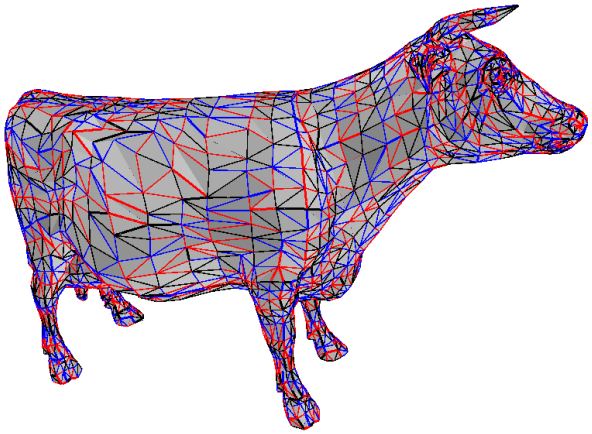
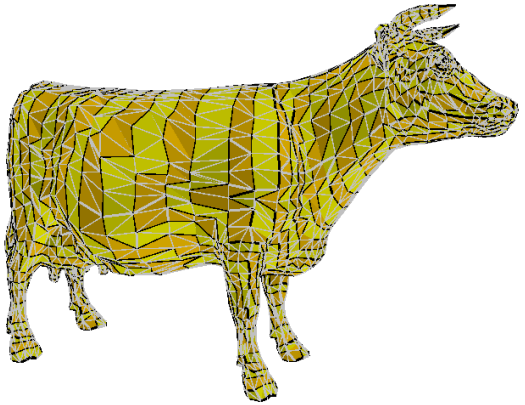
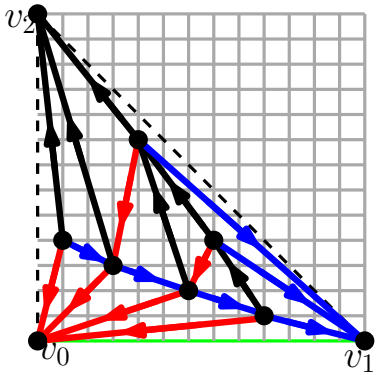
Adaptive analysis of compact representations for surface meshes: empirical evaluations and theoretical guarantees



sept. 9, GCCR2019
(Paris Diderot, Irif)

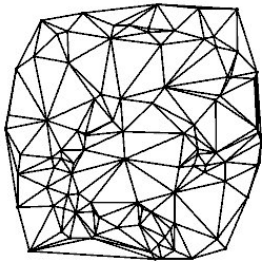




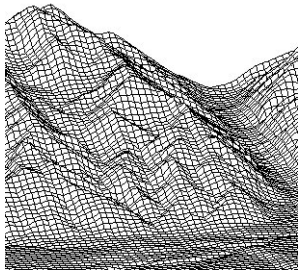
Luca Castelli Aleardi
(joint work with Olivier Devillers, Loria)

Planar and surface meshes

(from computational geometry to computer graphics, geometric processing, ...)

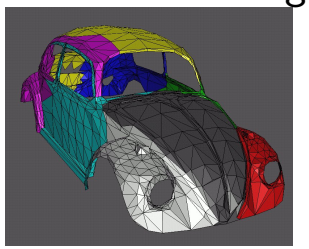


Delaunay triangulation

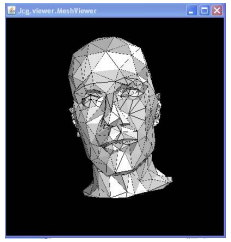


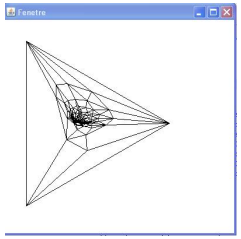
GIS Technology

geometric modeling



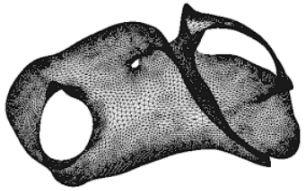
mesh parameterization





3D reconstruction

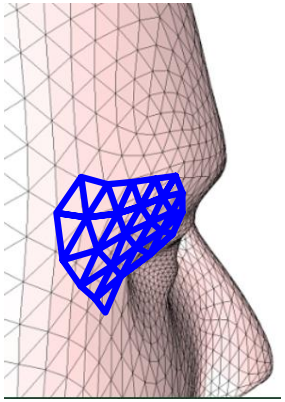
David statue (Stanford's Digital Michelangelo Project, 2000)

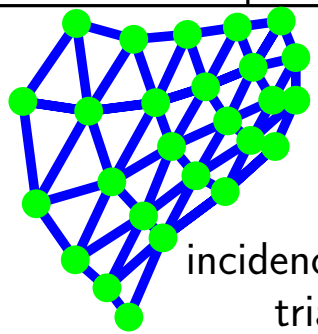


force-directed layout of the `4elt` graph

What is a surface mesh?

informally: a set of vertices, edges and faces (polygons) defining a polyhedral surface embedded in 3D (discrete approximation of a shape)





incidence relations between triangles, vertices

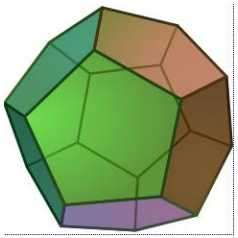
for us this is equivalent to

combinatorial map: a graph + a combinatorial embedding (on a surface)

$+$

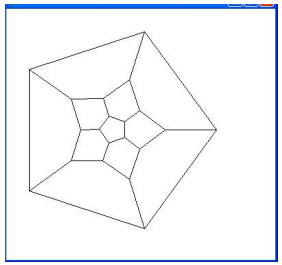
geometric realization in 2D or 3D

genus 0 polyhedral mesh



$\approx$

planar map



Planar and surface meshes

(from computational geometry to computer graphics, geometric processing, ...)

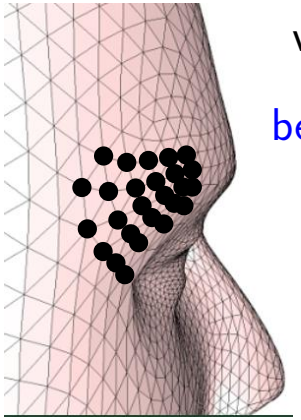
David statue (Stanford's Digital Michelangelo Project, 2000)

2 billions polygons

32 Giga bytes
(without compression)

Geometric v.s combinatorial information

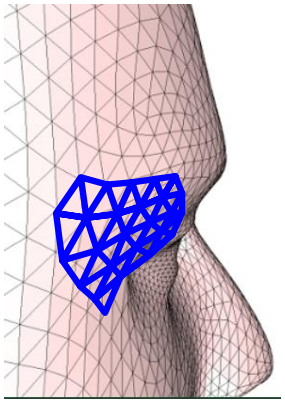
Geometry

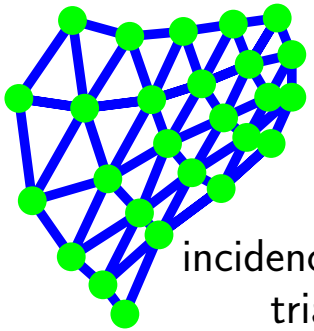


vertex coordinates

between 30 et 96 bits/vertex

Connectivity





incidence relations between triangles, vertices

between 13 and 19 references/vertex
($13 \log n$ and $19 \log n$ bits/vertex)

in practice between 416 and 608 bits/vertex

combinatorial map: a graph + a combinatorial embedding (on a surface)

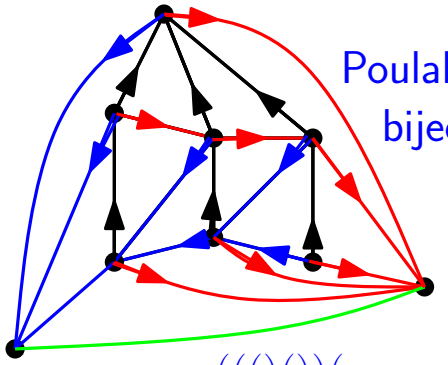
we do not mind about the geometric realization in 2D or 3D

Mesh encoding (worst case analysis)

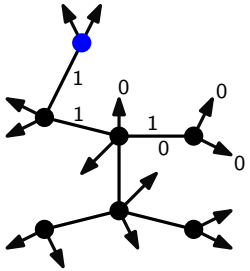
Mesh compression (no support of fast queries)

number of planar triangulations of size n

$$\Psi_n = \frac{2(4n+1)!}{(3n+2)!(n+1)!} \approx \frac{16}{27}\sqrt{\frac{3}{2\pi}}\, n^{-5/2} \left(\frac{256}{27}\right)^n$$



Poulalhon, Schaeffer bijection (2003)



110100010 . . .

$2n$ symbols ((()())(. . .

$2(n-2)$ symbols 1111010101000

information theory bound $\frac{1}{n}\log_2 \Psi_n \approx \log_2\left(\frac{256}{27}\right) \approx$ **3.2451** bits/vertex

asymptotic optimal bound

(standard) data structures

triangle based DS (CGAL)

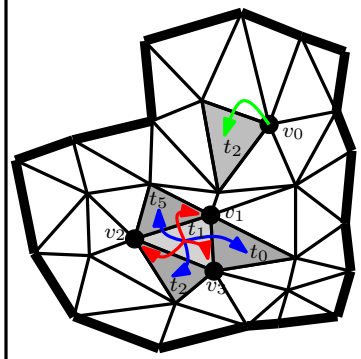
```
class Triangle{
  Triangle t1, t2, t
  Vertex v1, v2, v3;
}
 class Vertex{
  Triangle  root;
  Point p;
}
```

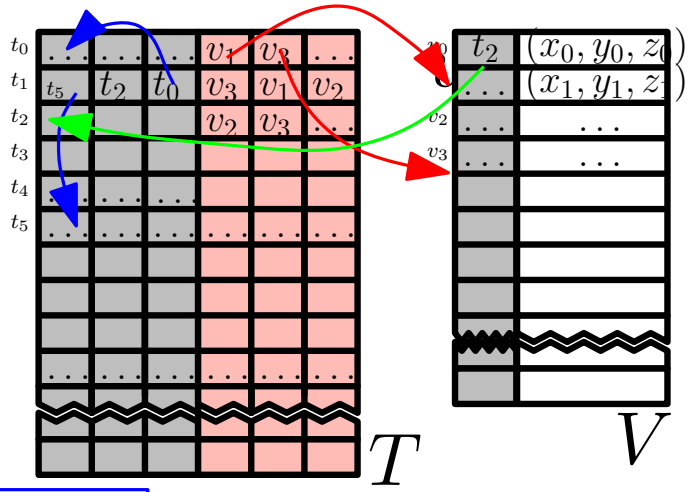
for each triangle, store:

- 3 references to neighboring faces
- 3 references to incident vertices

for each vertex, store:

- 1 reference to an incident face





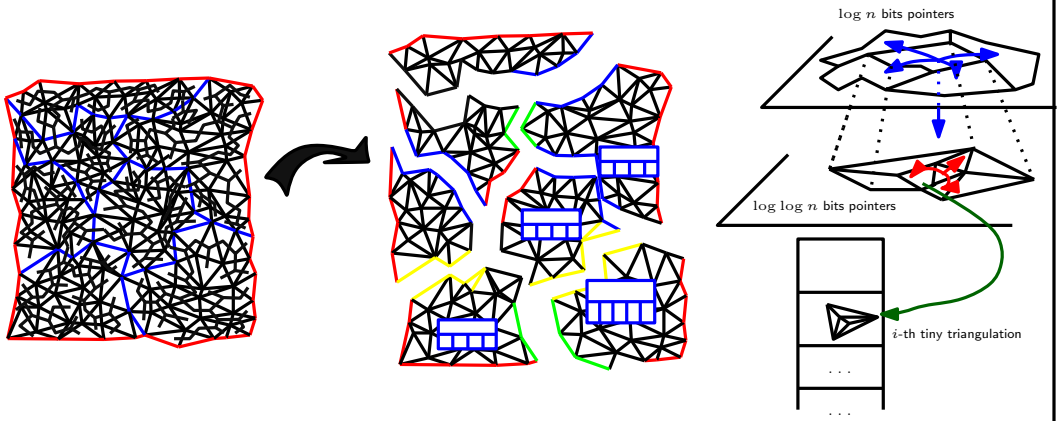
$(3+3) \times f + n$

$6 \times 2n + n =$ **$13n$** references

Succinct representations (supporting queries in $O(1)$ time)

(Castelli Aleardi, Devillers, Schaeffer 2005, 2006)

in the Word-Ram model



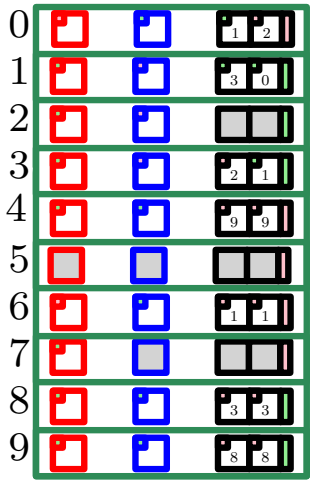
$$3.2451n + O\left(\frac{\log\log n}{\log n}\right) = 3.2451n + o(n) \text{ bits/vertex}$$

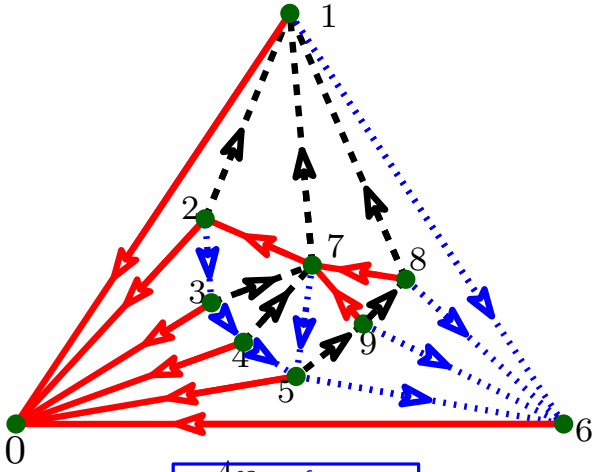
asymptotic optimal bound

Practical compact data structures (fast implementations)

(Castelli Aleardi, Devillers 2011)

1.2 - 1.9 times slower than standard data structures





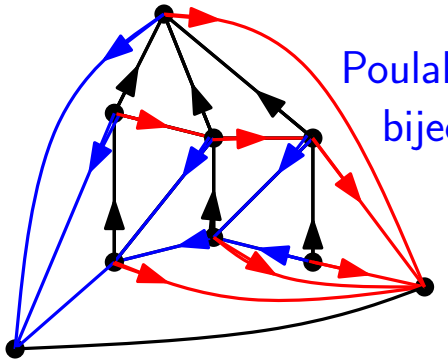
$4n$ references

Mesh encoding (worst case analysis)

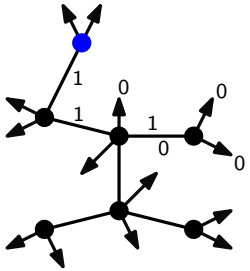
Mesh compression (no support of fast queries)

number of planar triangulations of size n

$$\Psi_n = \frac{2(4n+1)!}{(3n+2)!(n+1)!} \approx \frac{16}{27}\sqrt{\frac{3}{2\pi}}\, n^{-5/2} \left(\frac{256}{27}\right)^n$$



Poulalhon, Schaeffer bijection (2003)



110100010 . . .

information theory bound $\frac{1}{n}\log_2 \Psi_n \approx \log_2\left(\frac{256}{27}\right) \approx$ 3.2451 bits/vertex

asymptotic optimal bound

(standard) data structures

triangle based DS (CGAL)

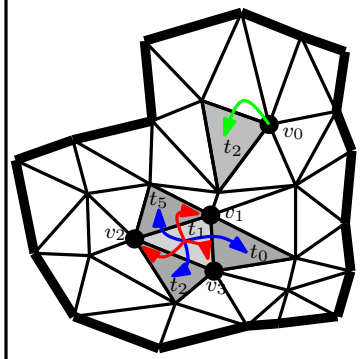
```
class Triangle{
  Triangle t1, t2, t
  Vertex v1, v2, v3;
}
 class Vertex{
  Triangle  root;
  Point p;
}
```

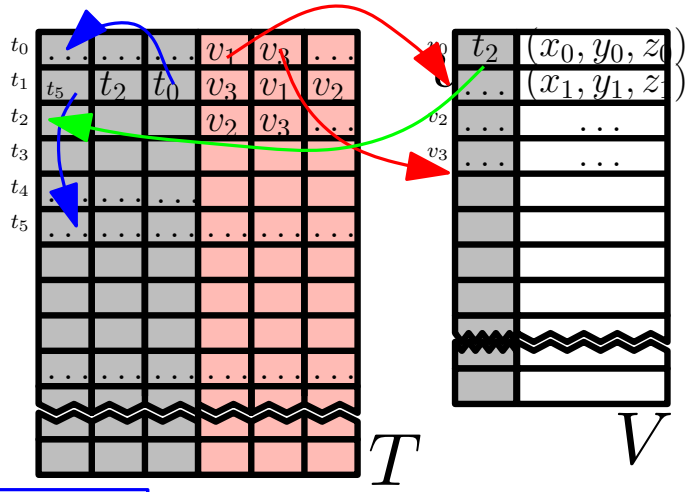
for each triangle, store:

- 3 references to neighboring faces
- 3 references to incident vertices

for each vertex, store:

- 1 reference to an incident face



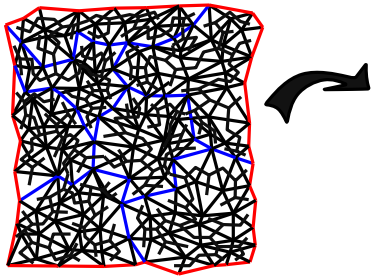


$(3+3) \times f + n$

$6 \times 2n + n =$ $13n$ references

Succinct representatio

(Castelli Aleardi, Devillers, Schaeffer 2



$3.2451n + O\left(\frac{\log\log n}{\log n}\right)$

Bit-efficient compact data structures (slower implementations)

Ferres, Fuentes-Sepulveda, Gagied, He, Navarro, 2017

size: ≈ 17.70 bits/vertex

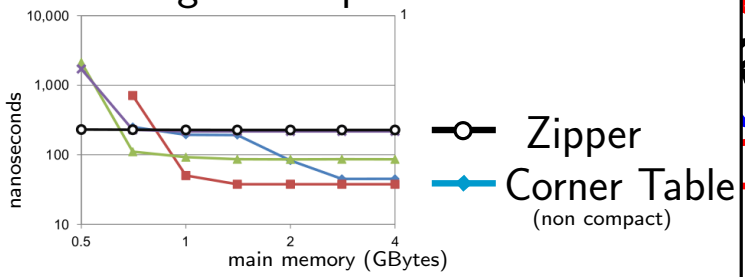
degree computation: ≈ 200 times slower (compared to standard non compact DS)

Zipper (Gurung, Luffel, Lindstrom, Rossignac '13)

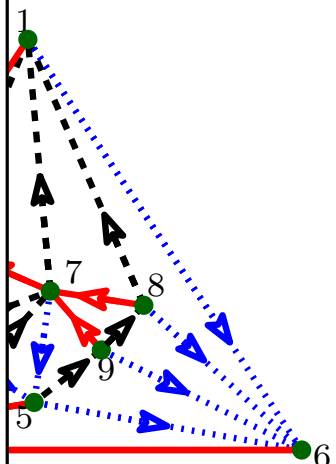
size: ≈ 12 bits/vertex

several times times slower (compared to non compact DS)

vertex degree computation



t implementations)



$4n$ references

Mesh encoding: real-world vs. pathological and random graphs

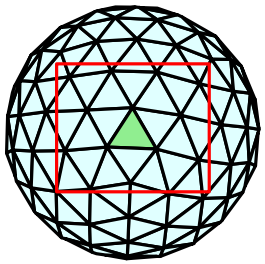
Mesh compression (no support of fast queries)

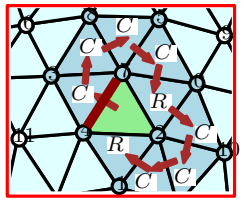
number of planar triangulations of size n

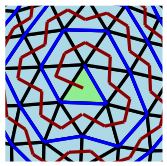
$$\Psi_n = \frac{2(4n+1)!}{(3n+2)!(n+1)!} \approx \frac{16}{27}\sqrt{\frac{3}{2\pi}}\, n^{-5/2}\left(\frac{256}{27}\right)^n \qquad \frac{1}{n}\log_2 \Psi_n \approx \log_2\left(\frac{256}{27}\right) \boxed{\approx 3.2451 \text{ bits/vertex}}$$

asymptotic optimal bound

Edgebreaker (Rossignac '99)







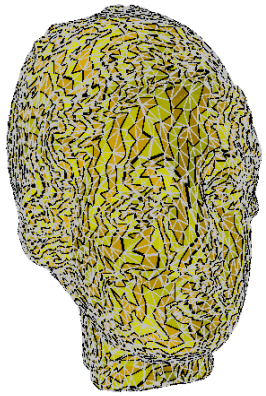
$$CCCCRCCCCRCCR\ldots$$

encoding word of length $f - 1 = 2n - 5$
(on the alphabet $\{C, L, E, R, S\}$)

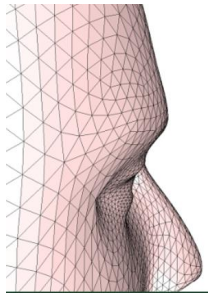
worst-case bound: $3.67n$ bits

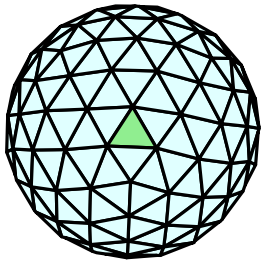
practical results (regular meshes): $\approx 2n$ bits

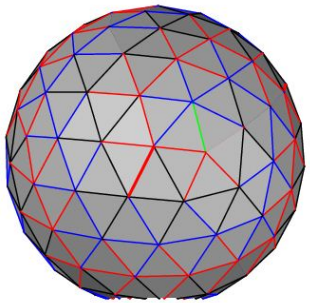
Irregular meshes

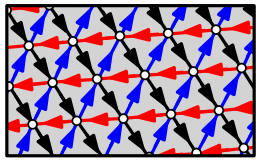


Regular meshes

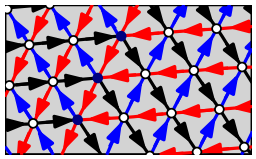




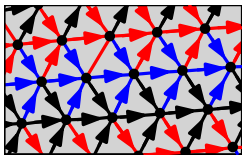




perfectly balanced



well balanced

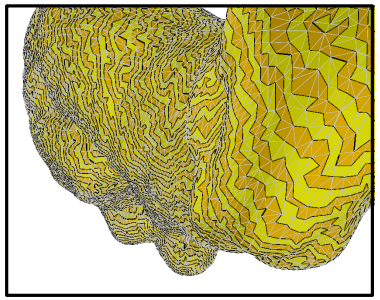


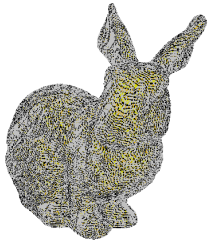
strongly unbalanced

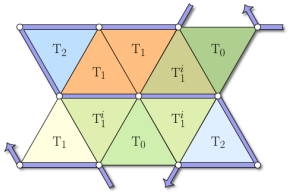
Practical compact data structures (fast implementations)

sometimes faster than standard data structures

Laced Ring data structure
(Gurung, Luffel, Lindstrom, Rossignac '11)







worst-case bound: ?

practical results (regular meshes): $\approx 2.2n$ references

Some facts about planar graphs

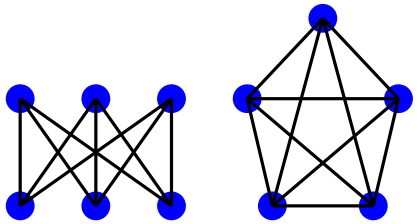
(genus 0 meshes)

("As I have known them")

Major results on planar graphs

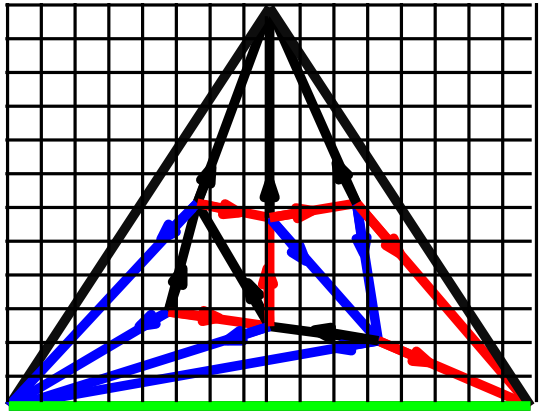
Kuratowski theorem (1930) (cfr Wagner's theorem, 1937)

- G contains neither K_5 nor $K_{3,3}$ as minors



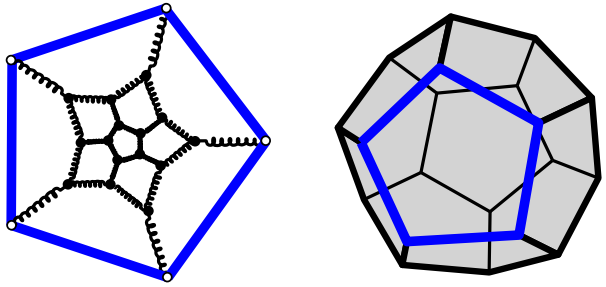
Schnyder woods ('89)

- planarity criterion via dimension of partial orders: $dim(G) \leq 3$
- linear-time grid drawing, with $O(n) \times O(n)$ resolution



Thm (Tutte barycentric method, 1963)

Every 3-connected planar graph G admits a convex representation in R^2.



Thm (Colin de Verdière, 1990) Colin de Verdiere invariant (multiplicity of λ_2 eigenvalue of a generalized laplacian)

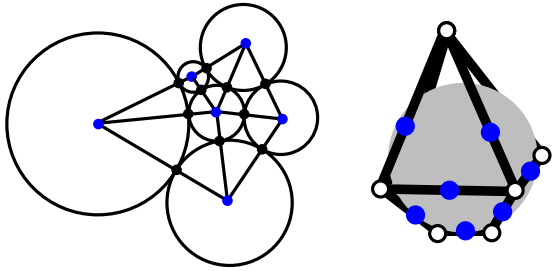
- $\mu(G) \leq 3$

$$\begin{bmatrix} 4 & -1 & \dots & \dots & 0 \\ -1 & 5 & \dots & & \\ \dots & & \dots & & \\ \dots & & & & \dots \\ 0 & \dots & & & 3 \end{bmatrix}$$

$$L_G[i,k] = \begin{cases} deg(v_i) \\ -A_G[i,j] \end{cases}$$

Thm (Koebe-Andreev-Thurston)

Every planar graph with n vertices is isomorphic to the intersection graph of n disks in the plane.

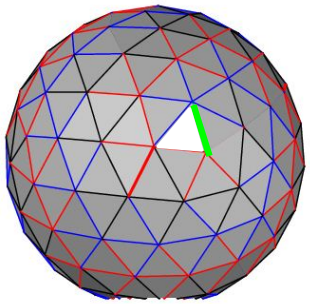


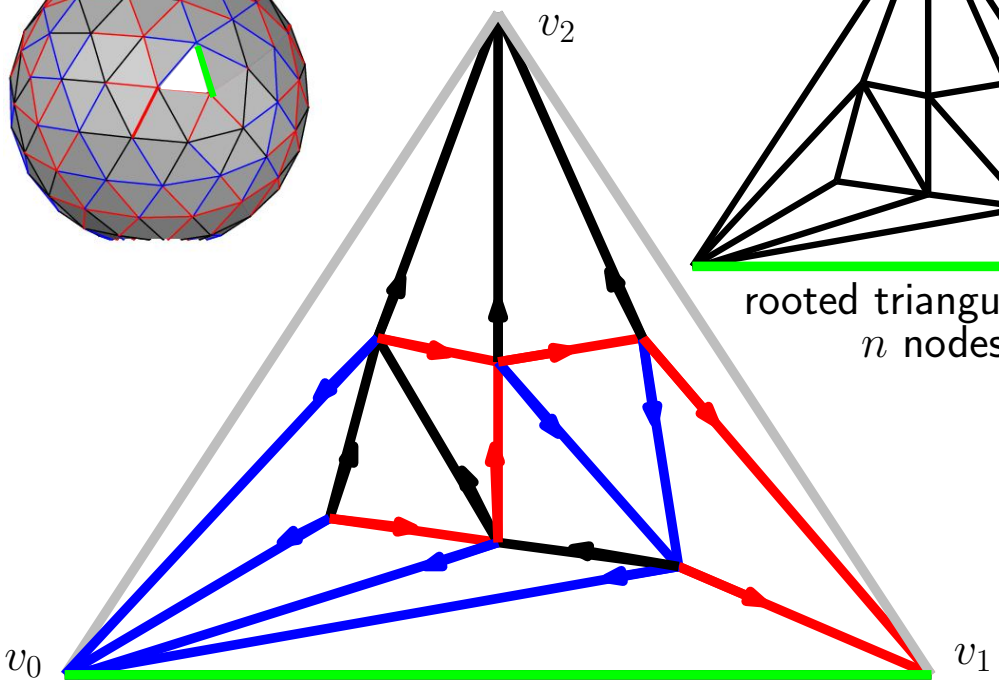
Schnyder woods

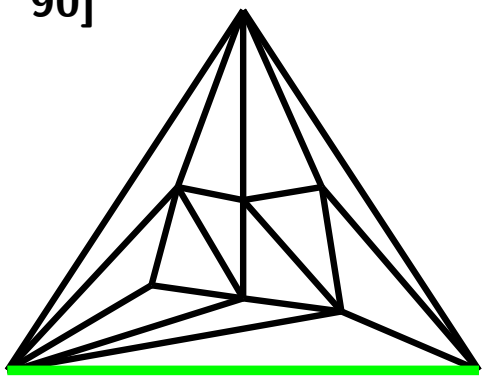
(overview)

Schnyder woods: (planar) definition

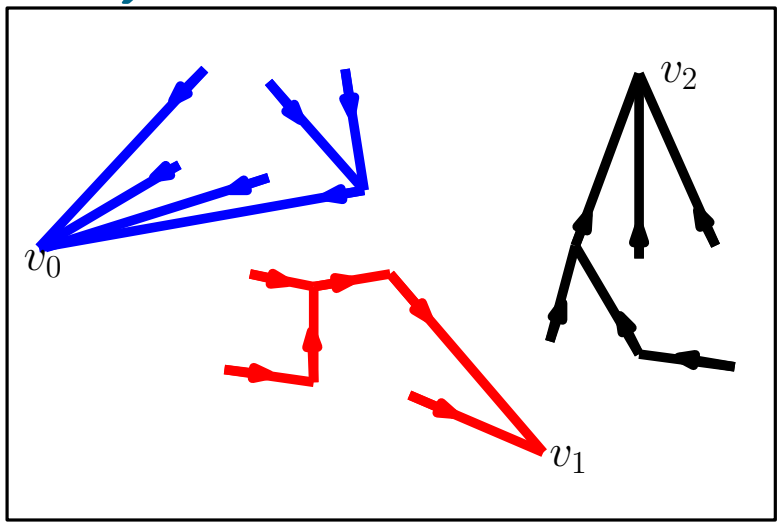
[Schnyder '89, '90]



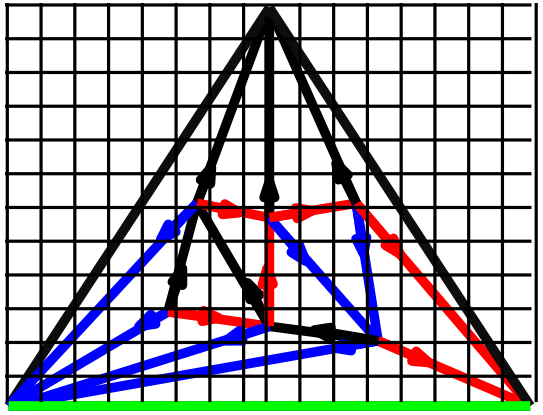




rooted triangulation on n nodes



T_0, T_1 and T_2 are vertex spanning trees



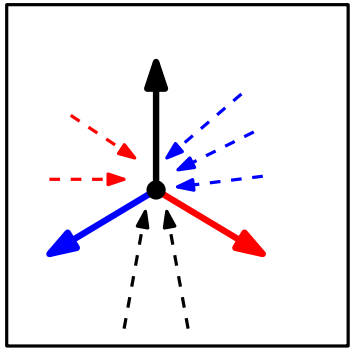
linear-time algorithm for grid drawing, with resolution $O(n) \times O(n)$

Def

A Schnyder wood of a (rooted) planar triangulation is partition of all inner edges into three sets T_0, T_1 and T_2, s.t.

i) edge are colored and oriented in such a way that each inner nodes has exaclty one outgoing edge of each color

ii) colors and orientations around each inner node must respect the local Schnyder condition

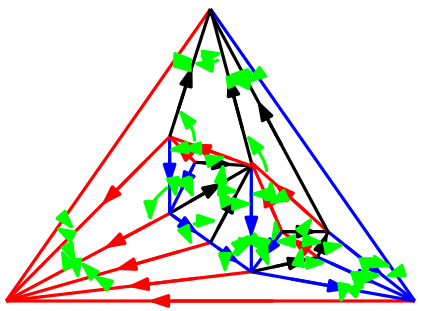


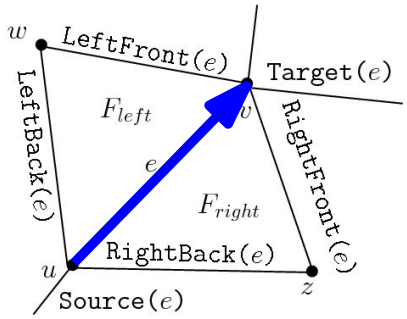
Compact (practical) mesh data structures

(non compact) data structures

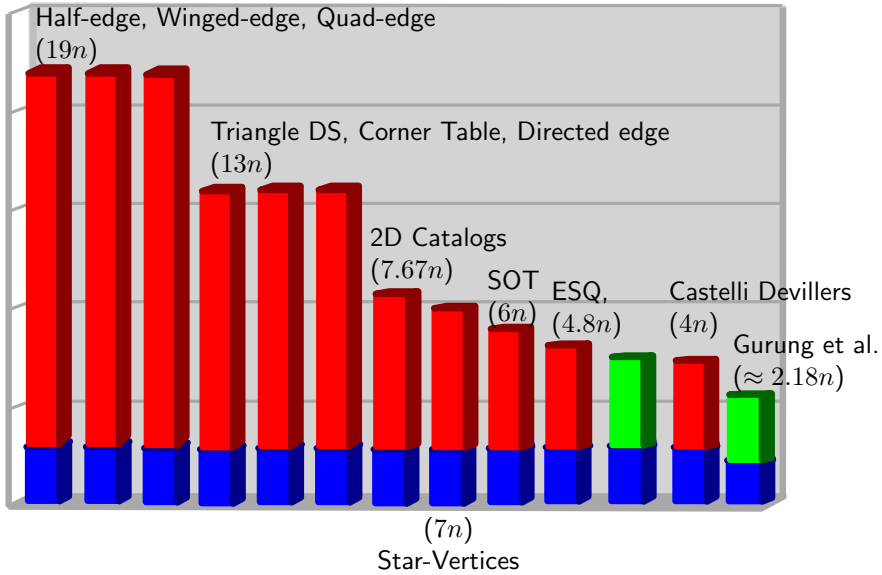
compact data structures

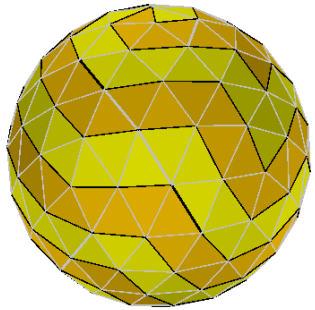
Data Structure	size	navigation time	vertex access	dynamic
Half-edge/Winged-edge/Quad-edge	$18n + n$	$O(1)$	$O(1)$	*yes*
Triangle based DS / Corner Table	$12n + n$	$O(1)$	$O(1)$	*yes*
Directed edge (Campagna et al. '99)	$12n + n$	$O(1)$	$O(1)$	*yes*
2D Catalogs (Castelli Aleardi et al., '06)	$7.67n$	$O(1)$	$O(1)$	*yes*
Star vertices (Kallmann et al. '02)	$7n$	$O(d)$	$O(1)$	*no*
TRIPOD (Snoeyink, Speckmann, '99)	$6n$	$O(1)$	$O(d)$	*no*
SOT (Gurung et al. 2010)	$6n$	$O(1)$	$O(d)$	*no*
SQUAD (Gurung et al. 2011)	$(4+\varepsilon)n$	$O(1)$	$O(d)$	*no*
ε between 0.09 and 0.3				
ESQ (Castelli Aleardi, Devillers, Rossignac'12)	$4.8n$	$O(1)$	$O(d)$	*yes*
Castelli Aleardi and Devillers (Isaac '11, JoCG'18)	$4n$ (or $5n$)	$O(1)$	$O(d)$ (or $O(1)$)	*no*
LR (Gurung et al. 2011)	$(2+\delta)n$	$O(1)$	$O(1)$	*no*
δbetween 0.2 and 0.3				





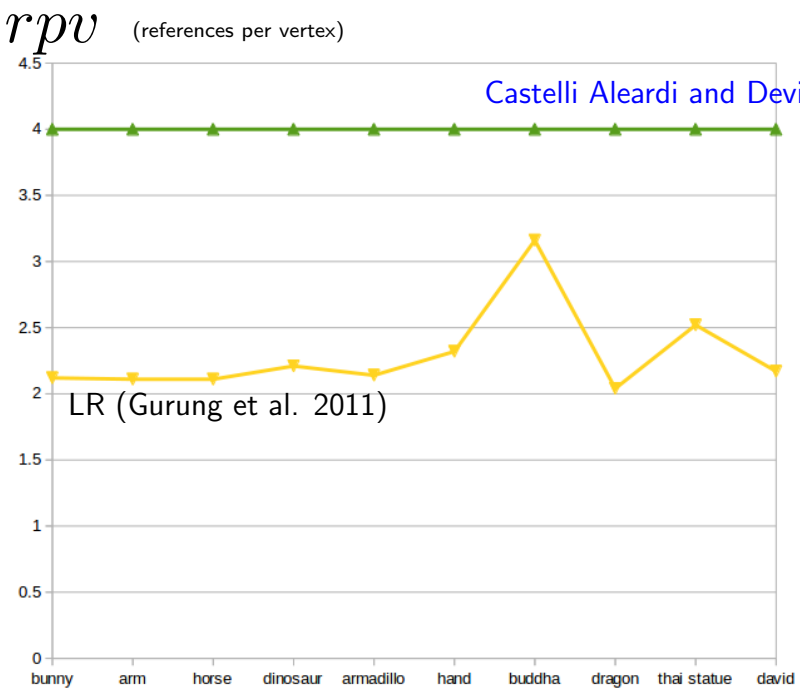
Winged-edge



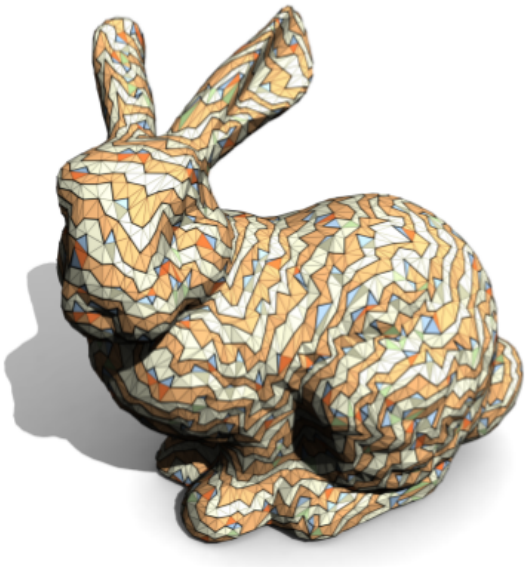


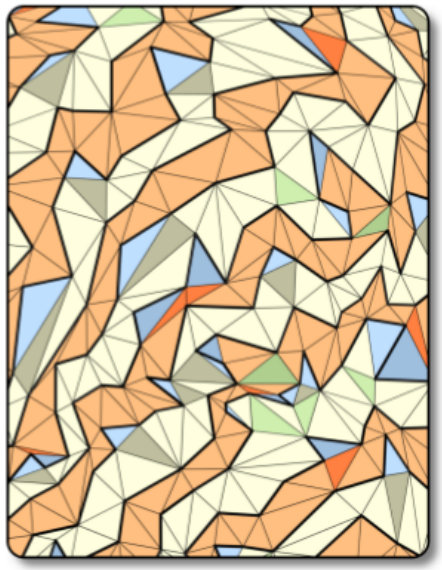
LR: Laced Ring data structure

LR (Gurung et al. 2011)



space performances in practice (from oriignal paper)

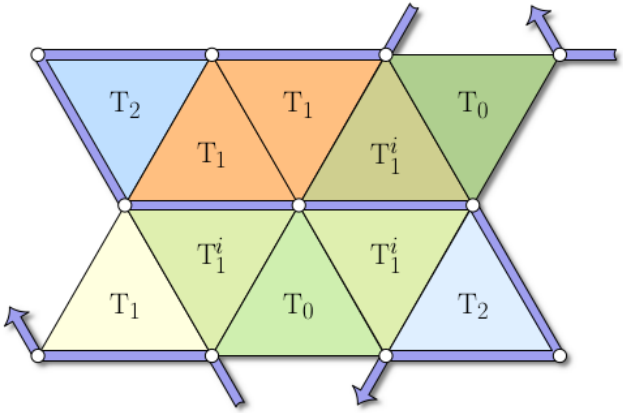




Ring-expander

```
c = s;                          // start at the seed corner s
c.n.v.m = c.p.v.m = true;       // mark vertices as visited
do {
    if (!c.v.m) c.v.m = c.t.m = true;      // invade c.t
    else if (!c.t.m) c = c.o;    // go back one triangle
    c = c.r;   // advance to next ring edge on the right
} while (c != s.o);          // until back at the beginning
```

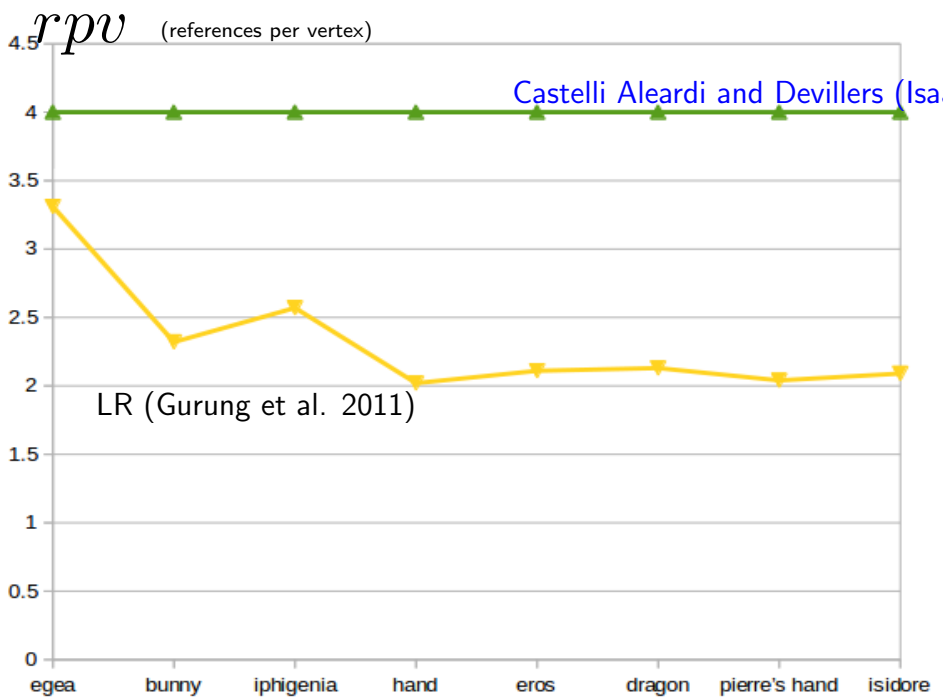
triangle classification



$$size(LR) := \boxed{2v_r} + v_i + 6|T_0| + 3|T_1^i + T_2^i|$$

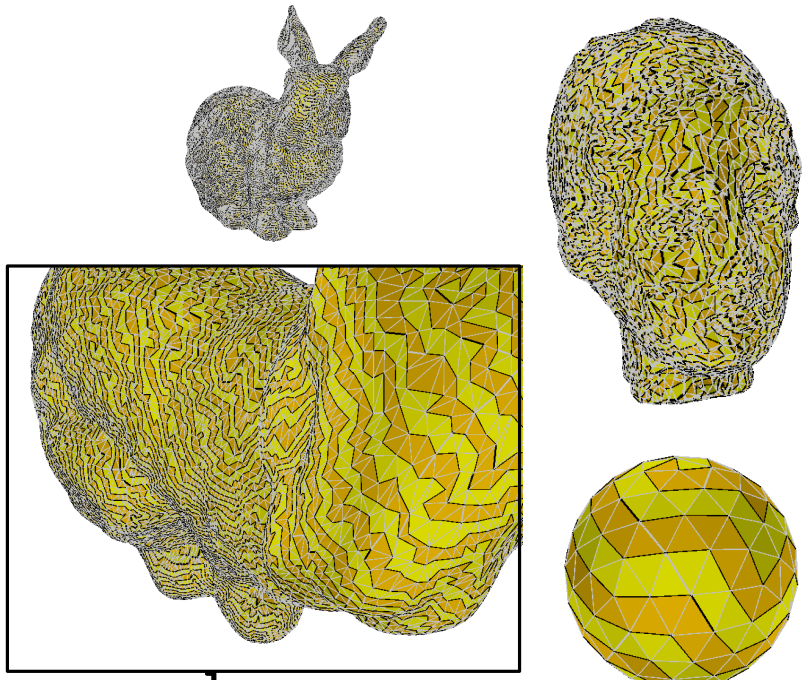
isolated vertices

LR: Laced Ring data structure



testing 3d meshes from aim@shape (with our implementation)

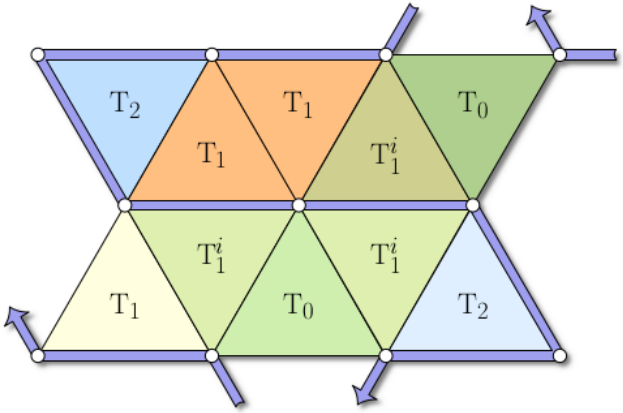
LR (Gurung et al. 2011)



Ring-expander

```
c = s;                          // start at the seed corner s
c.n.v.m = c.p.v.m = true;       // mark vertices as visited
do {
    if (!c.v.m) c.v.m = c.t.m = true;      // invade c.t
    else if (!c.t.m) c = c.o;    // go back one triangle
    c = c.r;   // advance to next ring edge on the right
} while (c != s.o);          // until back at the beginning
```

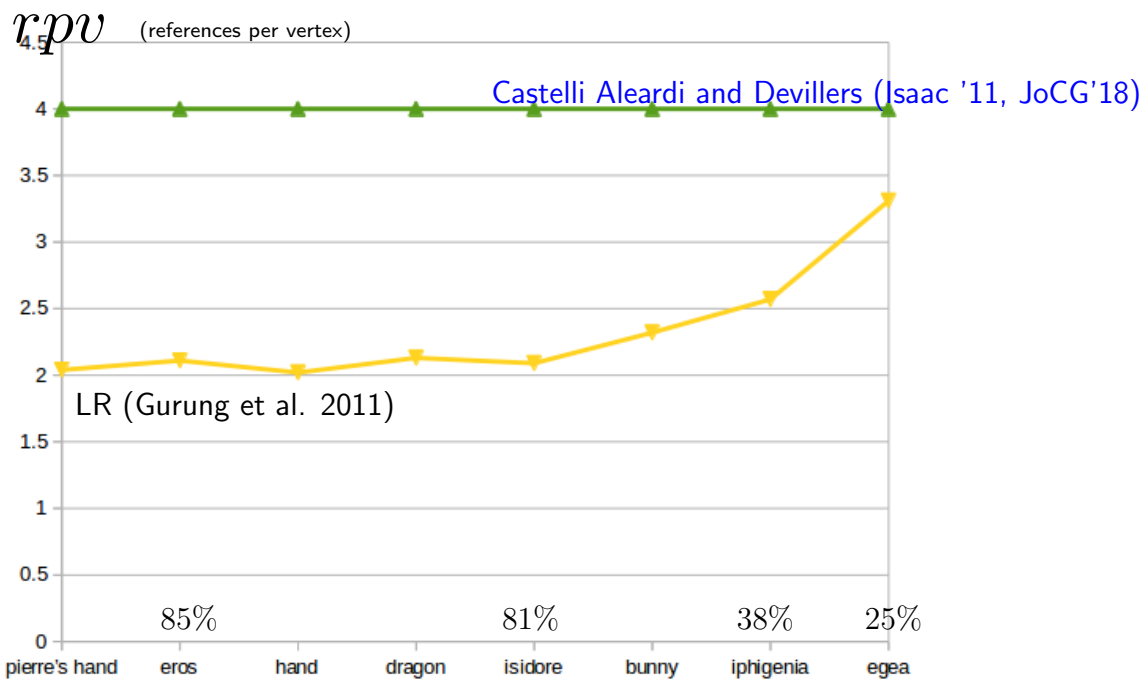
triangle classification



$$size(LR) := \boxed{2v_r} + v_i + 6|T_0| + 3|T_1^i + T_2^i|$$

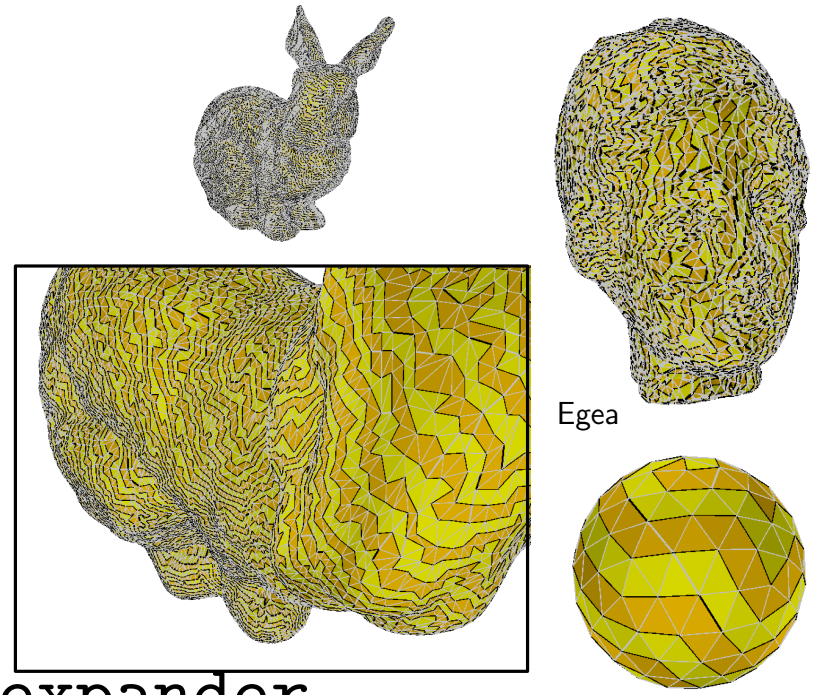
isolated vertices

LR: Laced Ring data structure



meshes sorted according to % of degree 6 vertices

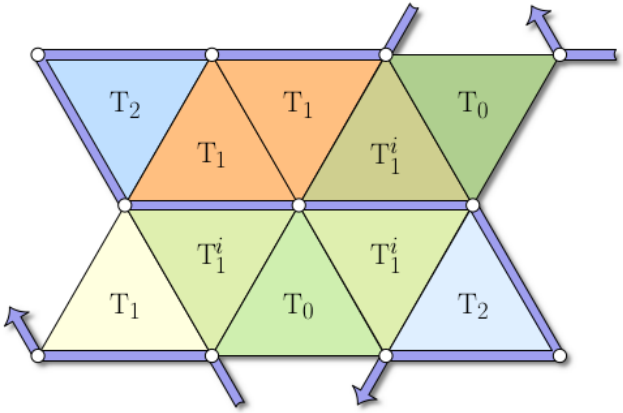
LR (Gurung et al. 2011)



Ring-expander

```
c = s;                          // start at the seed corner s
c.n.v.m = c.p.v.m = true;       // mark vertices as visited
do {
    if (!c.v.m) c.v.m = c.t.m = true;      // invade c.t
    else if (!c.t.m) c = c.o;    // go back one triangle
    c = c.r;   // advance to next ring edge on the right
} while (c != s.o);        // until back at the beginning
```

triangle classification

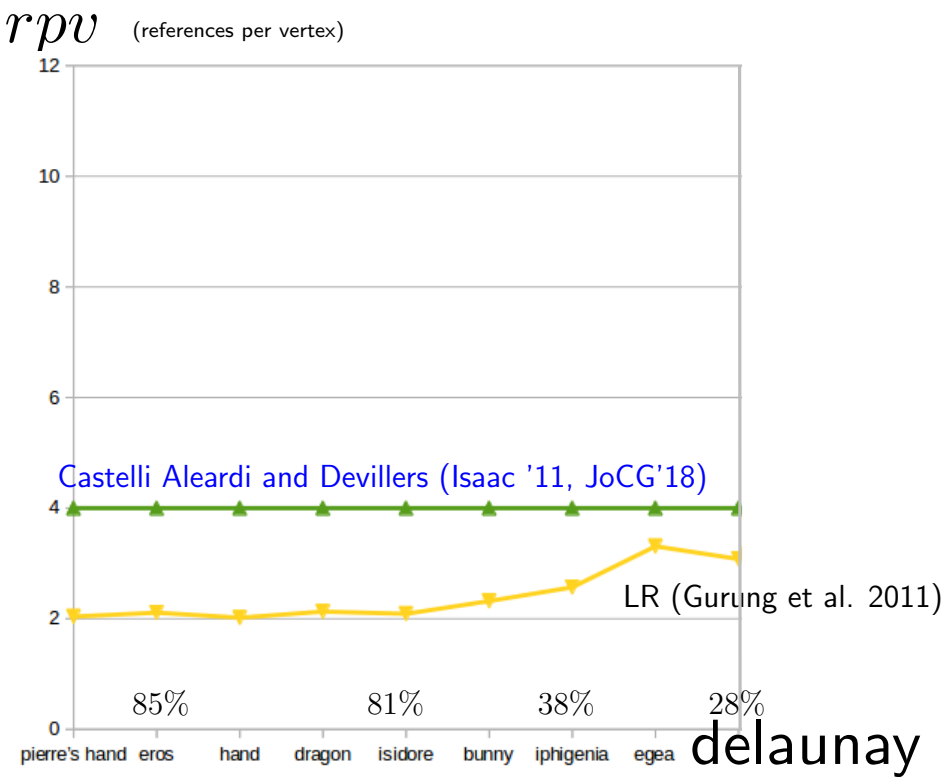


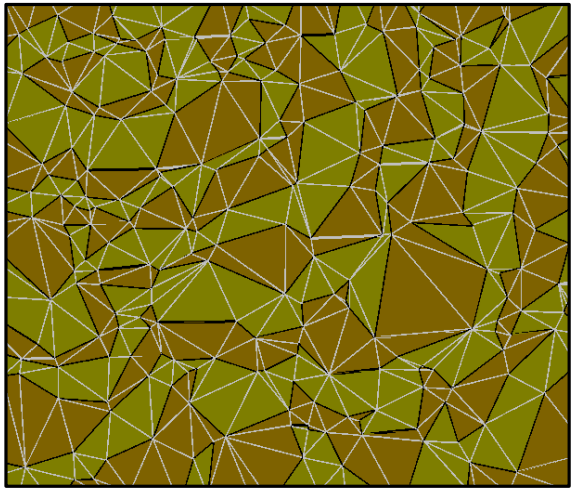
$$size(LR) := \boxed{2v_r} + v_i + 6|T_0| + 3|T_1^i + T_2^i|$$

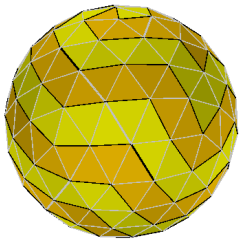
isolated vertices

LR: Laced Ring data structure

LR (Gurung et al. 2011)

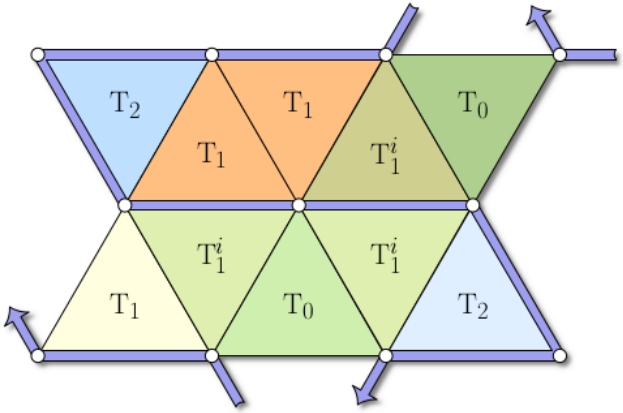






delaunay

triangle classification



Ring-expander

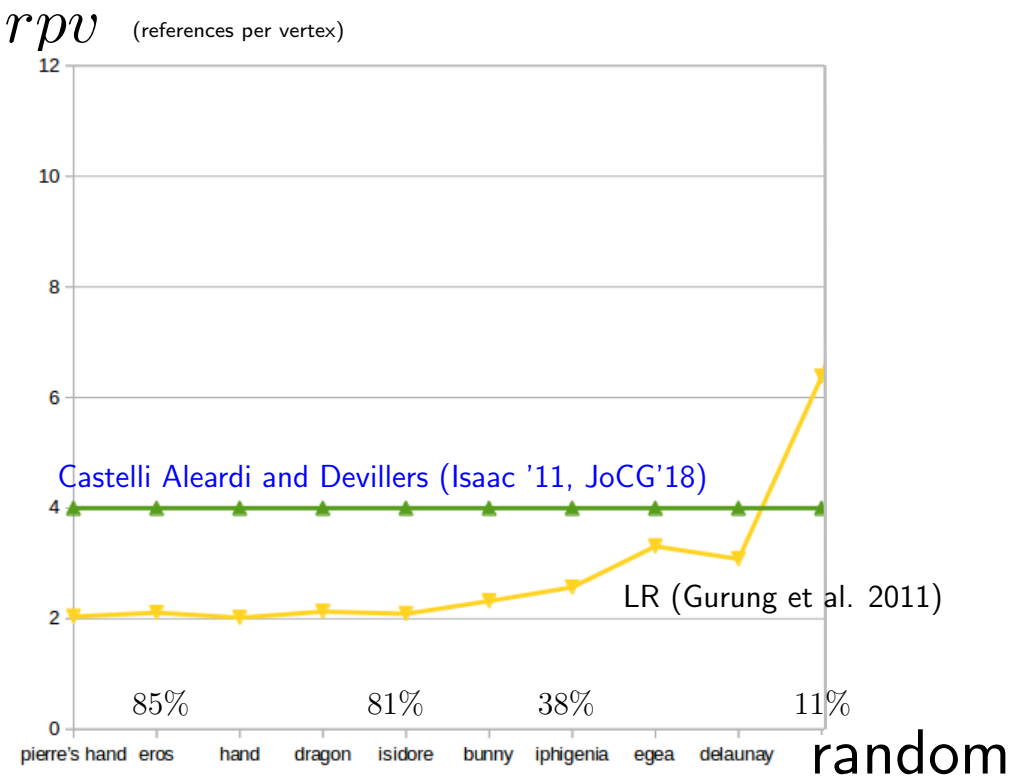
```
c = s;                          // start at the seed corner s
c.n.v.m = c.p.v.m = true;       // mark vertices as visited
do {
    if (!c.v.m) c.v.m = c.t.m = true;      // invade c.t
    else if (!c.t.m) c = c.o;    // go back one triangle
    c = c.r;   // advance to next ring edge on the right
} while (c != s.o);          // until back at the beginning
```

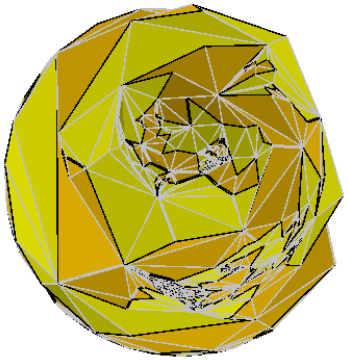
$$size(LR) := \boxed{2v_r} + v_i + 6|T_0| + 3|T_1^i + T_2^i|$$

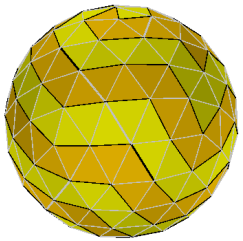
isolated vertices

LR: Laced Ring data structure

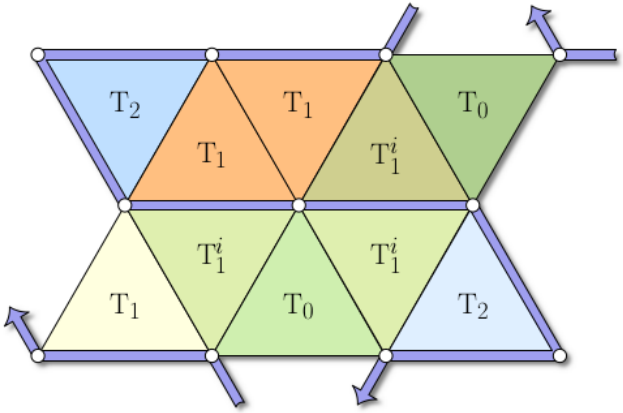
LR (Gurung et al. 2011)







triangle classification



Ring-expander

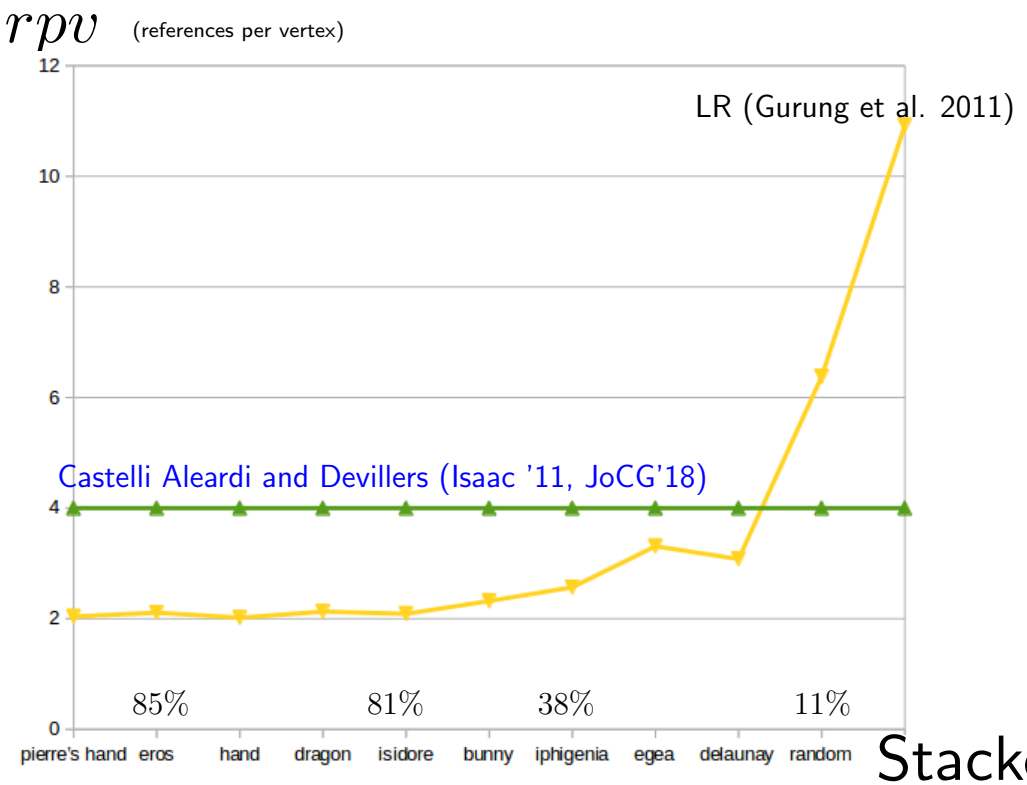
```
c = s;                          // start at the seed corner s
c.n.v.m = c.p.v.m = true;       // mark vertices as visited
do {
    if (!c.v.m) c.v.m = c.t.m = true;      // invade c.t
    else if (!c.t.m) c = c.o;    // go back one triangle
    c = c.r;   // advance to next ring edge on the right
} while (c != s.o);          // until back at the beginning
```

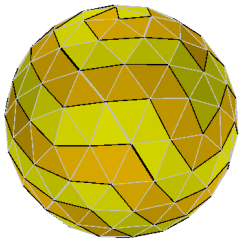
$$size(LR) := \boxed{2v_r} + v_i + 6|T_0| + 3|T_1^i + T_2^i|$$

isolated vertices

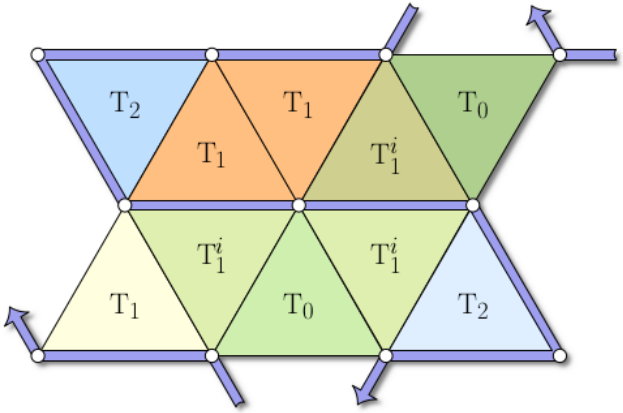
LR: Laced Ring data structure

LR (Gurung et al. 2011)





Stacked triangle classification



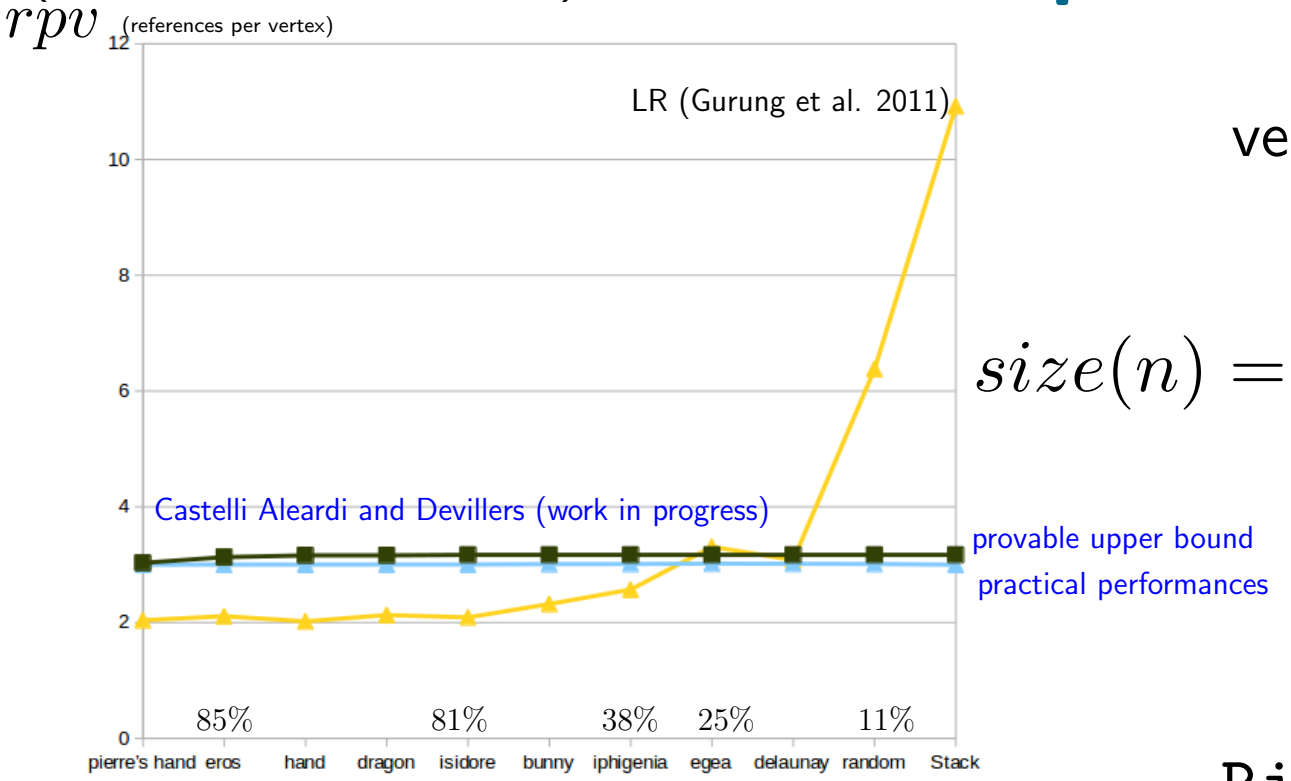
Ring-expander

```
c = s;                          // start at the seed corner s
c.n.v.m = c.p.v.m = true;       // mark vertices as visited
do {
    if (!c.v.m) c.v.m = c.t.m = true;      // invade c.t
    else if (!c.t.m) c = c.o;    // go back one triangle
    c = c.r;   // advance to next ring edge on the right
} while (c != s.o);          // until back at the beginning
```

$$size(LR) := \boxed{2v_r} + v_i + 6|T_0| + 3|T_1^i + T_2^i|$$

isolated vertices

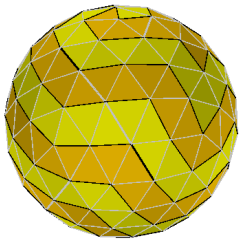
(Gurung et al. 2011) LR vs. adaptive CDT (Castelli Devillers, 20xx)



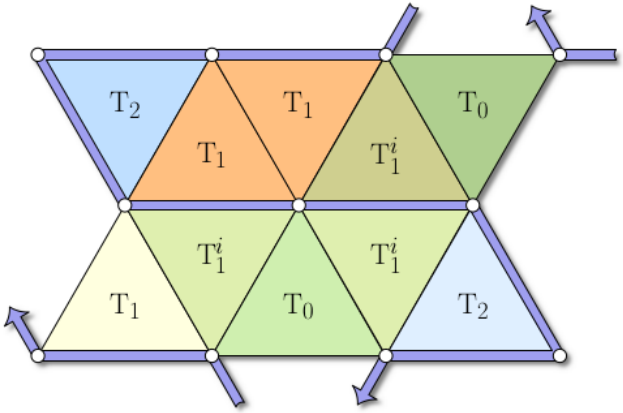
vertex degree distribution

$$(p_3, p_4, \ldots, p_{n-1})$$

$$size(n) = 3n + min(\tfrac{1}{6}, 2 \cdot \textstyle\sum_{i=7}^{n-1} p_i)$$



triangle classification



Ring-expander

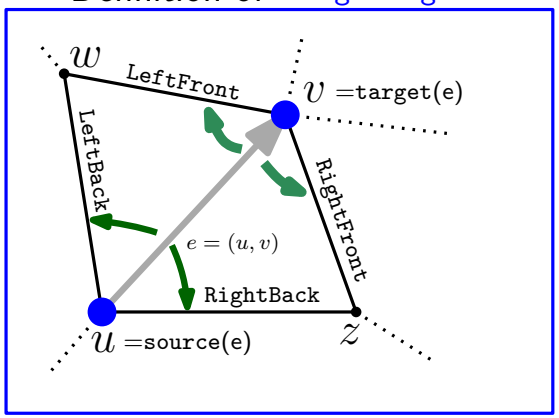
```
c = s;                          // start at the seed corner s
c.n.v.m = c.p.v.m = true;       // mark vertices as visited
do {
    if (!c.v.m) c.v.m = c.t.m = true;      // invade c.t
    else if (!c.t.m) c = c.o;    // go back one triangle
    c = c.r;   // advance to next ring edge on the right
} while (c != s.o);          // until back at the beginning
```

$$size(LR) := \boxed{2v_r} + v_i + 6|T_0| + 3|T_1^i + T_2^i|$$

isolated vertices

Winged Edge DS (size $19n$) (Baumgart, 1972)

Definition of Winged-edge



$0 \leq e \leq (3n - 6) - 1$

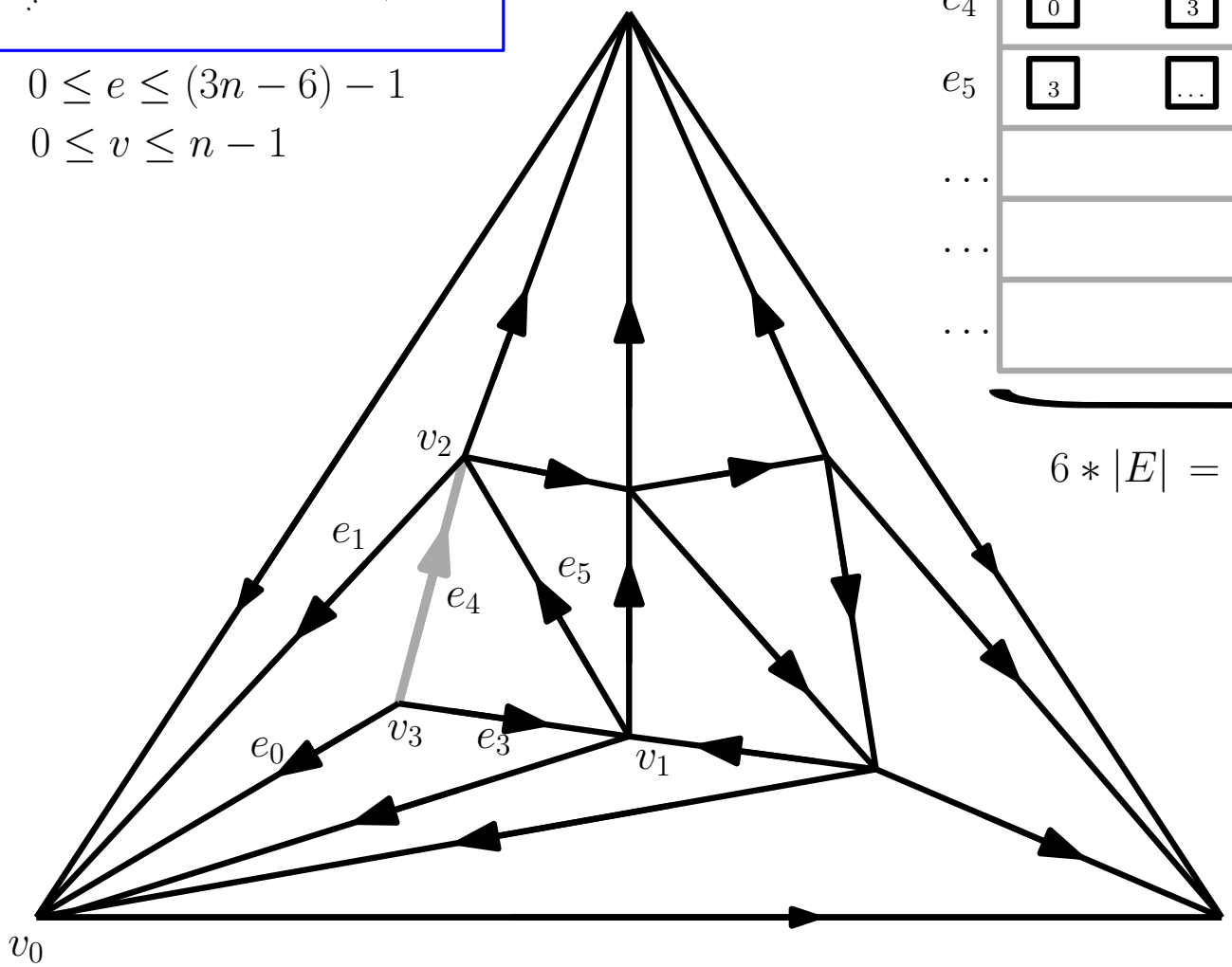
$0 \leq v \leq n - 1$

	LeftBack	RightBack	LeftFront	RightFront	source	target
e_0						
e_1						
e_2						
e_3						
e_4	0	3	1	5	3	2
e_5	3	...	4	...	1	2
...						
...						
...						

$6 * |E| = 6 * (3n - 6) \approx 18n$ entries

	incident edge
v_0	e_0
v_1	e_3
v_2	
v_3	e_4
v_4	
v_5	
	...
	...
	...

n entries




```
Degree(u)
{
        e = Edge(u);
        f = e; d = 0;
        do{
                if u =Source(f) f =LeftBack(f) ;
                else            f =RightFront(f) ;
                d = d + 1;
        } while f ≠ e;
        return d;
}
```

Our first simple Compact DS (size $6n$)

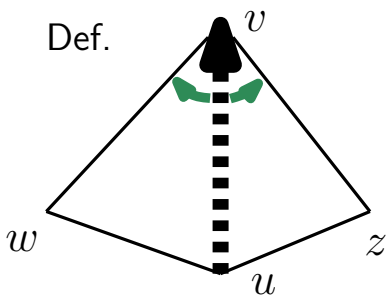
(Castelli Aleardi, Devillers, 2011)

$e := (u, v) \qquad 0 \le v \le n-1$

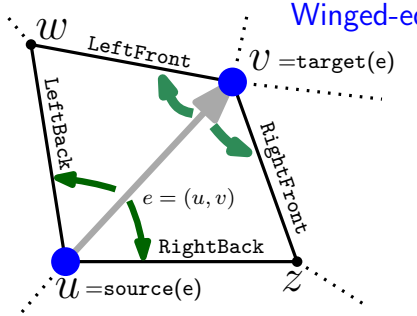
$0 \le e \le 3n$

store only 2 references per edge

Def.

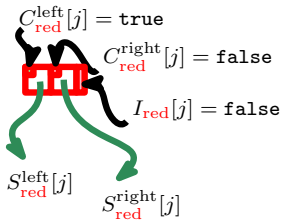


Winged-edge



retrieve

$$(w, u) := \begin{cases} (e+1)\%3 & \text{case } 1 \\ (T[e]+2)\%3 & \text{case } 2 \\ T[T[e]] & \text{case } 3 \end{cases}$$

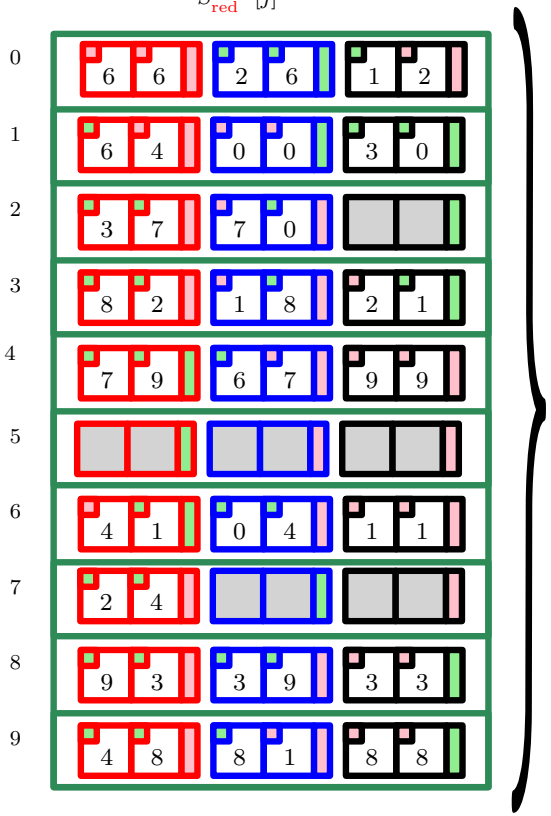


$u := \texttt{source}(e) = e/3$

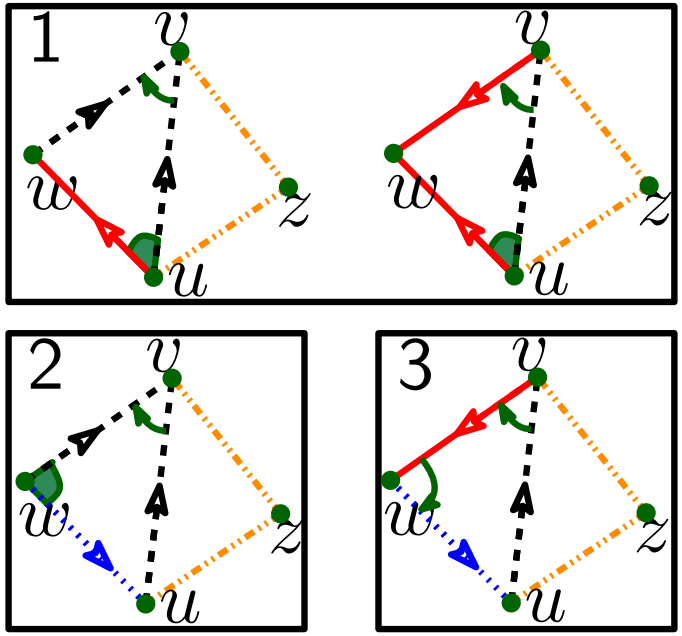
(w,v):=T[2e]

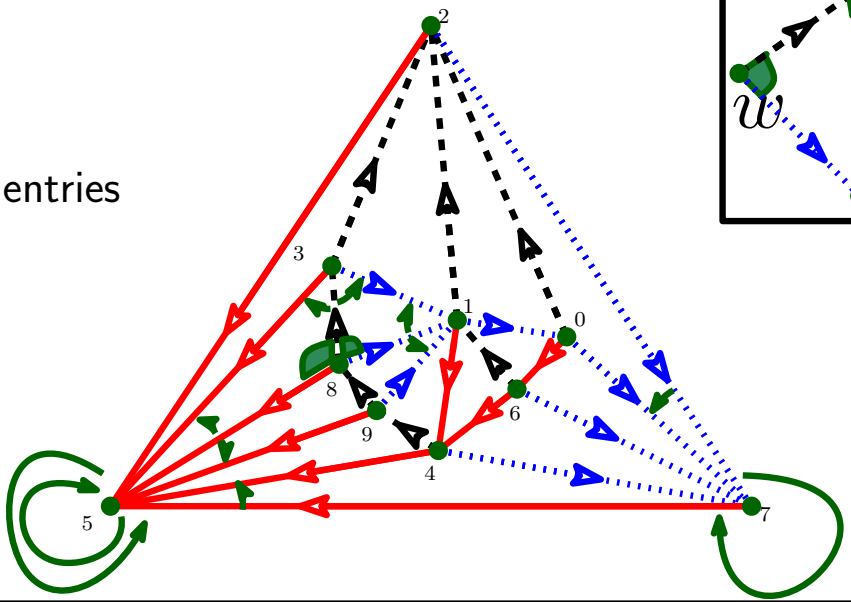
$(z, v) := T[2e+1]$

$\texttt{color}(e) = e\%3$



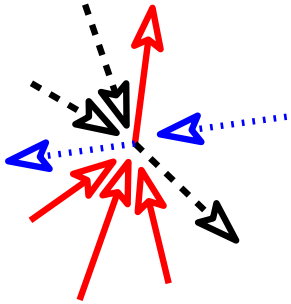
$6 * n$ entries



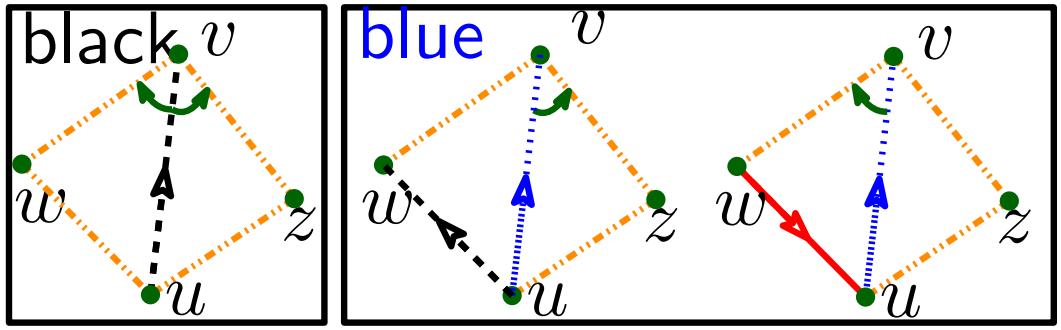


More compact DS (size $5n$): use maximal Schnyder woods

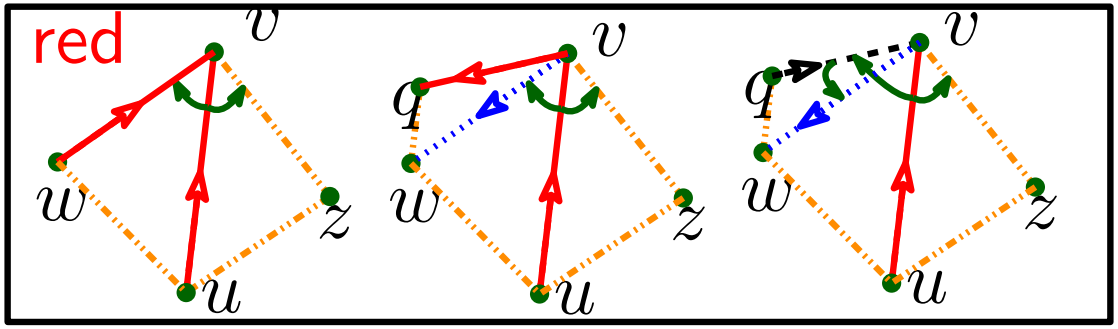
(less redundant and "more difficult to implement")

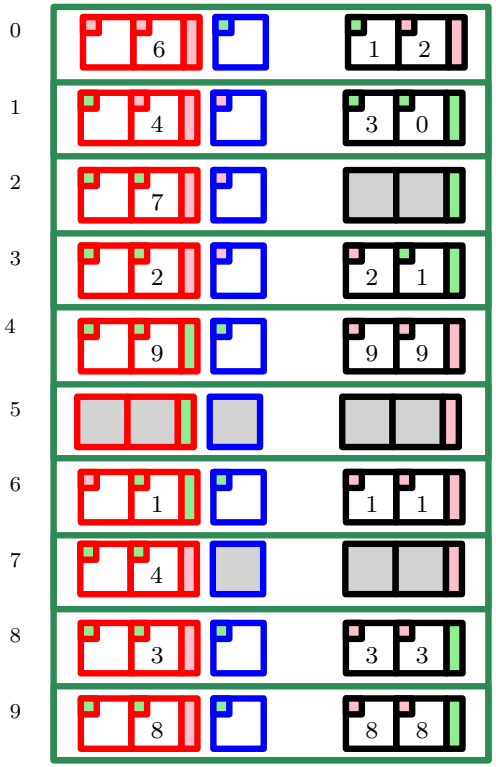


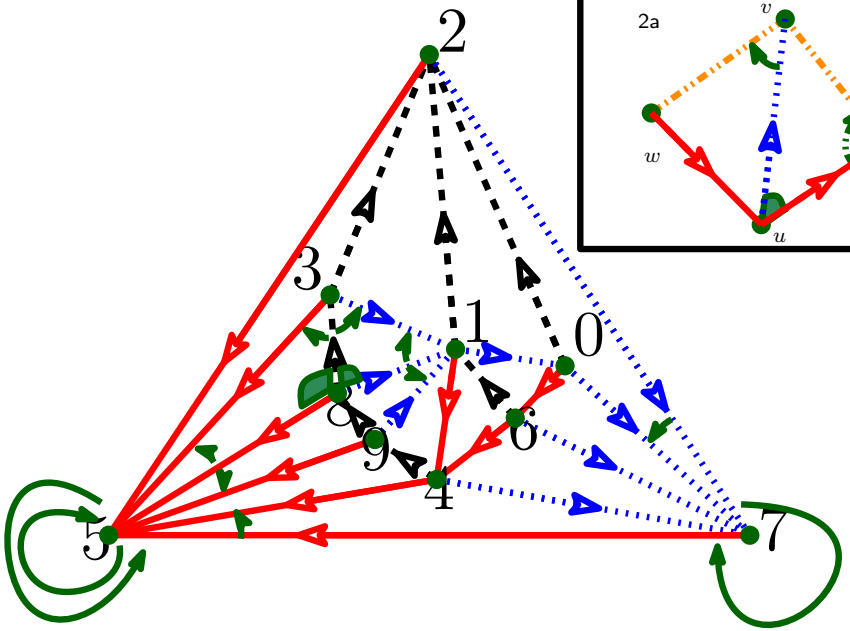
remove one blue column

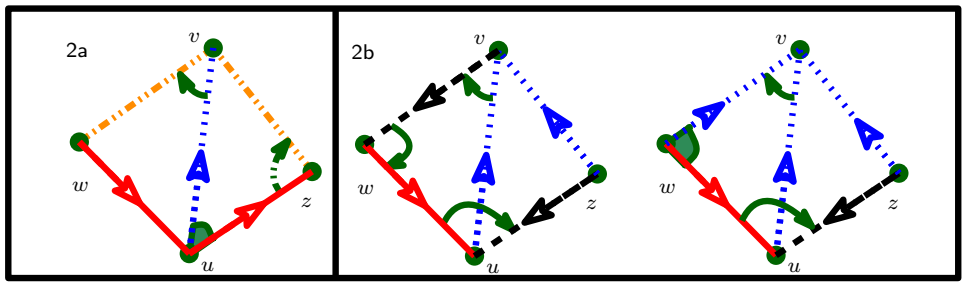


(sometimes) change one red left reference (for red edges)

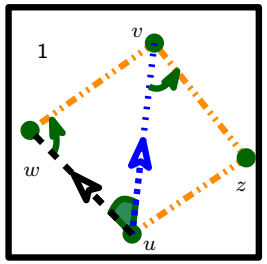








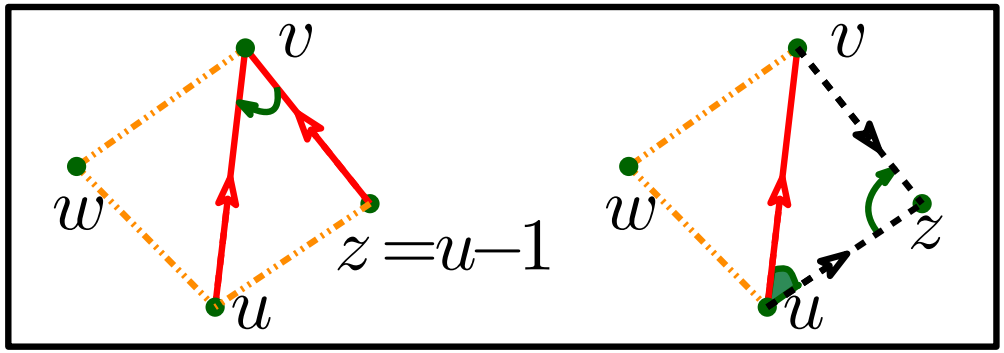
only difficult case

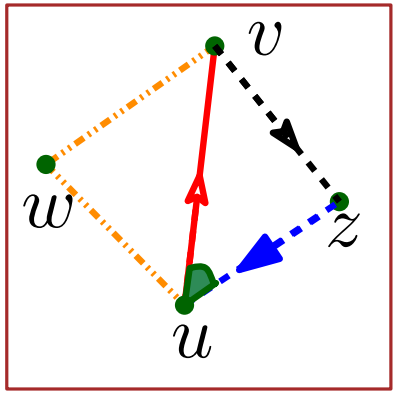


More compact DS (size $4n$): use maximal Schnyder woods

(reorder vertices according to a BFS traversal of T_0)

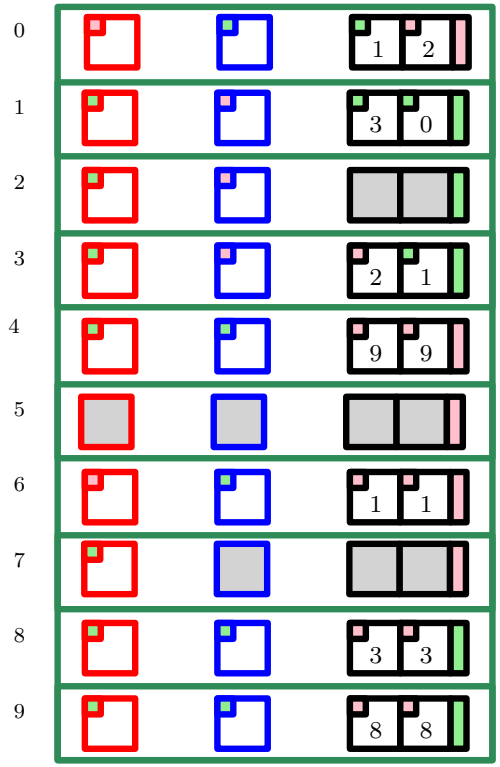
("slightly more difficult to implement")

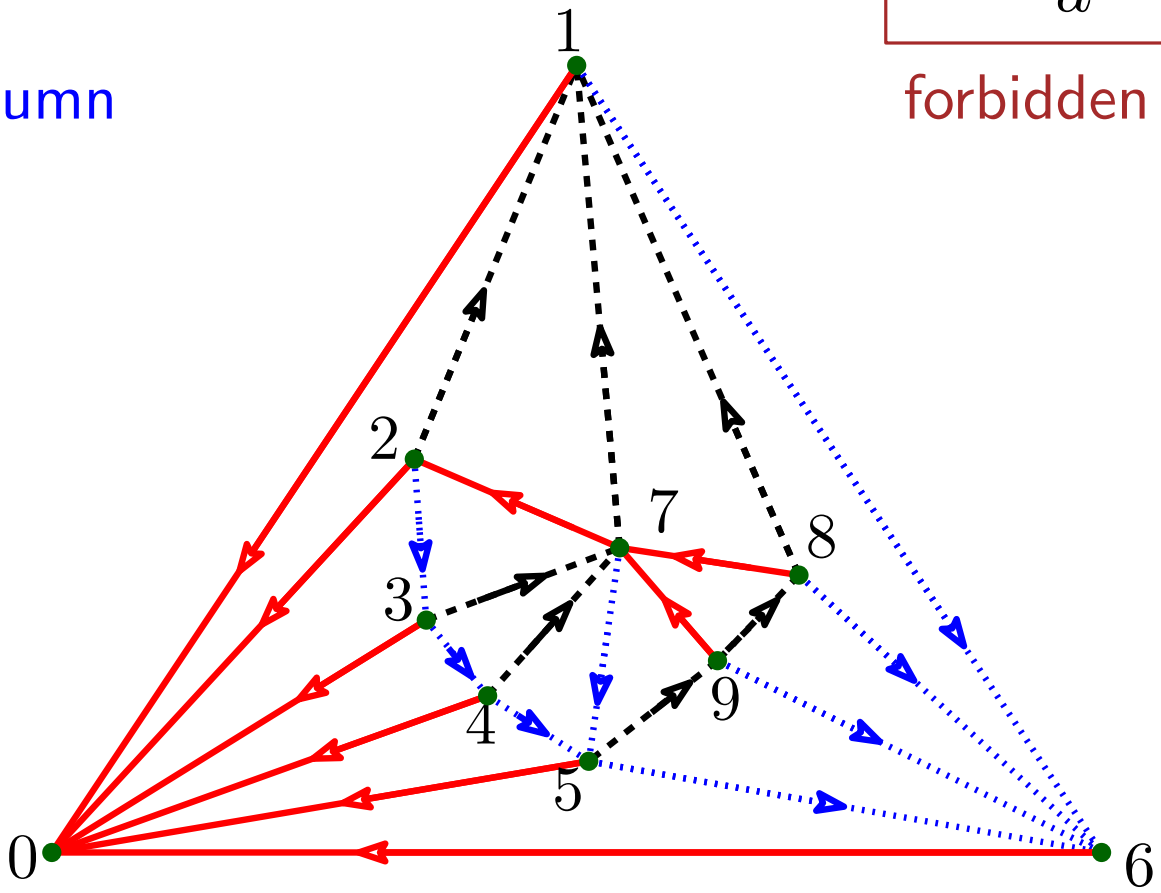




forbidden case

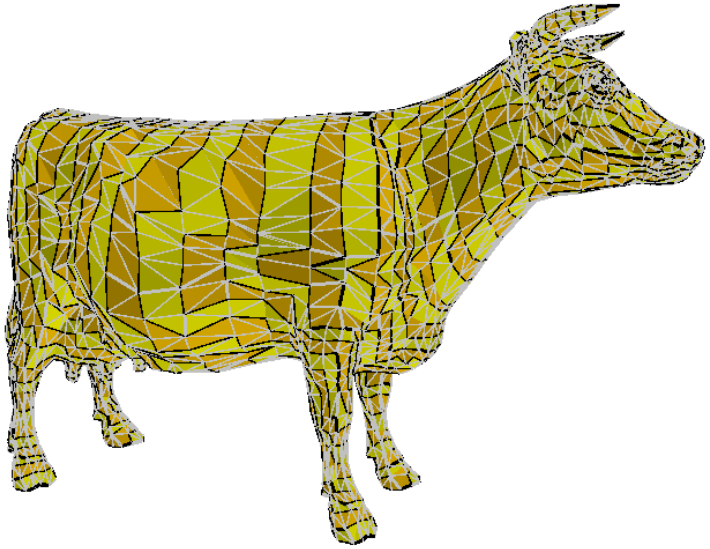
remove one red column

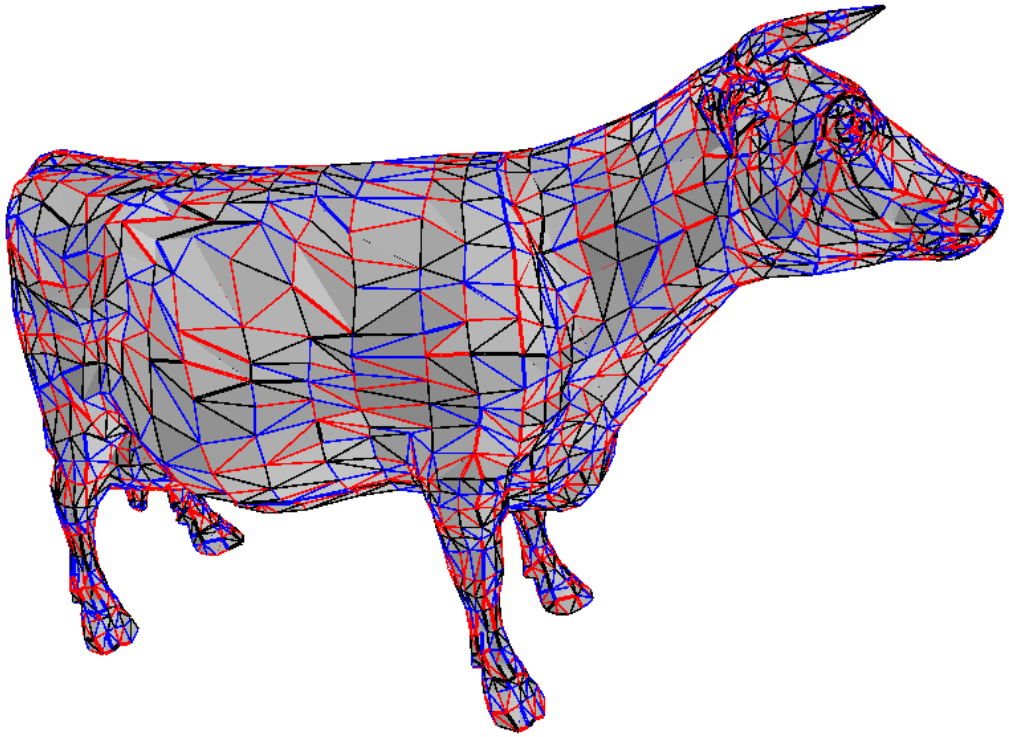


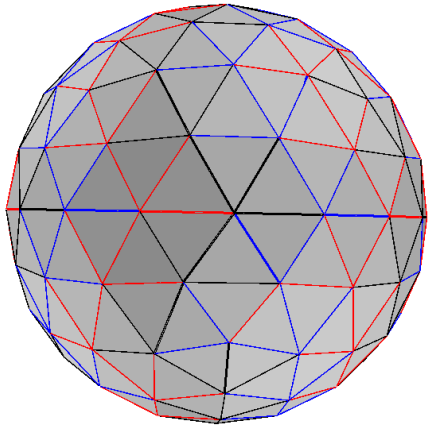


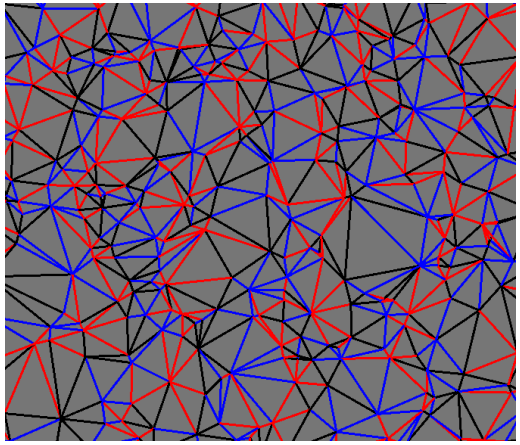
More compact DS: size $< 4n$?

(can we exploit the regularity of the triangulation?)



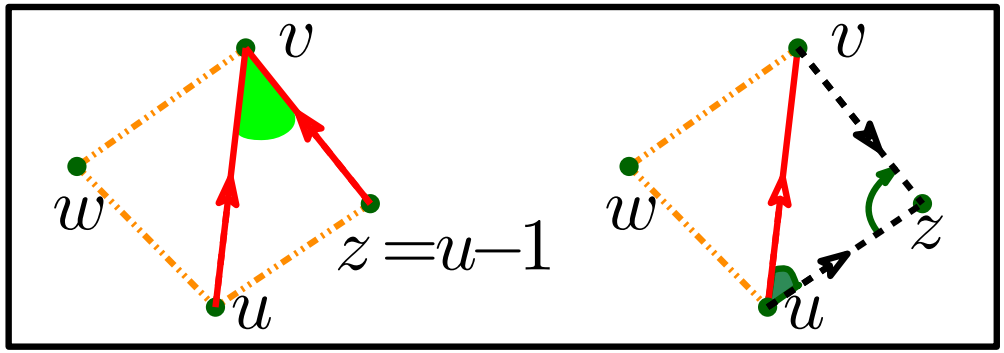


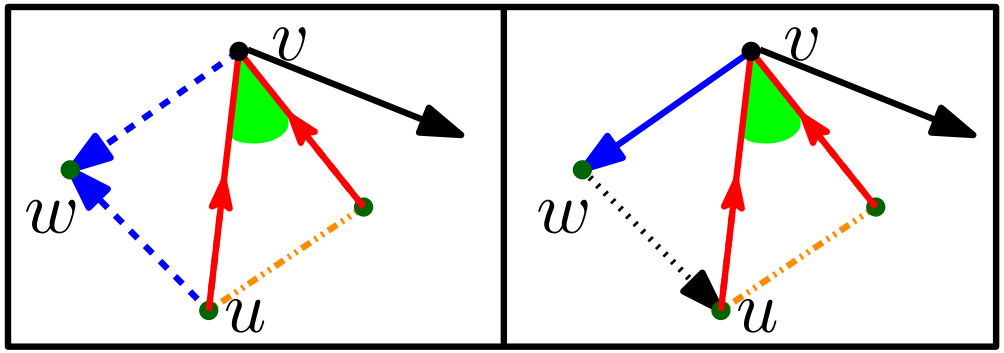




Adaptive compact DS (size $\approx 3n$) (alllow slightly slower navigation)

("more difficult to implement?")

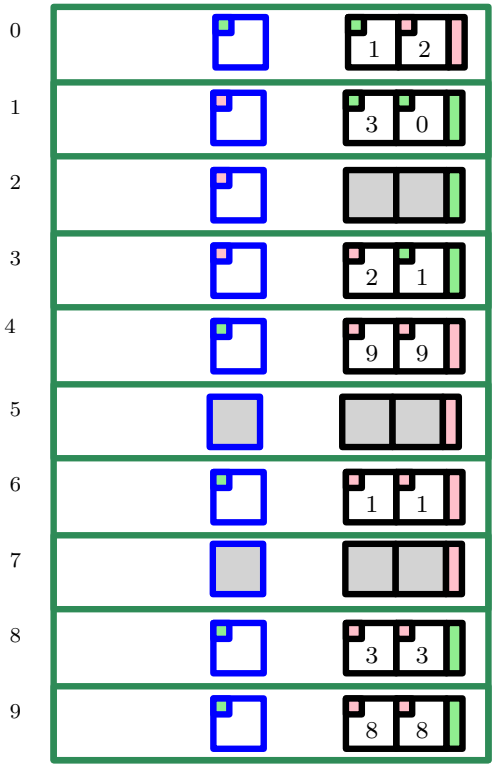


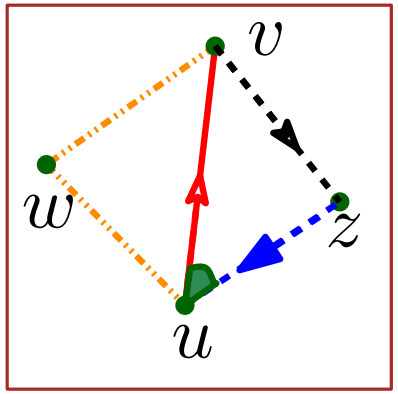


we have a problem: impossible to reach w

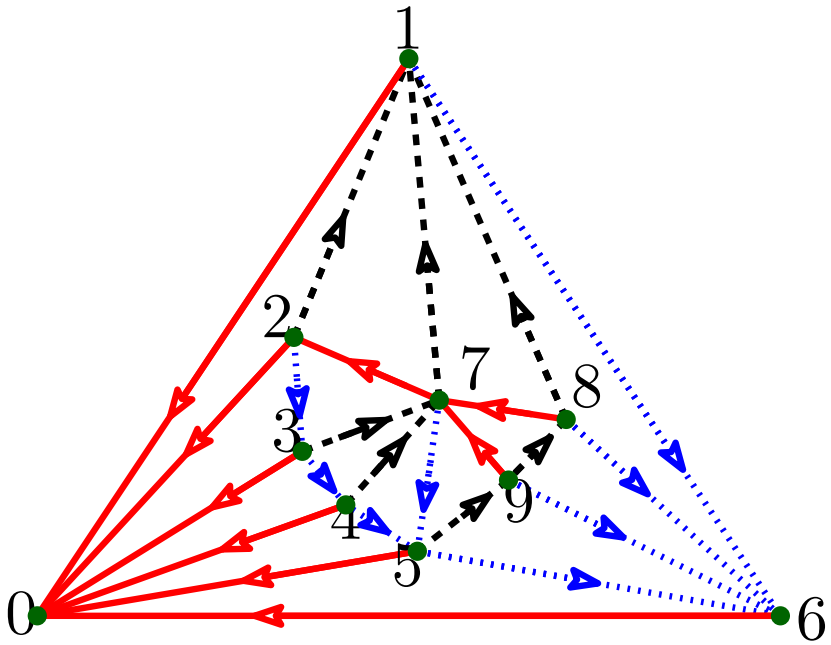
going to the left is more difficult

remove all red columns



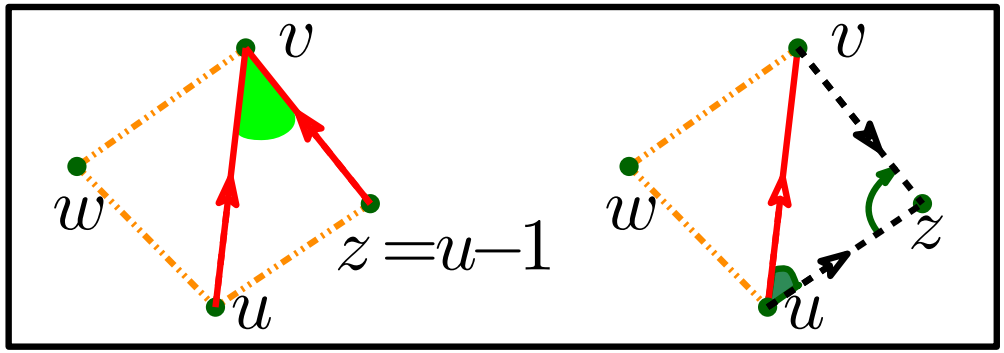


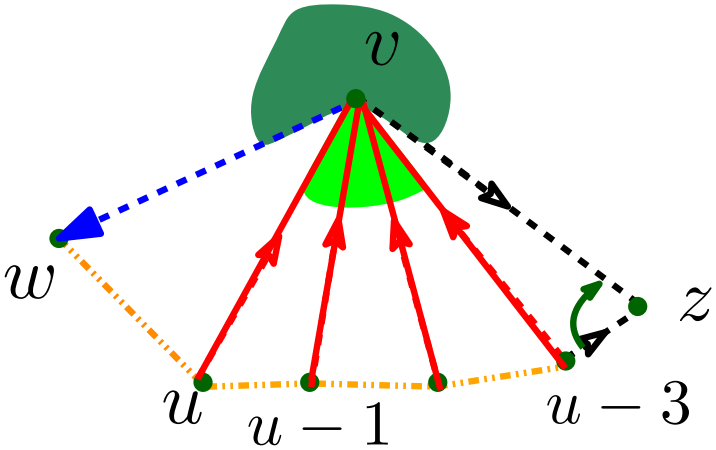
forbidden case



Adaptive compact DS (size $\approx 3n$) (alllow slightly slower navigation)

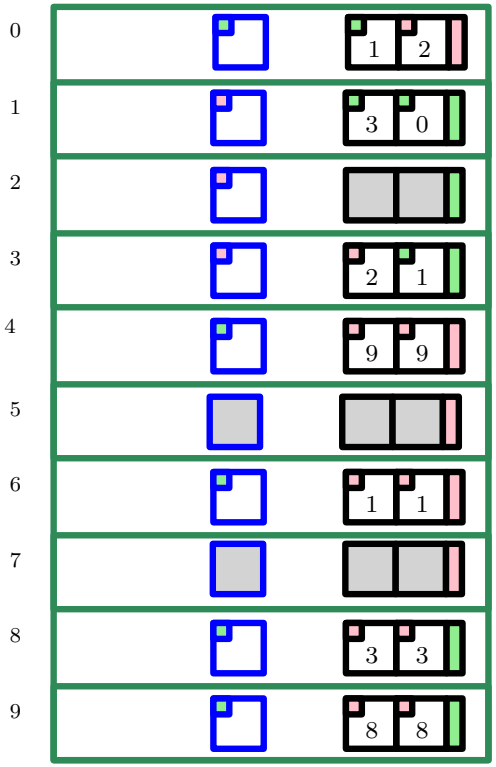
("more difficult to implement?")

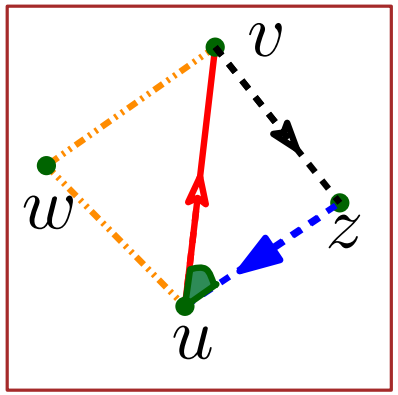




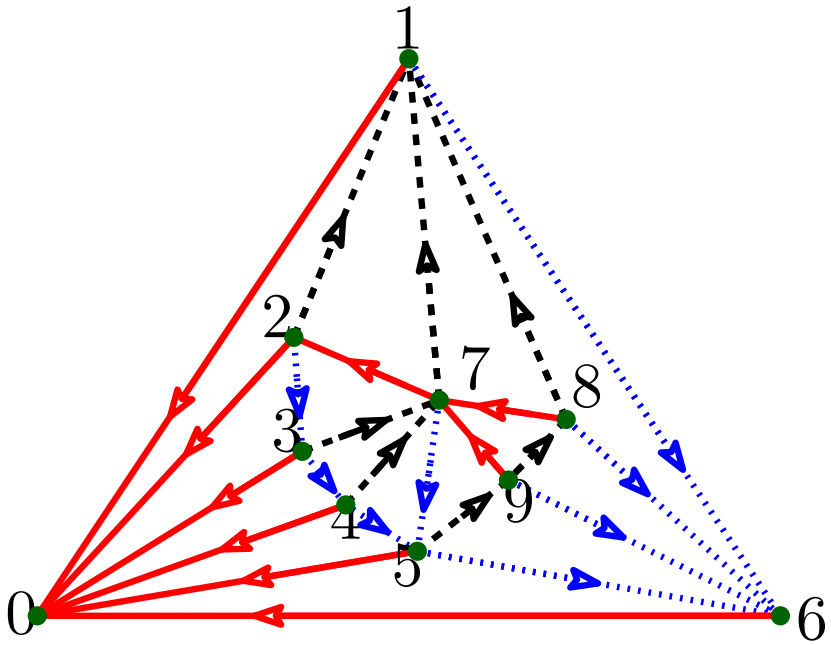
solution for going left: jump to the right

remove all red columns



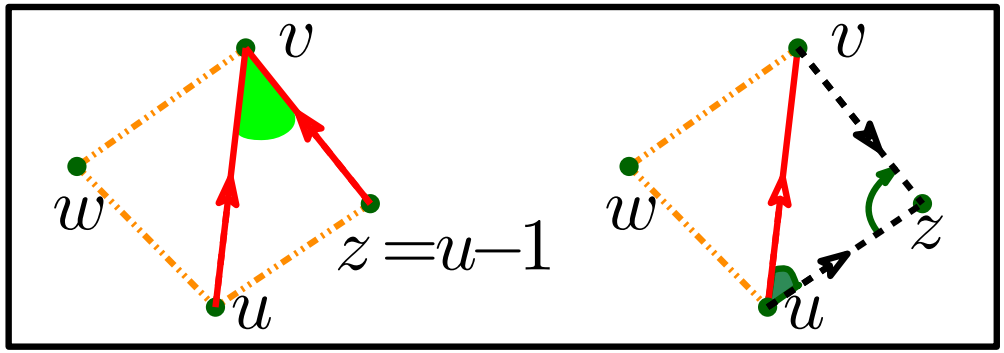


forbidden case

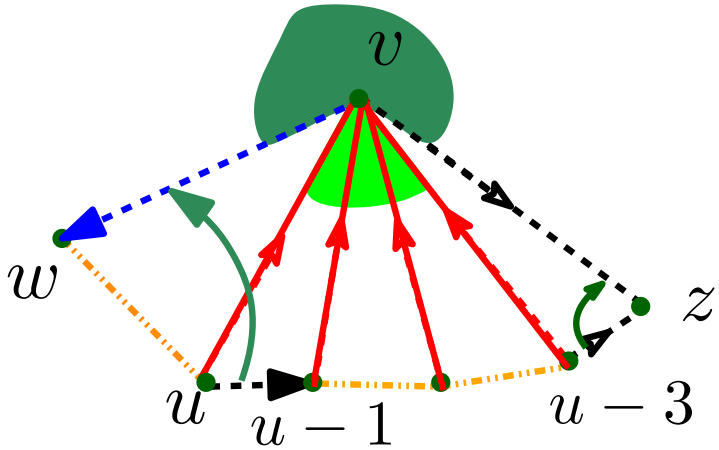


Adaptive compact DS (size $\approx 3n$) (alllow slightly slower navigation)

("more difficult to implement?")

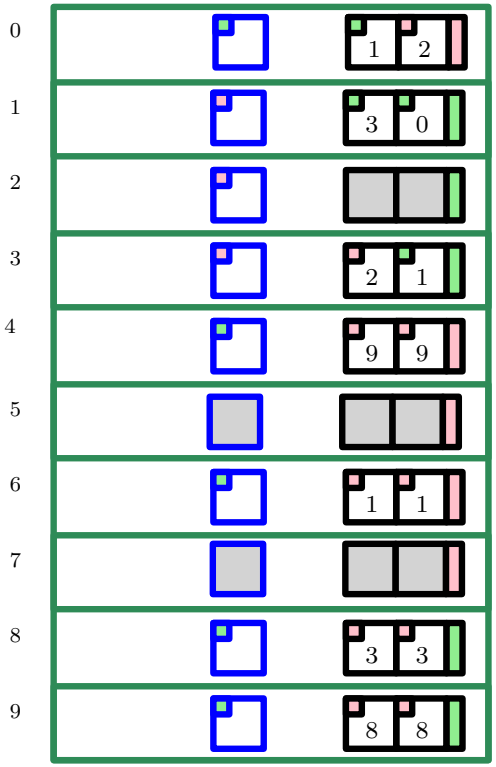


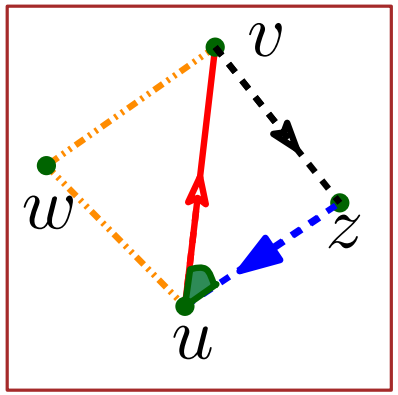
solution for going left: jump to the right



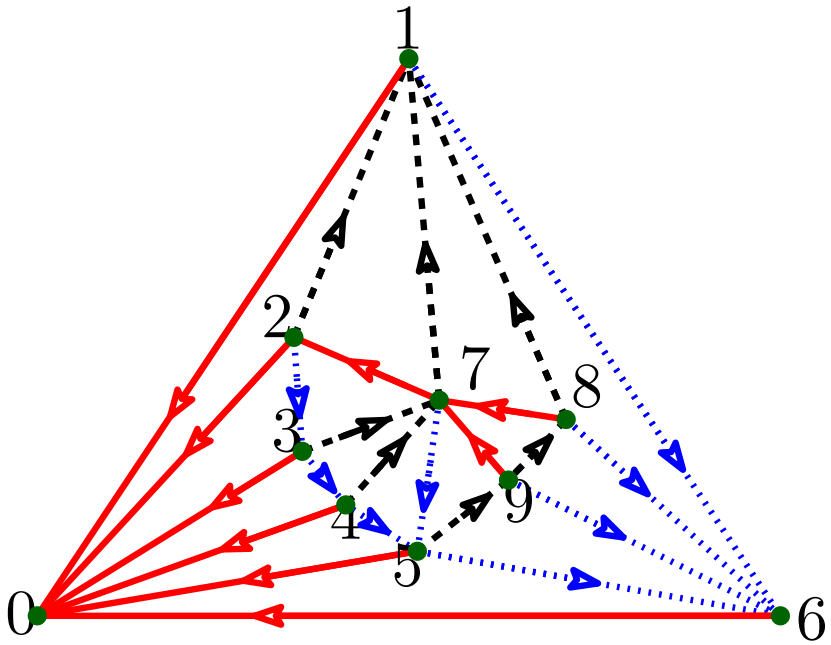
use the closest black edge
(not useful for right navigation)

remove all red columns



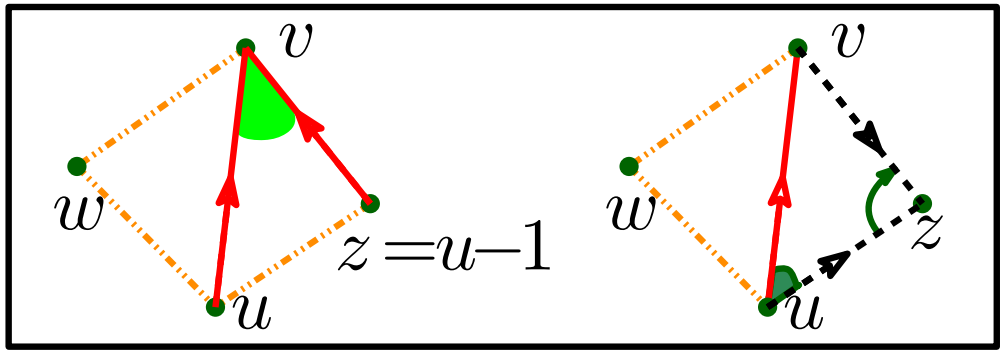


forbidden case

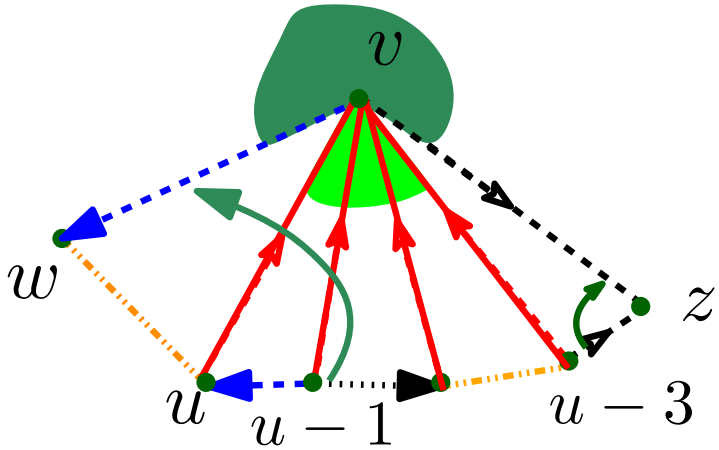


Adaptive compact DS (size $\approx 3n$) (alllow slightly slower navigation)

("more difficult to implement?")

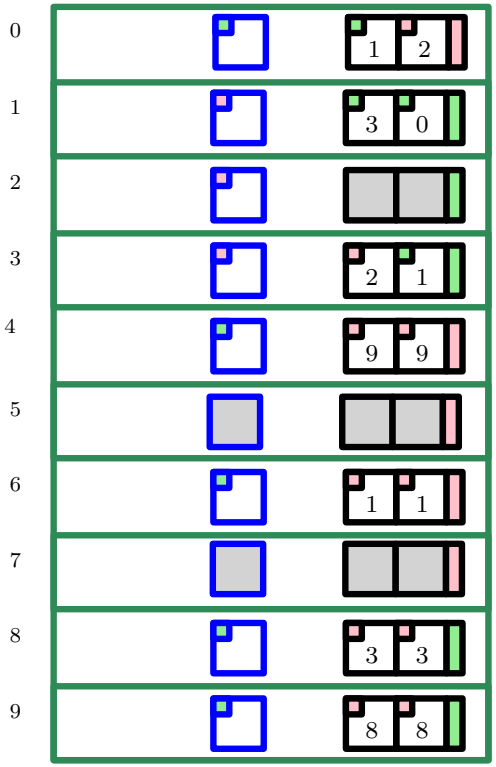


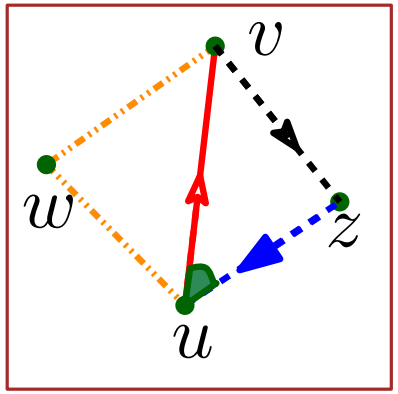
solution for going left: jump to the right



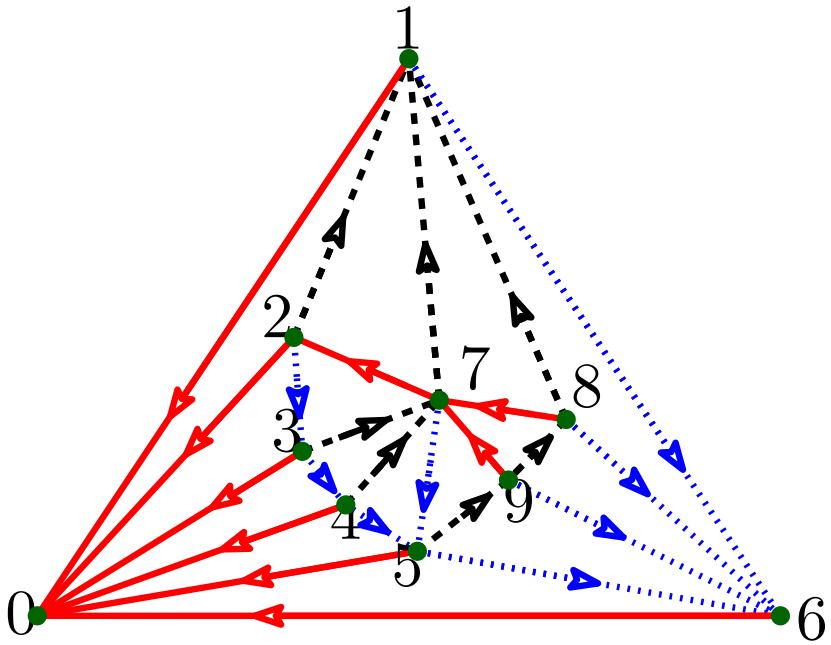
use the closest black edge
(not useful for right navigation)

remove all red columns



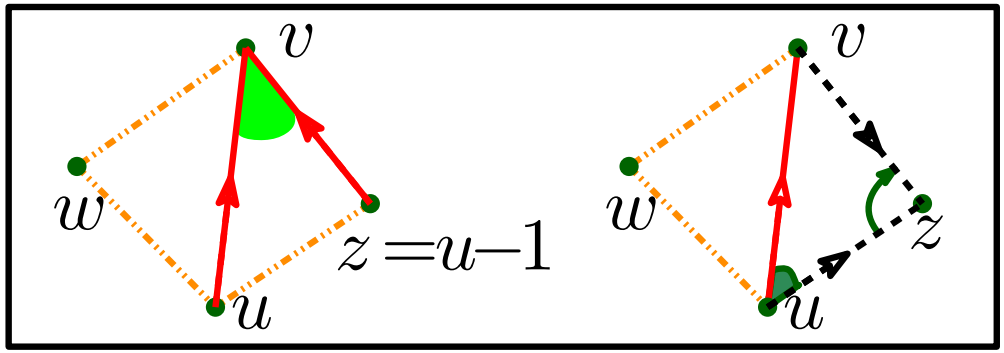


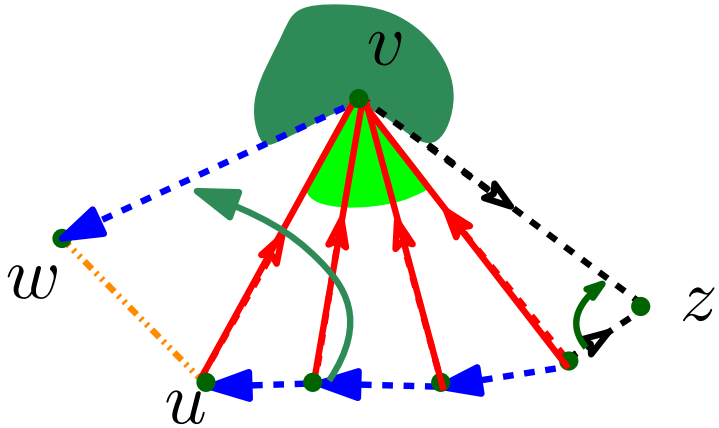
forbidden case



Adaptive compact DS (size $\approx 3n$) (alllow slighlty slower navigation)

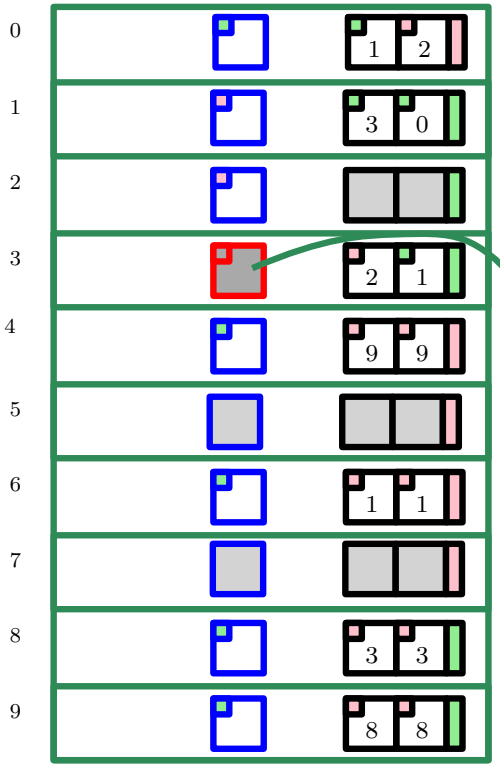
("more difficult to implement?")

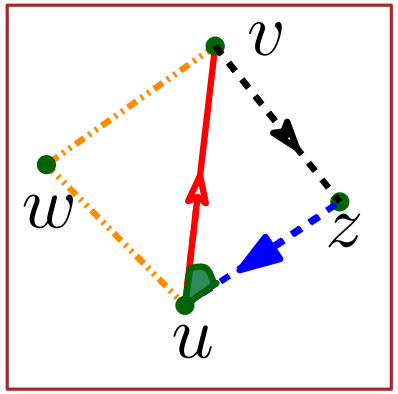




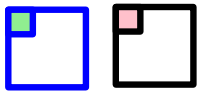
solution for going left: jump to the right

remove all red columns

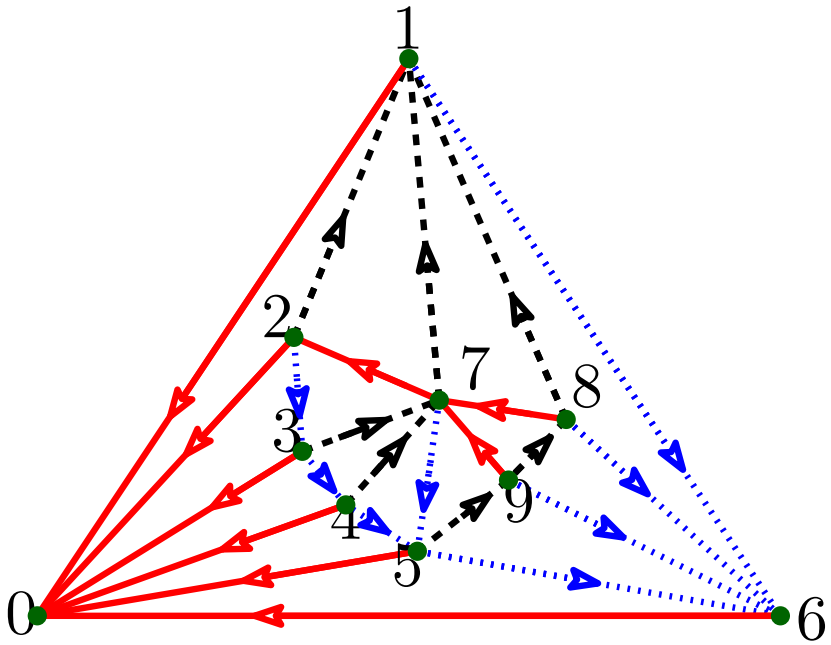




forbidden case

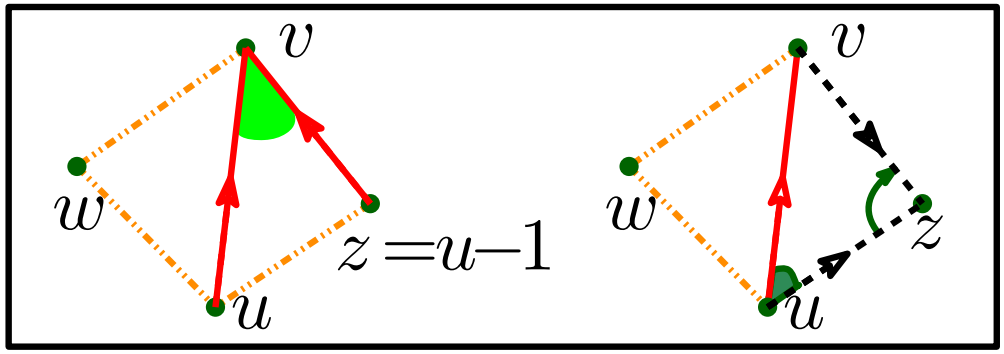


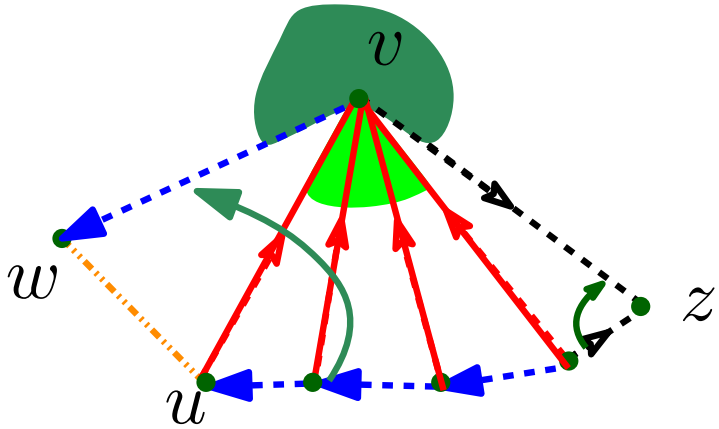
if the black edge is too far
store an additional reference



Adaptive compact DS (size $\approx 3n$) (alllow slower navigation)

("more difficult to implement?")

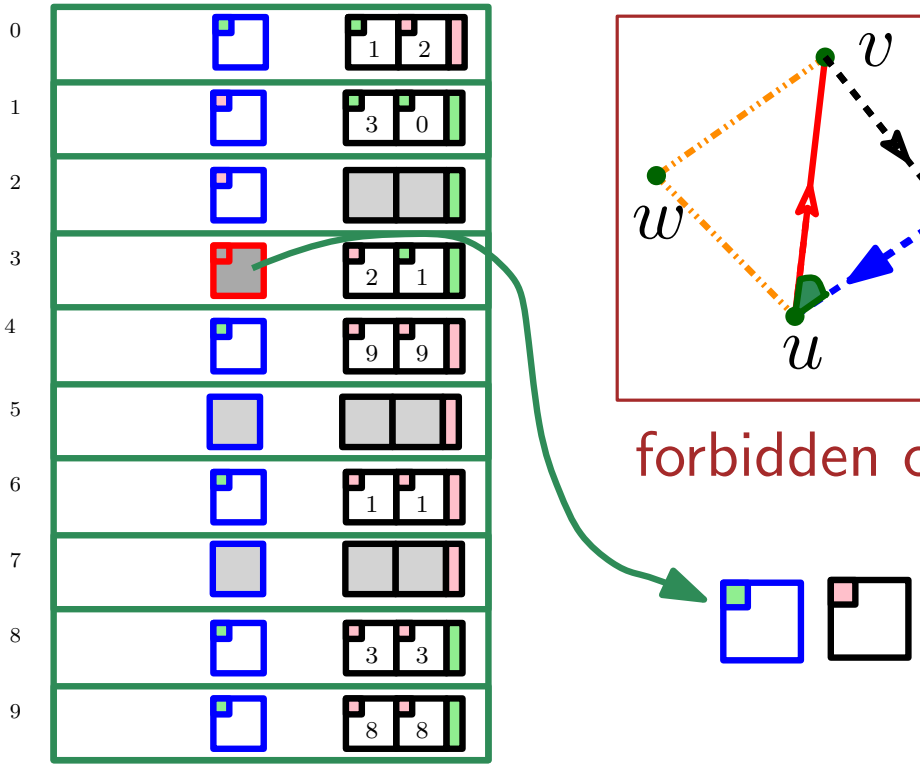


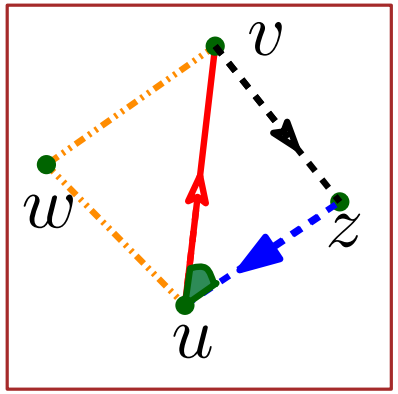


solution for going left: jump to the right

fix a small value k

remove all red columns





forbidden case

upper bound depending of vertex degree distribution

$$size(n) = 3n + 2(\sum_{i=k+3}^{n-1} p_i)$$

for $k = 4$

$$size(n) = 3n + 2(\sum_{i=7}^{n-1} p_i)$$