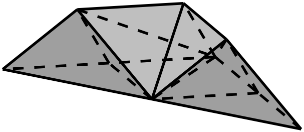


$\alpha = \arccos(-1/3)$:



(special cases of Hill tetrahedra)

non-Hill:



(Liu & Joe 1994)

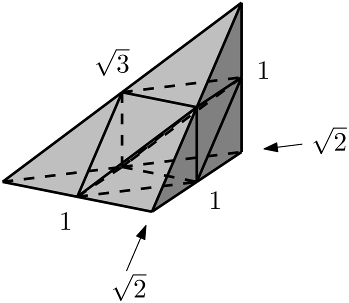
Tetrahedral meshes

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

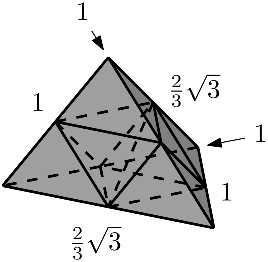
3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

$\alpha = \pi/2$:

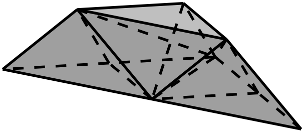


$\alpha = \arccos(-1/3)$:



(special cases of Hill tetrahedra)

non-Hill:



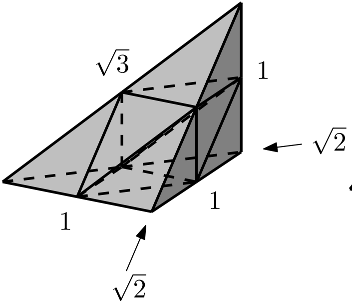
(Liu & Joe 1994)

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

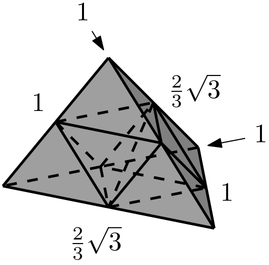
3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

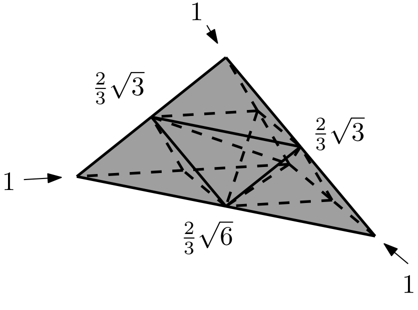
$\alpha = \pi/2$:



$\alpha = \arccos(-1/3)$:



non-Hill:



(special cases of Hill tetrahedra)

(Liu & Joe 1994)

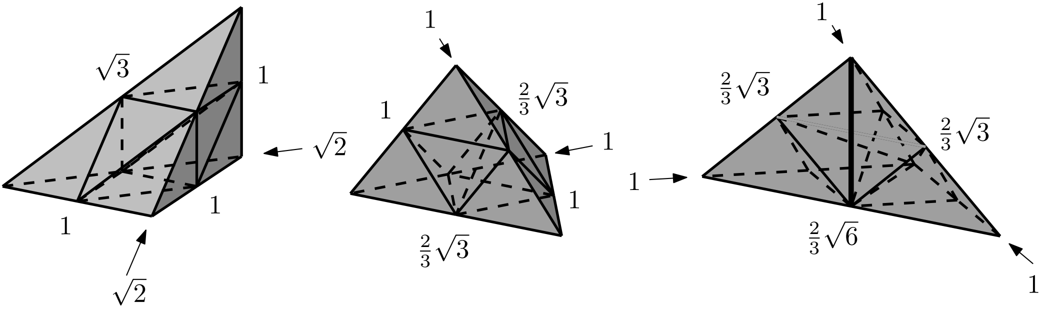
Tetrahedral meshes

2D: any $\triangle$ can be dissected into four similar (but smaller) triangles.

3D: If T is a k-reptile tetrahedron (= divisible into k congruent tetrahedra similar to T), then k is a cubic number (≥ 8). (Matousek & Savernova 2011)

Most known 8-reptile tetrahedra are *Hill tetrahedra* (after Hill 1895):
convex hull of 0, b_1, $b_1 + b_2$, $b_1 + b_2 + b_3$, where
b_1, b_2, b_3 are vectors of equal length, with equal angles $\alpha < \frac{2}{3}\pi$ between each pair

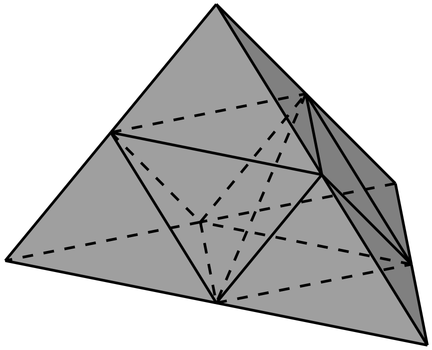
$\alpha = \pi/2$: $\alpha = \arccos(-1/3)$: non-Hill:



(special cases of Hill tetrahedra) (Liu & Joe 1994)

Tetrahedral meshes: Hill tetrahedra $\alpha \neq \pi/2$

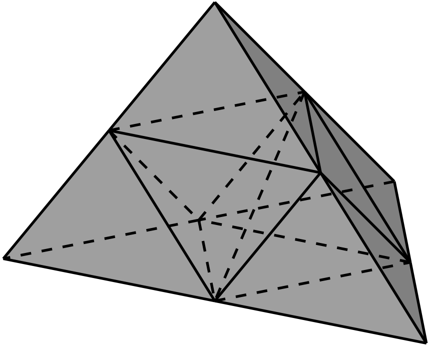
Only one way to subdivide into 8 equal, similar tetrahedra.



Tetrahedral meshes: Hill tetrahedra $\alpha \neq \pi/2$

Only one way to subdivide into 8 equal, similar tetrahedra.

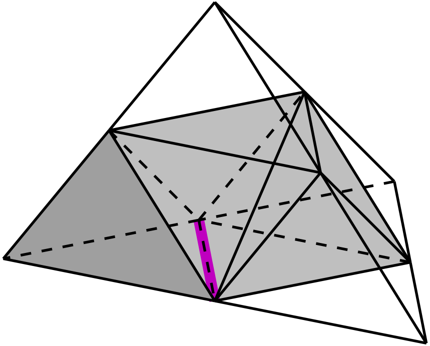
Four subtetrahedra share a face with only one other $\rightarrow$ no face-continuous traversal



Tetrahedral meshes: Hill tetrahedra $\alpha \neq \pi/2$

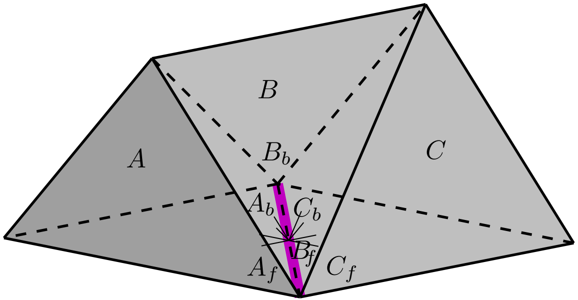
Only one way to subdivide into 8 equal, similar tetrahedra.

Four subtetrahedra share a face with only one other $\rightarrow$ no face-continuous traversal



Only one way to subdivide into 8 equal, similar tetrahedra.

Four subtetrahedra share a face with only one other $\rightarrow$ no face-continuous traversal



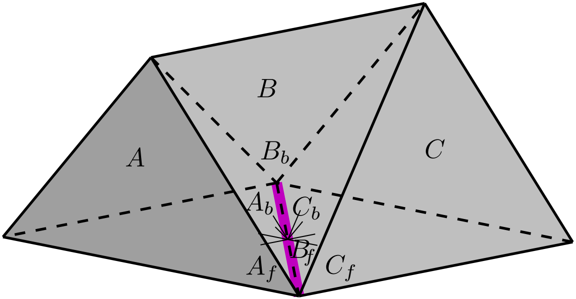
X_f, X_b: subtetrahedron of X in the front, back

Palindromic implies: $(A_f \prec A_b) = (B_b \prec B_f) = (C_f \prec C_b)$

Tetrahedral meshes: Hill tetrahedra $\alpha \neq \pi/2$

Only one way to subdivide into 8 equal, similar tetrahedra.

Four subtetrahedra share a face with only one other $\rightarrow$ no face-continuous traversal



X_f, X_b: subtetrahedron of X in the front, back

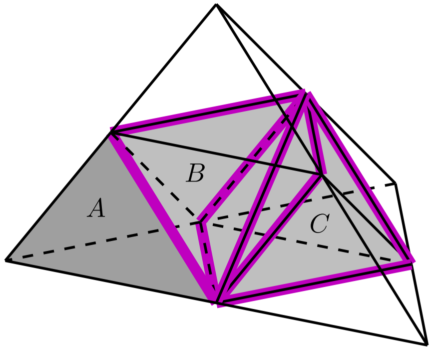
Palindromic implies: $(A_f \prec A_b) = (B_b \prec B_f) = (C_f \prec C_b)$

$\rightarrow$ between traversals of A and C, stack with vertices of common edge must be reversed

$\rightarrow$ B must be traversed between A and C

Only one way to subdivide into 8 equal, similar tetrahedra.

Four subtetrahedra share a face with only one other $\rightarrow$ no face-continuous traversal



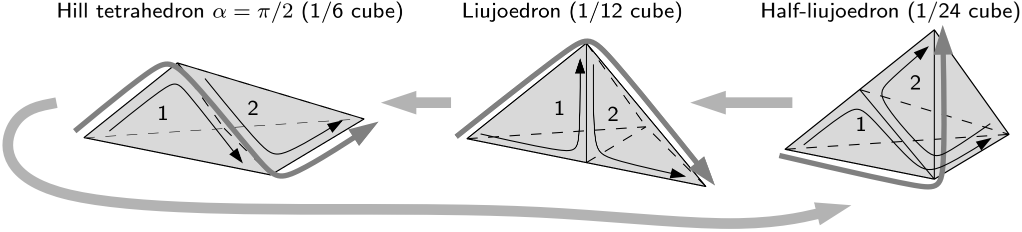
$\rightarrow$ between traversals of A and C, stack with vertices of common edge must be reversed

$\rightarrow$ B must be traversed between A and C

$\rightarrow$ combine conditions on all edges $\rightarrow$ $\lightning$ no palindromic traversal possible

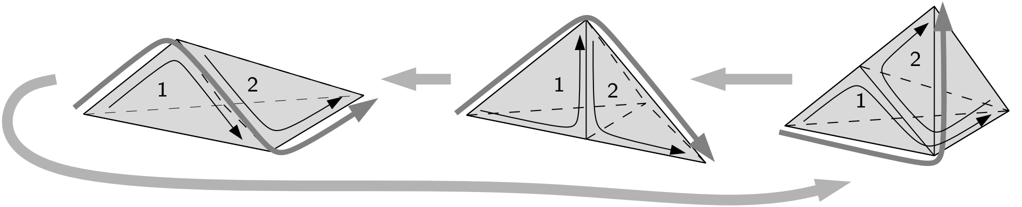
(still hope for: quasi-face-continuous, homodromic traversal)

Tetrahedral meshes: liujoedron bisection scheme

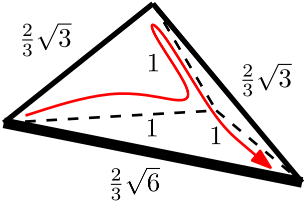


quasi-face-continuous, homodromic

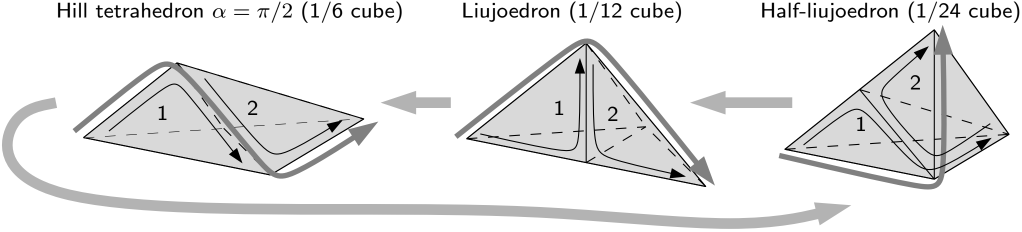
Hill tetrahedron $\alpha = \pi/2$ (1/6 cube) Liujoedron (1/12 cube) Half-liujoedron (1/24 cube)



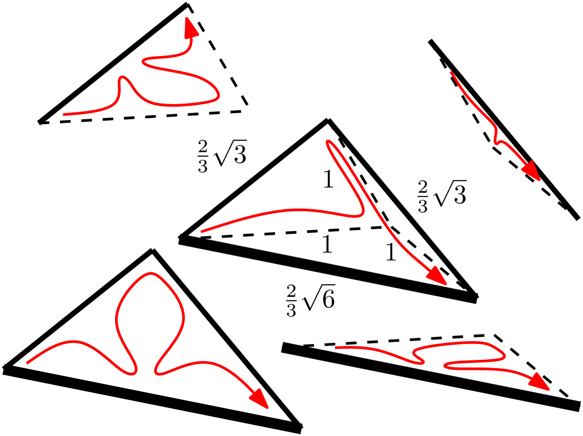
quasi-face-continuous, homodromic

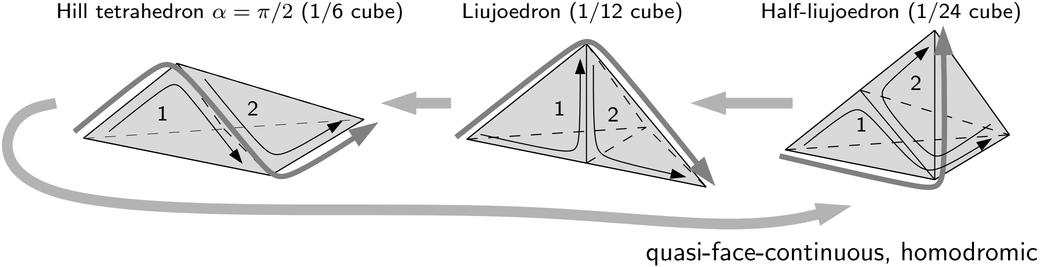


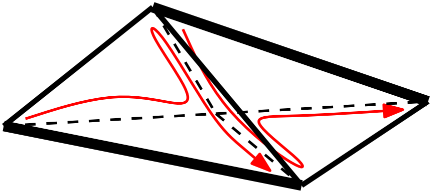
Tetrahedral meshes: liujoedron bisection scheme

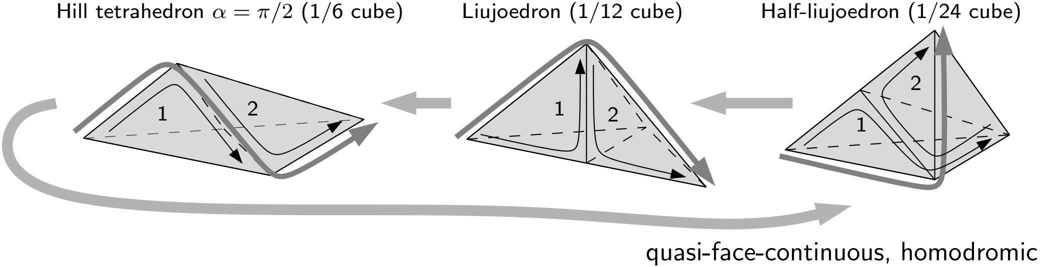


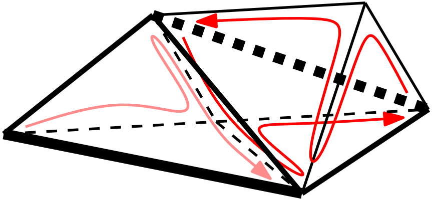
quasi-face-continuous, homodromic



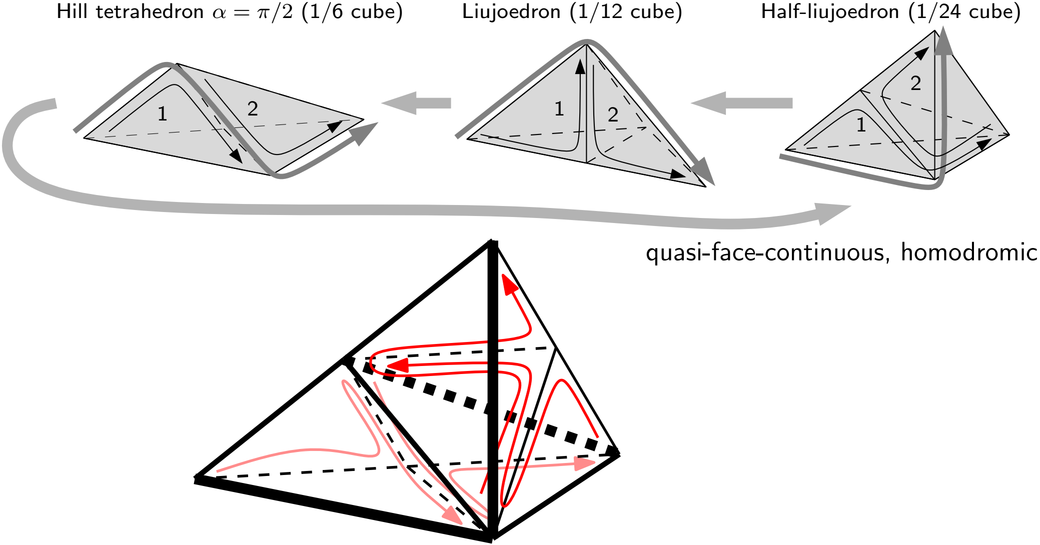

Hill tetrahedron $\alpha = \pi/2$ (1/6 cube)
Liujoedron (1/12 cube)
Half-liujoedron (1/24 cube)
1
2
1
2
1
2
quasi-face-continuous, homodromic



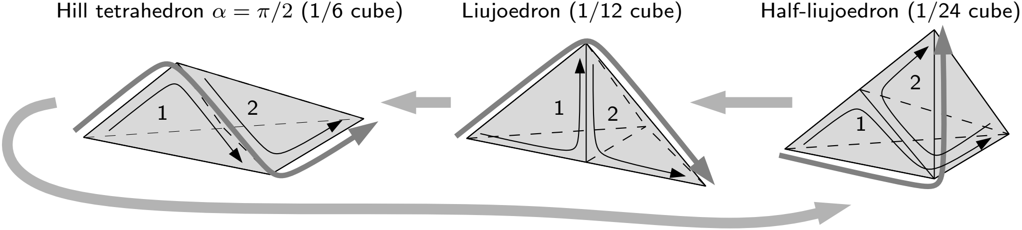

Hill tetrahedron $\alpha = \pi/2$ (1/6 cube)
Liujoedron (1/12 cube)
Half-liujoedron (1/24 cube)
1
2
1
2
1
2
quasi-face-continuous, homodromic



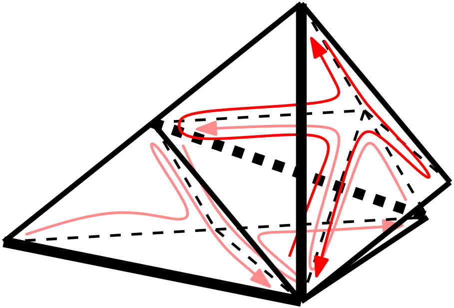
Tetrahedral meshes: liujoedron bisection scheme



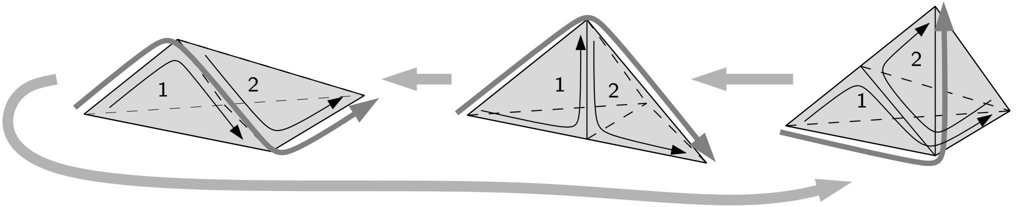
Tetrahedral meshes: liujoedron bisection scheme



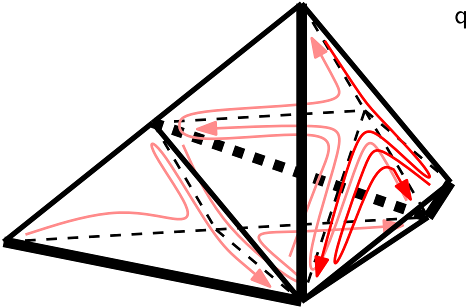
quasi-face-continuous, homodromic

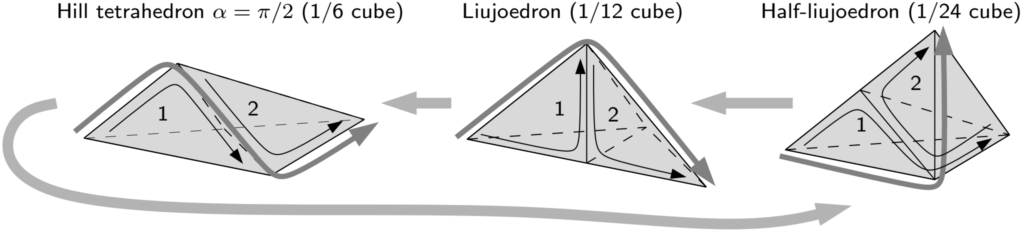


Hill tetrahedron $\alpha = \pi/2$ (1/6 cube) Liujoedron (1/12 cube) Half-liujoedron (1/24 cube)

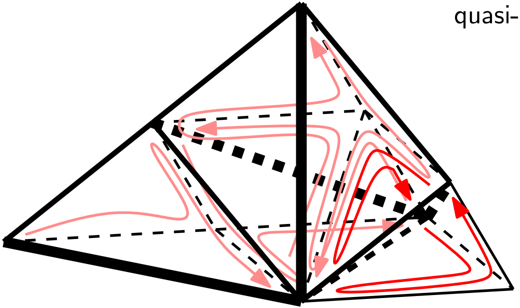


quasi-face-continuous, homodromic

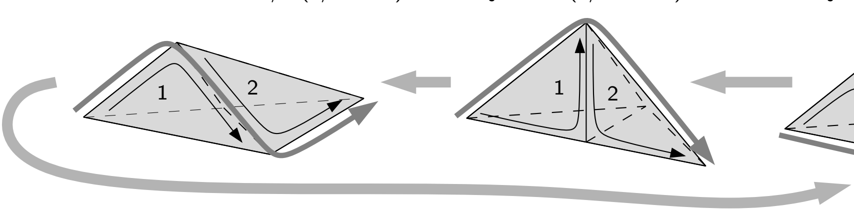




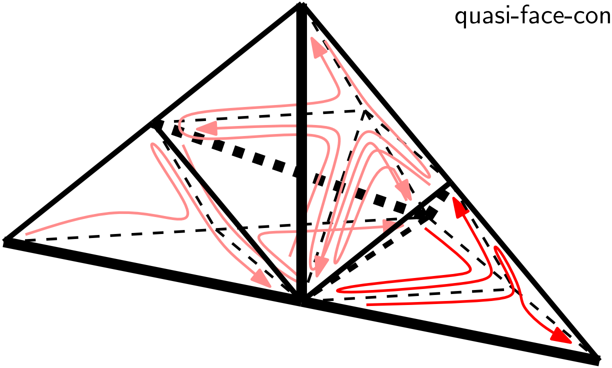
quasi-face-continuous, homodromic



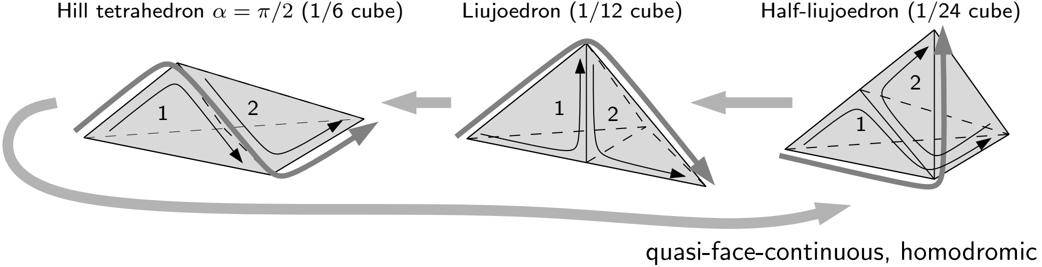
Tetrahedral meshes: liujoedron bisection scheme

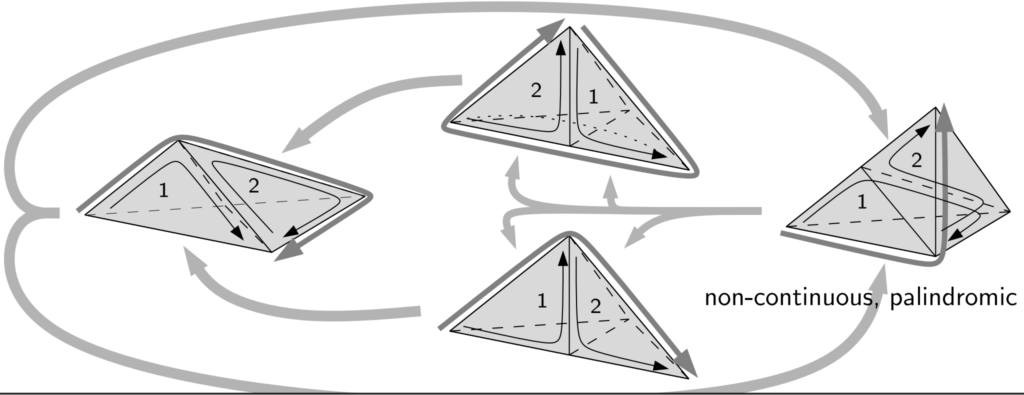


quasi-face-continuous, homodromic



Tetrahedral meshes: liujoedron bisection scheme





Summary: results on octree traversals

+	traversal known
− (light red)	impossible (almost proven)
− (red)	impossible

Cubes

	...-continuous non-	quasi-	face-
non-homodr.	+	+	+
homodromic	+	+	−
palindromic	+	?	−

Hill-tetrahedra $(\alpha \neq \pi/2)$

	...-continuous non-	quasi-	face-
n.-h.	+	?	−
h.dr.	?	?	−
p.dr.	−	−	−

Bisected tetrahedra

	...-continuous non-	quasi-	face-
n.-h.	+	+	−
h.dr.	+	+	−
p.dr.	+	− (almost proven)	−

Open problems:

- insightful proof of negative results on cubes
- meaningful surface-to-volume measures—and how to compute them?
- what tetrahedra are reptiles?
- (traversals for 1/6, 1/12, 1/24 cube tetrahedra that do not follow bisection scheme?)
- hypercubes in > 3 dimensions?
- simplexes in > 3 dimensions?
- accommodating adaptive cell shapes?

Summary: results on octree traversals

+	traversal known
– (light red)	impossible (almost proven)
– (red)	impossible

Cubes

	...-continuous non-	quasi-	face-
non-homodr.	+	+	+
homodromic	+	+	–
palindromic	+	?	–

Hill-tetrahedra ($\alpha \neq \pi/2$)

	...-continuous non-	quasi-	face-
n.-h.	+	?	–
h.dr.	?	?	–
p.dr.	–	–	–

Bisected tetrahedra

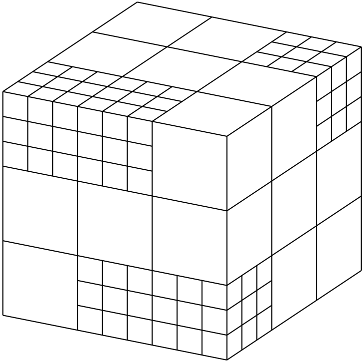
	...-continuous non-	quasi-	face-
non-homodr.	+	+	–
homodromic	+	+	–
palindromic	+	– (almost proven)	–

Open problems:

- insightful proof of negative results on cubes
- meaningful surface-to-volume measures—and how to compute them?
- what tetrahedra are reptiles?
- (traversals for 1/6, 1/12, 1/24 cube tetrahedra that do not follow bisection scheme?)
- hypercubes in > 3 dimensions?
- simplexes in > 3 dimensions?
- accommodating adaptive cell shapes?

THANK YOU FOR YOUR ATTENTION

Bonus slide: 3D Peano traversal

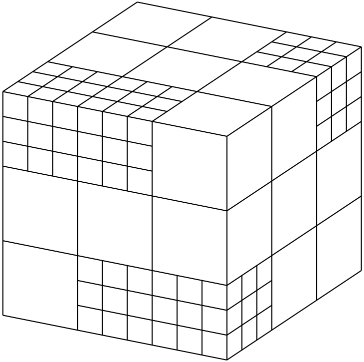


Solid object with heat sources and sinks on boundary;
simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

Bonus slide: 3D Peano traversal



Solid object with heat sources and sinks on boundary;
simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

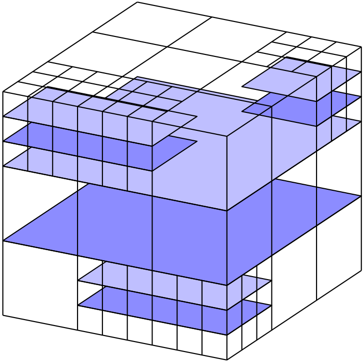
- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

WEINZIERL 2009:

current vertex/edge/face values popped from *input stack* when first visited;
new vertex/edge/face values pushed on *output stack* when last visited;
alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

Bonus slide: 3D Peano traversal



Solid object with heat sources and sinks on boundary; simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

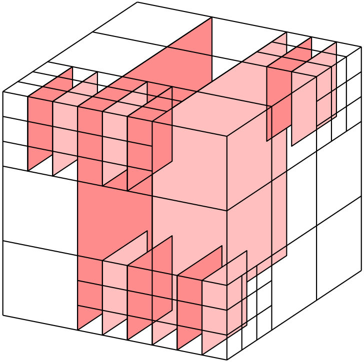
Weinzierl 2009:

current vertex/edge/face values popped from *input stack* when first visited;
new vertex/edge/face values pushed on *output stack* when last visited;
alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)

Bonus slide: 3D Peano traversal



Solid object with heat sources and sinks on boundary; simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

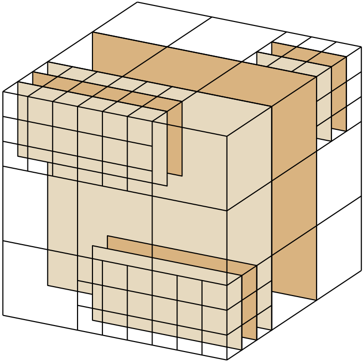
WEINZIERL 2009:

current vertex/edge/face values popped from *input stack* when first visited;
new vertex/edge/face values pushed on *output stack* when last visited;
alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)

Bonus slide: 3D Peano traversal



Solid object with heat sources and sinks on boundary; simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

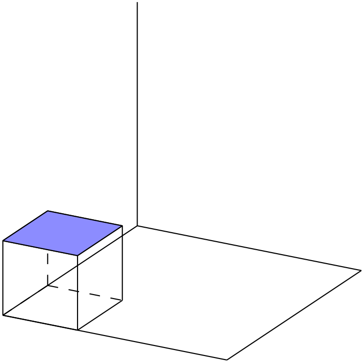
- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

WEINZIERL 2009:

current vertex/edge/face values popped from *input stack* when first visited;
new vertex/edge/face values pushed on *output stack* when last visited;
alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

WEINZIERL 2009:

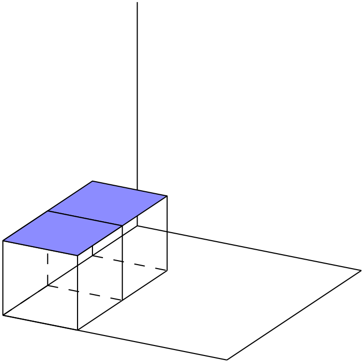
current vertex/edge/face values popped from *input stack* when first visited;

new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

Weinzierl 2009:

current vertex/edge/face values popped from *input stack* when first visited;

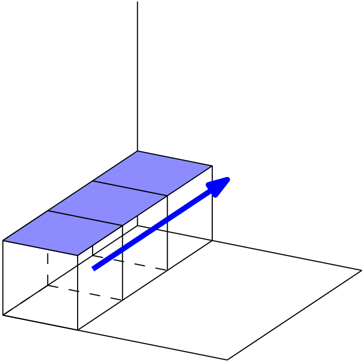
new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)

Bonus slide: 3D Peano traversal



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

Weinzierl 2009:

current vertex/edge/face values popped from *input stack* when first visited;

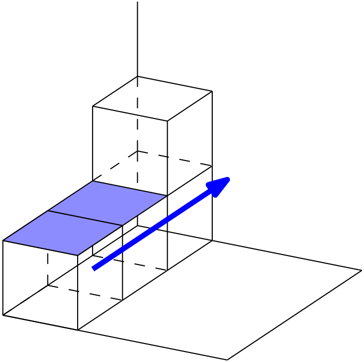
new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)

Bonus slide: 3D Peano traversal



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

Weinzierl 2009:

current vertex/edge/face values popped from *input stack* when first visited;

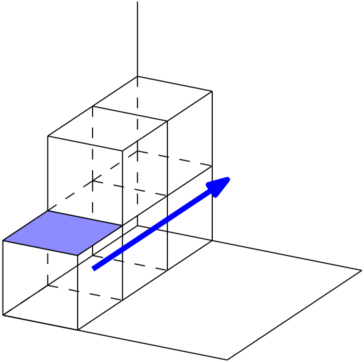
new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)

Bonus slide: 3D Peano traversal



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

WEINZIERL 2009:

current vertex/edge/face values popped from *input stack* when first visited;

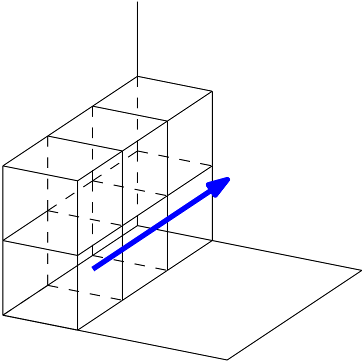
new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)

Bonus slide: 3D Peano traversal



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

WEINZIERL 2009:

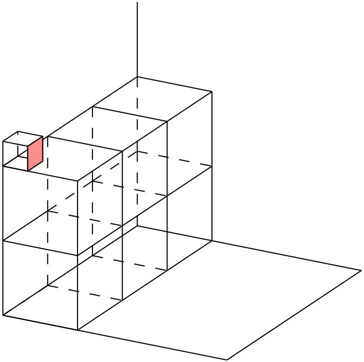
current vertex/edge/face values popped from *input stack* when first visited;

new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

WEINZIERL 2009:

current vertex/edge/face values popped from *input stack* when first visited;

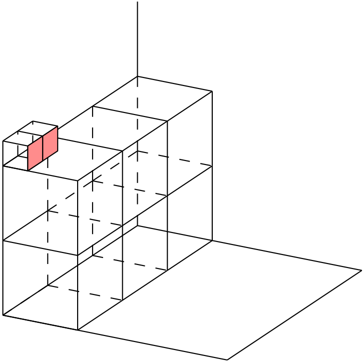
new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)

Bonus slide: 3D Peano traversal



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

WEINZIERL 2009:

current vertex/edge/face values popped from *input stack* when first visited;

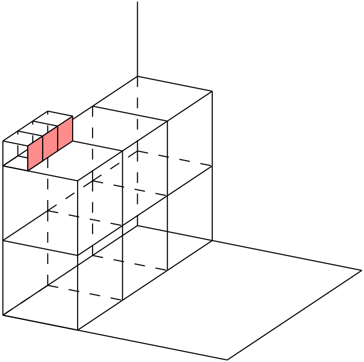
new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)

Bonus slide: 3D Peano traversal



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

Weinzierl 2009:

current vertex/edge/face values popped from *input stack* when first visited;

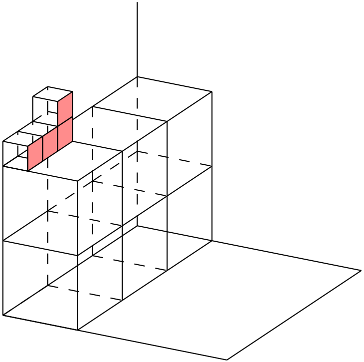
new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)

Bonus slide: 3D Peano traversal



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

WEINZIERL 2009:

current vertex/edge/face values popped from *input stack* when first visited;

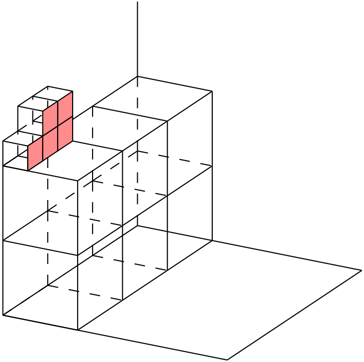
new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)

Bonus slide: 3D Peano traversal



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

WEINZIERL 2009:

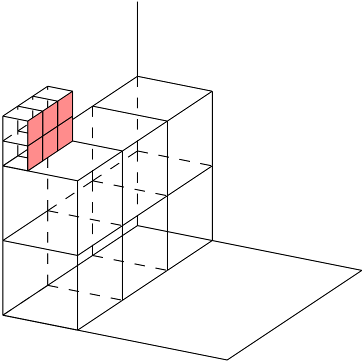
current vertex/edge/face values popped from *input stack* when first visited;

new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

WEINZIERL 2009:

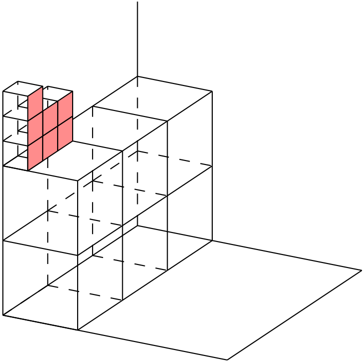
current vertex/edge/face values popped from *input stack* when first visited;

new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

WEINZIERL 2009:

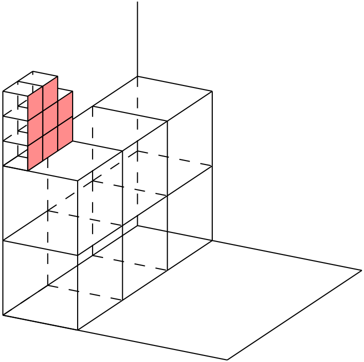
current vertex/edge/face values popped from *input stack* when first visited;

new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

WEINZIERL 2009:

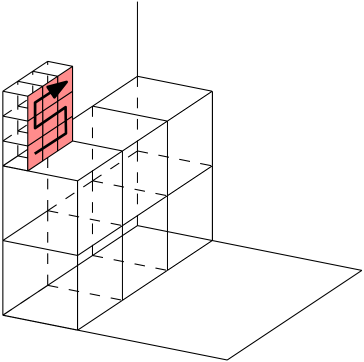
current vertex/edge/face values popped from *input stack* when first visited;

new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

WEINZIERL 2009:

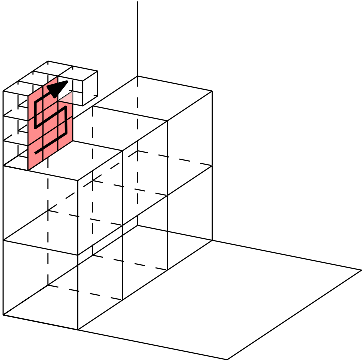
current vertex/edge/face values popped from *input stack* when first visited;

new vertex/edge/face values pushed on *output stack* when last visited;

alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)



Solid object with heat sources and sinks on boundary; simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

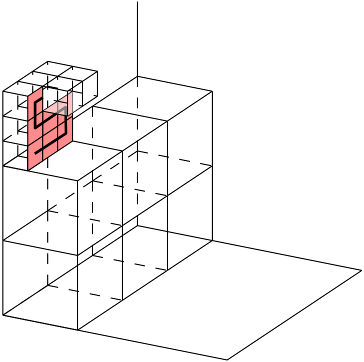
WEINZIERL 2009:

current vertex/edge/face values popped from *input stack* when first visited;
new vertex/edge/face values pushed on *output stack* when last visited;
alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)

Bonus slide: 3D Peano traversal



Solid object with heat sources and sinks on boundary;

simulate heat distribution on vertices, flow through faces by repeatedly iterating over all cells.

Requires:

- storing vertex/edge/face values between iterations
- repeated access to v/e/faces on boundaries between cells
- adaptive refinement of the mesh

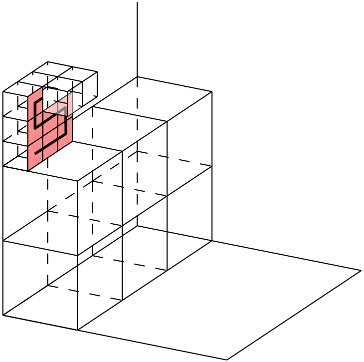
WEINZIERL 2009:

current vertex/edge/face values popped from *input stack* when first visited;
new vertex/edge/face values pushed on *output stack* when last visited;
alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)

Bonus slide: 3D Peano traversal



- cache-efficient!
- adaptive refinement = some pushes on stacks:
  no complicated vertex/edge/face index!
- easy to parallelize! each processor gets part of traversal
  (only diff: values for variables on boundary with other processor are read/written to different stacks)

WEINZIERL 2009:

current vertex/edge/face values popped from *input stack* when first visited;
new vertex/edge/face values pushed on *output stack* when last visited;
alternate forward and reverse iterations (stacks swap roles);

between first and last visits to a vertex/edge/face, value stored on intermediate stacks:

- two stacks for horizontal faces (one for odd heights, one for even heights)
- two stacks for left/right faces (one for odd coordinates, one for even coordinates)
- two stacks for front/back faces (one for odd coordinates, one for even coordinates)