Graph Theory MATH 442 Term 2 2018-09-11

CPSC 536E ~~topics~~

matrix representation of graph ① computing path

② Google's page rank how connected is the graph?

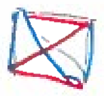
③ useful for graph drawing eigenvalue

major topics:

TODO: ① spectral graph theory

~~② Ramsey's theory~~ Complete graph: color edge red or blue, smallest number of vertices.

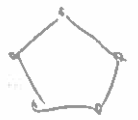
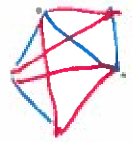
clique:



$R(3,3)=3$ = size of smallest complete graph so that for all red/blue colorings the exists either a red 3-clique or a blue 3-clique.

$R(4,4)=18$

$R(5,5)=43$

③ Random walks on graphs

Give a graph, how many perfect matchings are there?

① ④ planar graphs

characterization drawn on a plane without edges crossing.

Pointers for things to look at:

To get started with planar graphs

① Things for next time: things about planar graphs

how to tell? algorithms for planar detection

② graph drawing

② GRAPH DRAWING / 4 things interested in

ALGORITHMS FOR THE VISUALIZATION OF GRAPH,

italian GIUSEPPE DI BATTISTA

Australian PETER EADES

italian ROBERTO TAMASSIA

Greek IOANNIS G. TOLLIS

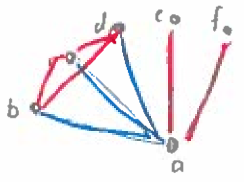
③ Graph Theory with Applications

J.A. Bondy U.S.R. Murty.

read chapter on planar graphs

Ramsey number
Proof that $R(3,3) = 6$

In a complete graph (K_6) for node a adjacent to all other 5 nodes, color them red or blue.



No matter how you color it, at least 3 edges are colored with the same color (either red or blue).

WLOG, let's say a is connected with b, c, d colored blue.

To avoid a clique with a, bc, cd, bd has to be colored red. In this way, b, c, d form a red 3-clique.

Thus, K_6 must have either a red 3-clique or a blue 3-clique. $\square$

Bondy and Murty

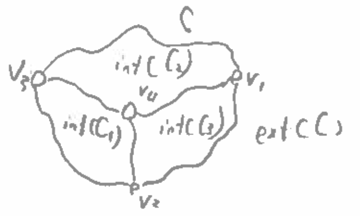
[1.1] P5 : <u>planar graph:</u> A graph which can be drawn in the plane in such a way that edges meet only at points corresponding to their common ends is called a <u>planar graph</u>, and such a drawing is called a <u>planar embedding</u> of the graph.

[10.1] P243 <u>planar graph:</u> A graph is said to be embeddable in the plane, or planar, if it can be drawn in the plane so that its edges intersect only at their ends.

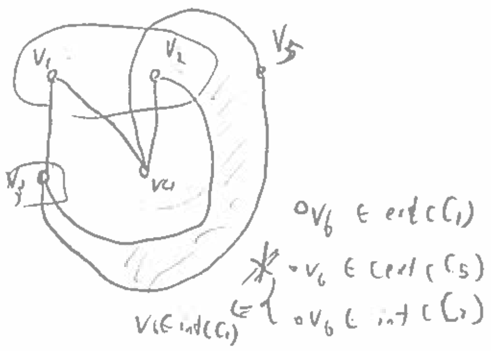
[10.1] P244 <u>plane graph:</u> we often refer to a planar embedding $\widetilde{G}$ of a planar graph G as a plane graph; and we call the vertices of $\widetilde{G}$ points and its edges lines.

[10.1] P245 <u>THE JORDAN CURVE THEOREM</u>: Any simple closed curve C in the plane partitions the rest of the plane into two <u>disjoint arcwise-connected open sets.</u>
└ called interior int(C) and exterior ext(C).

[10.1] P245 <u>K_5 is nonplanar:</u>



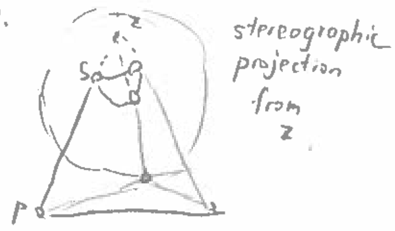
$K_{3,3}$:



[10.1] P246 <u>subdivision</u> : Any graph derived from a graph G by a sequence of edge subdivisions is called a <u>subdivision of G</u> or a <u>G-subdivision</u>

[10.1] P246 . Proposition 10.3 : A graph G is planar if and only if every subdivision of G is planar.

[10.1] P247 Theorem 10.4: A graph G is embeddable on the plane if and only if it is embeddable on the sphere.



[10.2] P250 Proposition 10.5: Let G be a planar graph, and let f be a face in some planar embedding of G. Then G admits a planar embedding whose outer face has the same boundary as f.
Proof idea – consider an embedding $\widetilde{G}$ of G on the sphere.

[10.2] P250 Theorem 10.6: <u>THE JORDAN-SCHÖNFLIESS THEOREM</u> ?
Any homeomorphism of a simple closed curve in the plane onto another simple closed curve can be extended to a homeomorphism of the plane.

[10.2] P251 Theorem 10.7: In a nonseparable plane graph other than K_1 or K_2, each face is bounded by a cycle.
? ?

[10.2] P252 <u>dual</u> : Given a plane graph G, one can define a second graph G^* as follows. Corresponding to each face f of G there is a vertex f^* of G^*, and corresponding to each edge e there is an edge e^* of G^*. Two vertices f^* and g^* are joined by the edge e^* in G^* iff their corresponding faces f and g are separated by the edge e in G. Observe that if e is a cut edge of G, then f = g, so e^* is a loop of G^*; conversely, if e is a loop of G, the edge e^* is a cut edge of G^*. The graph G^* is called the dual of G.

[10.3] P259 EULER'S FORMULA:

For a connected plane graph G, Proof by induction

$$v(G) - e(G) + f(G) = 2$$

[10.3] P259 Corollary 10.20:

All planar embeddings of a connected planar graph have the same number of faces.

[10.3] P259 Corollary 10.21:

Let G be a simple planar graph on at least three vertices. Then $m \leq 3n-6$. Furthermore, $m = 3n-6$ if and only if every planar embedding of G is a triangulation.

[10.3] P259 Corollary 10.22:

Every simple planar graph has a vertex of degress at most five.

[10.3] P259 Corollary 10.23.

K_5 is nonplanar proof: If K_5 were planar, corollary 10.21 would give $10 = e(K_5) \leq 3v(K_5) - 6 = 9$

[10.3] P259 Corollary 10.24:

$K_{3,3}$ is nonplanar. proof: If $K_{3,3}$ were planar, $K_{3,3}$ has no cycle of length < 4, so every face of G has degree ≥ 4. By Theorem 10.10, $4f(G) \leq \sum_{f \in F} d(f) = 2e(G) = 18$

$\therefore \underset{\text{integer}}{f(G)} \leq 4$

10.2] P253 Theorem 10.10: If G is a plane graph

$$\sum_{f \in F} d(f) = 2m$$

Euler's formula implies that

$\underset{6}{v(G)} - \underset{9}{e(G)} + f(G) = 2$

$6 - 9 + f(G) = 2 \Rightarrow f(G) = 5$

10.5] P268 Theorem 10.30: KURATOWSKI'S THEOREM

A graph is planar iff it contains no subdivision of either K_5 or $K_{3,3}$.

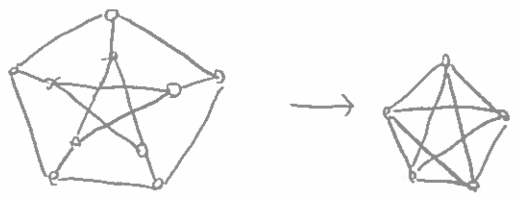
A subdivision of K_5 or $K_{3,3}$ is consequently called a Kuratowski subdivision.

10.5] P268 minor: A minor of a graph is any graph obtainable from G by means of a sequence of vertex and edge deletions and edge contractions ?

Alternatively, consider a partition $(V_0, V_1, \ldots, V_k)$ of V such that $G[V_i]$ is connected, $1 \leq i \leq k$, and let H be the graph obtained from G by deleting V_0 and shrinking each induced subgraph $G[V_i]$, $1 \leq i \leq k$, to a single vertex. The any spanning? subgraph F of H is a minor of G.

e.g. K_5 is a minor of the Petersen graph because it can be obtained by contracting the five 'spoke' edges of the latter graph.

Petersen graph



important:

1) any graph which contains an F-subdivision also has an F-minor

2) provided that F is a graph of maximum degree three or less, any graph which has an F-minor also contains an F-subdivision.

If F is a minor of G, we write $F \preceq G$.

Exc. find a $K_{3,3}$ minor from Petersen graph.

[10.5] P269 WAGNER'S THEOREM

A graph is planar iff it has no Kuratowski minor

[1.1] P_3 loop: an edge with identical ends is called a loop
link: an edge with distinct ends is called a link
parallel edges: two or more links with the same pair of ends are said to be parallel edges
simple: a graph is simple if it has no loops or parallel edges.

[D; Battista et al.]

{1} P_6 drawing: a drawing Γ of a graph (digraph) G is a function which maps each vertex v to a distinct point $\Gamma(v)$ and each edge (u,v) to a simple open Jordan curve $\Gamma(u,v)$, with endpoints $\Gamma(u)$ and $\Gamma(v)$.

{1} P_7 drawing planar: a drawing Γ is planar if no two distinct edges intersect.
A graph is planar if it admits a planar drawing

{1} P_8 connected: a graph is connected if there is a path between u and v for each pair (u,v) of vertices.
cut vertex: A cut vertex in graph G is a vertex whose removal disconnects G.
biconnected: A connected graph with no cutvertices is biconnected.
blocks: The maximal biconnected subgraphs of a graph are its blocks. (sometimes called biconnected components).
A graph is planar iff its blocks are planar.

{1} P_9 important: ?1) The skeleton of a convex polyhedron is a planar triconnected graph.
?2) A planar triconnected graph has a unique embedding, up to a reversal of the circular ordering of the neighbors of each vertex

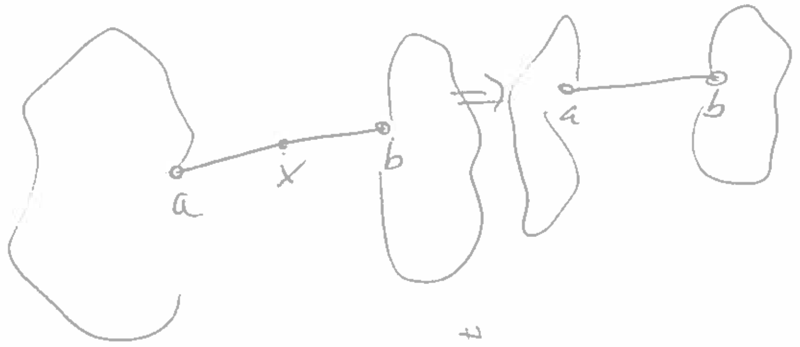
{2.1} P_{12} drawing convention: A drawing convention is a basic rule that the drawing must satisfy to be admissible.
- Polyline drawing
- Straight-line drawing
- Orthogonal drawing
- Grid Drawing
- Planar Drawing
- Upward (resp. downward) Drawing

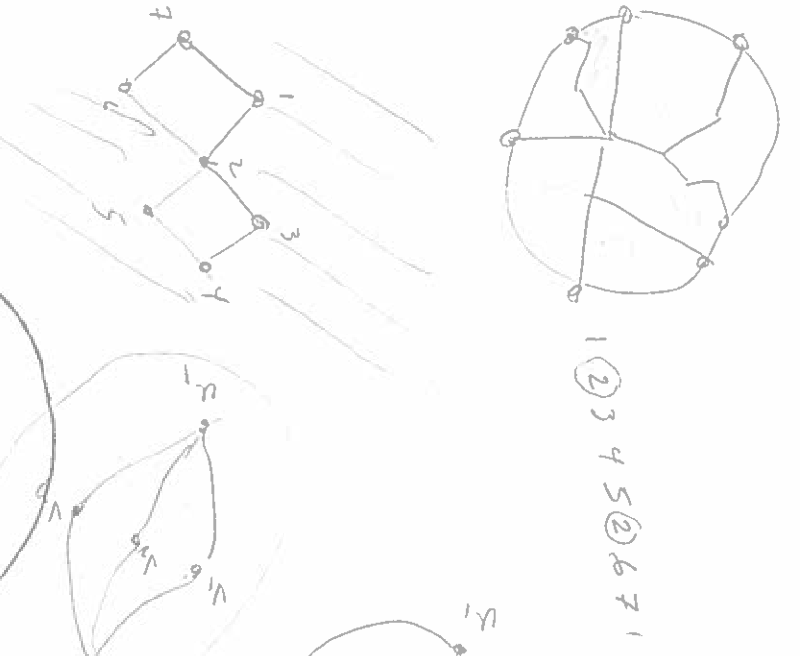
{2.1} P_{14} aesthetics: aesthetics specify graph properties of the drawing that we would like to apply as much as possible, to achieve readability.

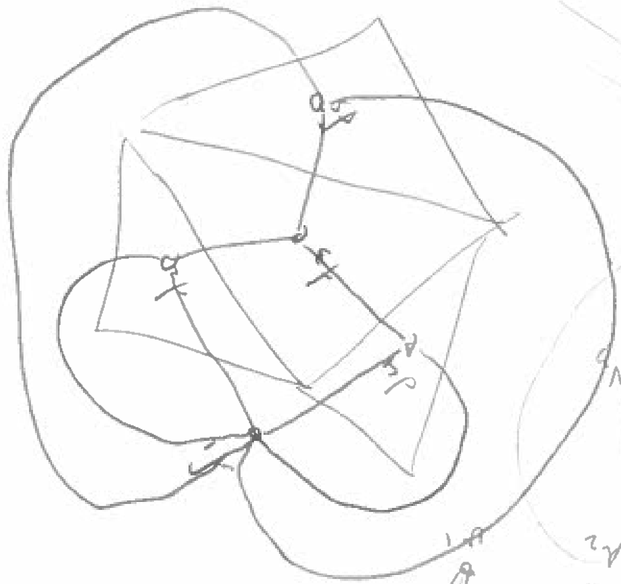
constraints:
refer to specific subgraphs or sub drawings

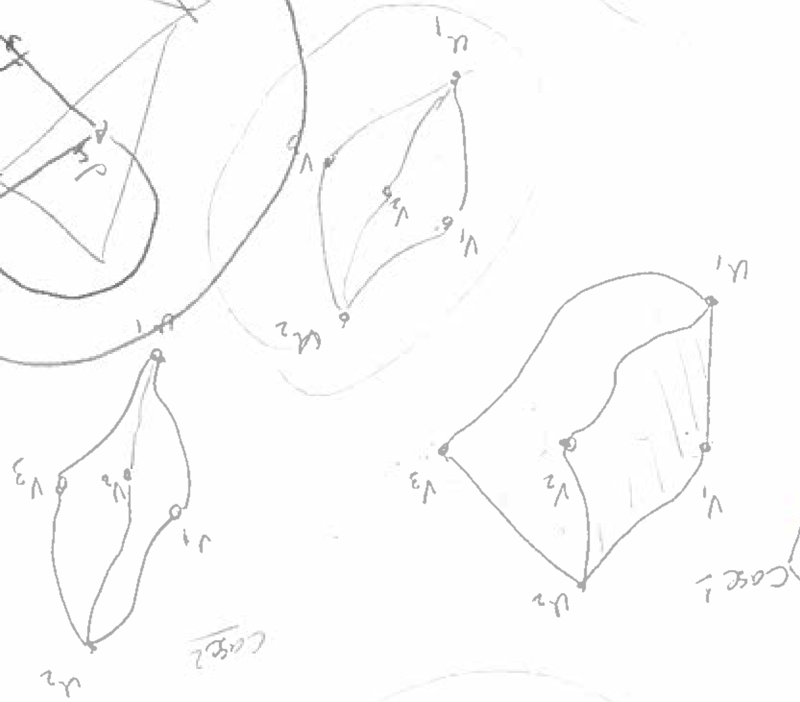
efficiency:
interactive applications

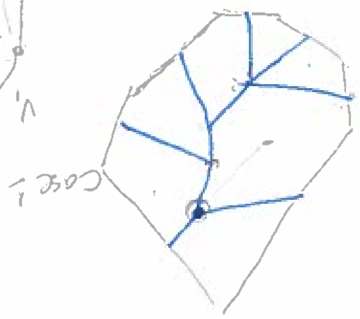
- crossings: minimize
- area: minimize (grid drawing. straight-line drawing where distance$(u,v) \geq 1$)
  convex hull
- total edge length: minimize
- maximum edge length: minimize.
- uniform edge length minimize variance of lengths of the edges
- total bends: minimize. (important for orthogonal drawings)
- maximum bends: minimize
- uniform bends: minimize
- angular resolution: maximize (straight-line drawing)
- aspect ratio: minimize longest-side ; shortest side
- symmetry

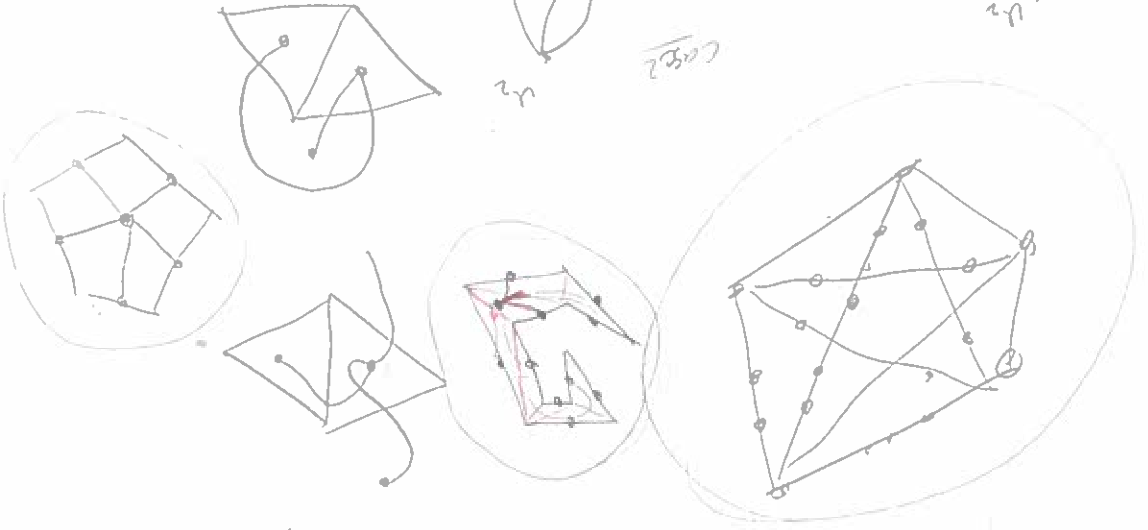

a
x
b
a
b

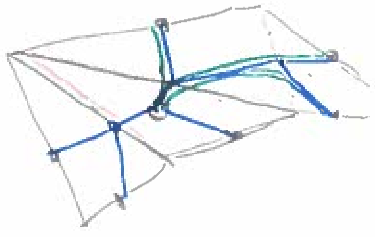

7
1
2
3
4
5
1 ② 3 4 5 ② 6 7 1

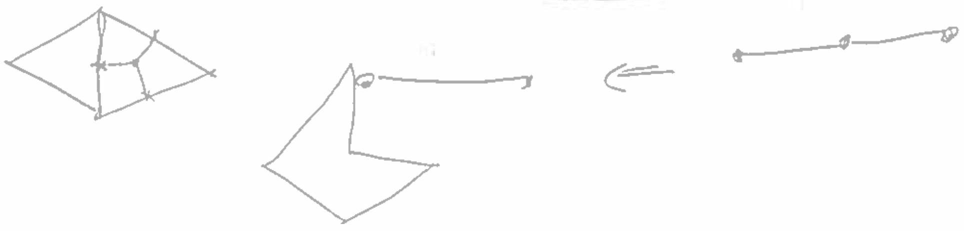

d
f
d
f
f
f


u₁
v₁
v₂
v₃
u₂
u₁
v₁
v₂
v₃
u₂
v₁
v₂
v₃
u₂
u₁
Case 2


Case 1







[The Left-Right Planarity Test] by Ulrik Brandes

- Efficient Planarity Testing by John Hopcroft & Robert Tarjan (1974)
- A Depth-first-search Characterization of Planarity by H. De Fraysseix & P. Rosenstiehl (1982)
- On the Realization of Complexes in Euclidean Spaces 吴文俊 Wen-Tsun Wu (1955)

{P4} There are only two significant ways to draw a simple cycle planarly, namely clockwise and counterclockwise.

{P4} Testing planarity amounts to deciding whether there is a consistent simultaneous orientation of all cycles.

{P5} In any planar drawing the back edges can be partitioned into left and right depending on whether their fundamental cycle is counterclockwise or clockwise.

{P6} In the oriented graph, we denote by $E^+(v) = \{(v,w) \in E : w \in V\}$ the set of all outgoing edges of $v \in V$, so that $E = \bigcup_{v \in V} E^+(v)$.

{P6} A DFS traversal yields a bipartition $E = T \uplus B$ of the edges, where those in T are called <u>tree edges</u>, and the non-tree edges in B are called <u>back edges</u>. We write $u \rightarrow v$ for $(u,v) \in T$ and $v \hookrightarrow w$ for $(v,w) \in B$.

{P6} <u>fundamental cycle</u>: $C(v \hookrightarrow w) = w \xrightarrow{+} v \hookrightarrow w$
<u>overlapping</u>. Two cycles are called overlapping, if they share an edge.

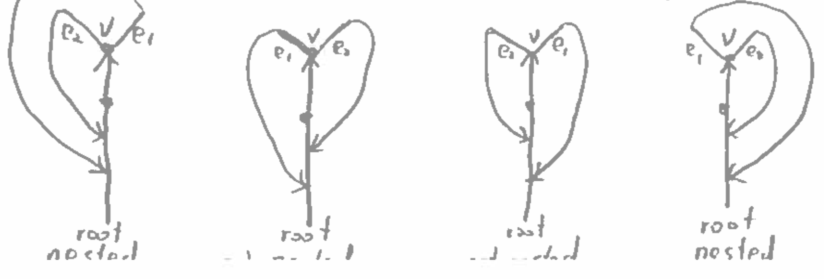
{P6} <u>Lemma 3</u> Let $G = (V, T \uplus B)$ be a DFS-oriented graph.
(1) The fundamental cycles are exactly the simple directed cycles of G.
(2) Two distinct fundamental cycles are either disjoint, or their intersection forms a tree path

{P7} <u>fork</u>: For two overlapping cycles, the last edge $u \rightarrow v$ on the shared tree path together with succeeding edges $e_1 = (v, w_1)$, $e_2 = (v, w_2)$ on each cycle is called their <u>fork</u>, and v its <u>branching point</u>

{P7} a linearization of the cyclic ordering: It is defined by splitting the clockwise order restricted to outgoing edges at the incoming tree edge, or between any two consecutive outgoing edges if v is the root of a DFS tree.

{P7} <u>nested</u>: Two overlapping fundamental cycles are called <u>nested</u>, if the part of one cycle that is not common to both is drawn completely inside the other cycle.

{P8} <u>Observation 1</u>: In a planar drawing of a DFS-oriented graph $G = (V, T \uplus B)$, two overlapping cycles are nested, if and only if they are oriented alike.



{P8} <u>return points</u>: The return points of a tree edge $v \to w \in T$ are the ancestors u of v with $u \xrightarrow{+} v \to w \xrightarrow{*} x \hookrightarrow u$ for some descendant x of w.
The return points of a vertex $v \in V$ are formed by the union of all return points of outgoing edges $(v,w) \in E^+(v) \subseteq T \uplus B$.

{P8} <u>lowpoint</u>: The lowpoint of an edge is its lowest return point, if any, or its source if none exists.

{P9} <u>Observation 2</u>: In a planar drawing of a connected DFS-oriented graph $G = (V, T \uplus B)$ with the root of the DFS tree on the outer face, overlapping fundamental cycles are nested according to their lowpoint order.

{P9} <u>left, right</u>: the side of a back edge in a planar drawing is right, if its fundamental cycle is oriented clockwise, and left otherwise.

{P9} <u>LR partition</u>: Let $G = (V, T \uplus B)$ be a DFS-oriented graph. A partition $B = L \uplus R$ of its back edges into two classes, referred to as left and right, is called left-right partition, or LR partition for short, if every fork consisting of $u \to v \in T$ and $e_1, e_2 \in E^+(v)$
(1) all return edges of e_1 ending strictly higher than $lowpt(e_2)$ belong to one class and
(2) all return edges of e_2 ending strictly higher than $lowpt(e_1)$ to the other.

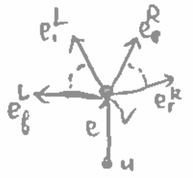
{P10} <u>Left-Right Planarity Criterion</u>: A graph is planar if and only if it admits an LR partition.

{P10} <u>aligned</u>: An LR partition is called aligned, if all return edges of a tree edge e that return to $lowpt(e)$ are on the same side.

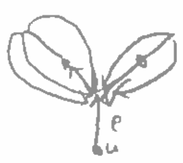
{P11} <u>Lemma 6</u>: Any LR partition can be turned into an aligned LR partition.

{P11} <u>$e_1 \prec e_2$</u>: we have to define $e_1 \prec e_2$ if and only if the lowpoint of e_1 is strictly lower than that of e_2. If both have the same lowpoint, but say, only e_2 has another return point, we say that e_2 is <u>chordal</u> and let $e_1 \prec e_2$.

{P12} <u>Definition 7 LR Ordering</u>: Given an LR partition, let $e_1^L \prec \dots \prec e_\ell^L$ be the left outgoing edges of a vertex v and $e_1^R \prec \dots \prec e_r^R$ its right outgoing edges. If v is not the root, let u be its parent. The clockwise leftright ordering, or LR ordering for short, of the edges around v is defined as follows:



(u,v),
$L(e_\ell^L), e_\ell^L, R(e_\ell^L), \dots, L(e_1^L), e_1^L, R(e_1^L)$,
$L(e_1^R), e_1^R, R(e_1^R), \dots, L(e_r^R), e_r^R, R(e_r^R)$

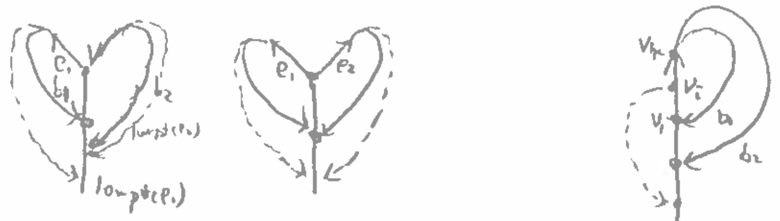


where (u,v) is absent if v is the root, and $L(e)$ and $R(e)$ denote the left and right incoming back edges whose cycles share e. For two back edges $b_1 = x_1 \hookrightarrow v$, $b_2 = x_2 \hookrightarrow v \in R(e)$ let $z \to x, (x,y_1), (x,y_2)$ be the fork of $C(b_1), C(b_2)$. Then, b_1 comes after b_2 in $R(e)$ if and only if $(x,y_1) \prec (x,y_2)$. If $b_1, b_2 \in L(e)$, the order is reversed.

{P12} Lemma 8. Given an LR partition, LR ordering yields a planar embedding.

{P14} Corollary 9. Let $G = (V, T \uplus B)$ be a DFS-oriented graph. For a pair of back edges $b_1, b_2 \in B$ with overlapping fundamental cycles let $v_1 \to \cdots \to v_k$ be the tree path of the intersection and $(v_{k-1}, v_k), e_1, e_2$ the corresponding fork with $e_1 \xrightarrow{*} b_1$ and $e_2 \xrightarrow{*} b_2$. Then, b_1 and b_2 are subject to

- a different-constraint, iff $\text{lowpoint}(e_2) < \text{lowpoint}(b_1)$ and $\text{lowpoint}(e_1) < \text{lowpoint}(b_2)$
- a same-constraint, iff $\text{lowpoint}(e') < \min\{\text{lowpoint}(b_1), \text{lowpoint}(b_2)\}$ for some $e' = (v_i, w) \in T \uplus B$, $1 < i < k$, $w \neq v_{i+1}$



(a) different-constraint

{P16} Definition 10: Let $G = (V, T \uplus B)$ be a DFS-oriented graph such that each pair of back edges $b_1, b_2 \in B$ is subject to at most one type of constraint. The signed graph $C(G) = (B, E(C); \sigma : E(C) \to \{-1, +1\})$ with

$$\sigma(b_1, b_2) = \begin{cases} -1 & \text{if } b_1, b_2 \in B \text{ are subject to a different constraint} \\ +1 & \text{if } b_1, b_2 \in B \text{ are subject to a same constraint} \end{cases}$$

is called constraint graph of G.

{P16} If any pair of back edges is subject to both a same-constraint and a different-constraint, no LR partition can exist and hence the graph is non-planar.

{P18}

```
Algorithm 1: Left-Right Planarity Algorithm
 input: simple, undirected graph $G = (V, E)$
 output: planar embedding (halts if graph is not planar)
 if $|V| > 2$ and $|E| > 3|V| - 6$ then HALT: not planar

 ▼ orientation
    for $s \in V$ do
       if $height[s] = \infty$ then
          $height[s] \leftarrow 0$; append $Roots \leftarrow s$
          DFS1(s)

 ▼ testing
    sort adjacency lists according to non-descending nesting_depth
    for $s \in Roots$ do DFS2(s)

 ▼ embedding
    for $e \in E$ do $nesting\_depth[e] = sign(e) \cdot nesting\_depth[e]$
    sort adjacency lists according to non-decreasing nesting_depth
    for $s \in Roots$ do DFS3(s)

 where
 integer sign(edge e)
    if $ref[e] \neq \perp$ then
       $side[e] \leftarrow side[e] \cdot sign(ref[e])$
       $ref[e] \leftarrow \perp$
```

HOW TO DRAW A PLANAR GRAPH ON A GRID

by H. DE FRAYSSEIX, J. PACH and R. POLLACK.

Theorem 1

The paper shows that every plane graph with n vertices has a Fáry embedding (i.e., straight-line embedding) on the $2n-4$ by $n-2$ grid and provides an $O(n)$ space, $O(n\log n)$ time algorithm to effect this embedding.

It also shows that any set F, which can support a Fáry embedding of every planar graph of size n, has cardinality at least $n + (1-o(1))\sqrt{n}$.

① Run Hopcroft–Tarjan planarity testing algorithm
outputs a topological embedding of a planar graph.

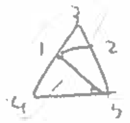
② maximal plane graph – triangulated (all faces are triangles).

Proposition 1

Given a maximal planar graph G and a face uvw, there is a labelling of the vertices, $v_1=u, v_2=v, v_3, \dots, v_n=w$ and a Fáry embedding such that the convex hull of $\{f(v_1), f(v_2), f(v_3), \dots, f(v_k)\}$ is the same as the convex hull of $\{f(v_1), f(v_2), f(v_k)\}$ for $k=4,\dots,n$

Proposition 2

Given a maximal planar graph G and a face uvw, there is a labelling of the vertices, $v_1=u, v_2=v, v_3, \dots, v_n=w$ and a Fáry embedding f such that the convex hull of $\{f(v_1), f(v_2), f(v_3), \dots, f(v_k)\}$ is the same as the convex hull of $\{f(v_{k-2}), f(v_{k-1}), f(v_k)\}$, for $k=4,\dots,n$.

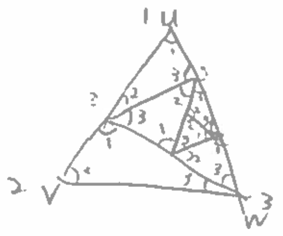


Proposition 3

Given a maximal planar graph G and a face uvw, there is a labelling of the vertices $v_1=u$, $v_2=v$, $v_3, \dots, v_n=w$ and a Fáry embedding f such that the boundary of the convex hull of $\{f(v_1), f(v_2), f(v_3), \dots, f(v_k)\}$ is a cycle in G and $f(v_{k+1})$ is not contained in the convex hull of $\{f(v_1), f(v_2), f(v_3), \dots, f(v_k)\}$.

Proposition 4 Schnyder

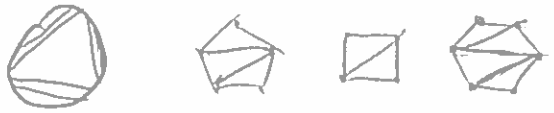
Given a maximal planar graph G with exterior face uvw, there is a labelling of the angles of the internal triangles with labels 1, 2 and 3 such that
(i) each triangle has labels 1, 2 and 3 in counterclockwise order.
(ii) all angles at u, v and w are labelled 1, 2 and 3, respectively.
(iii) around each internal vertex the angles of each label appear in a single block.



?

Theorem 2 If F is universal for planar graphs with n vertices then
$$|F| > n + (1 - o(1))\sqrt{n}$$

outerplanar A planar graph which can be obtained from a simple cycle by adding some of its internal diagonals is called outerplanar.



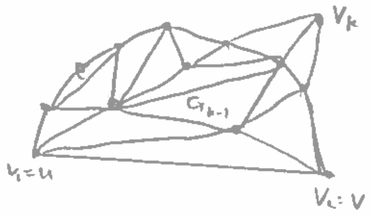
Proposition 5 Every set of n points in the plane, in general position, supports every outerplanar graph with n vertices. Moreover, this property characterizes the outerplanar graphs.

Lemma Let G be a simple planar graph embedded in the plane and $u = u_1, u_2, \dots, u_k = v$ be a cycle of G. Then there exists a vertex w' (resp. w'') on the cycle, different from u and v and not adjacent to any inside chord (resp. outside chord).

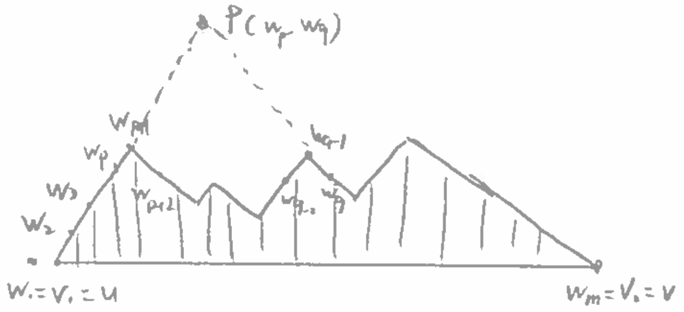
Canonical representation lemma for plane graphs
Let G be a maximal planar graph embedded in the plane with exterior face u, v, w. Then there exists a labelling of the vertices $v_1 = u, v_2 = v, v_3, \dots, v_n = w$ meeting the following requirements for every $4 \le k \le n$.

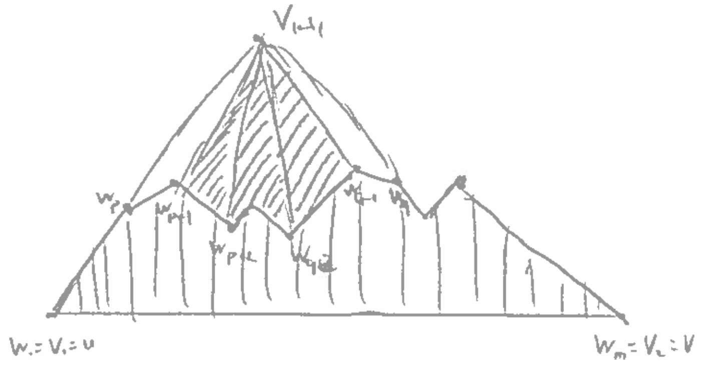
(i) The subgraph $G_{k-1} \subseteq G$ induced by $v_1, v_2, \dots, v_{k-1}$ is 2-connected, and the boundary of its exterior face is a cycle C_{k-1} containing the edge uv;

(ii) v_k is in the exterior face of G_{k-1}, and its neighbours in G_{k-1} form an (at-least 2-element) subinterval of the path $C_{k-1} - uv$.



Key idea of induction





To realize this goal, assume that for each vertex w_i on the exterior face of G_k we have already defined a subset $M(k, w_i) \subseteq V(G_k)$ such that

(a) $w_j \in M(k, w_i)$ iff $j \ge i$

(b) $M(k, w_1) \supset M(k, w_2) \supset \dots \supset M(k, w_m)$

(c) For any nonnegative numbers $\alpha_1, \alpha_2, \dots, \alpha_m$, if we sequentially translate all vertices in $M(k, w_i)$ with distance α_i to the right $(i = 1, 2, \dots, m)$, then the embedding of G_k remains a Fáry embedding.

Embedding Planar Graphs on the Grid by Walter Schnyder

This paper shows that each plane graph of order $n \geq 3$ has a straight line embedding on the $n-2$ by $n-2$ grid. This embedding is computable in time $O(n)$.

cf) I. Fáry, On straight line representation of planar graphs, Acta Sci. Math. Szeged 11 (1948), 229-233.

Characterization: planar graphs are graphs whose incidence relation is the intersection of three total orders.

THEOREM 1.1 Let $\lambda_1, \lambda_2, \lambda_3$ be three pairwise non parallel straight lines in the plane. Then each plane graph has a straight line embedding in which any two disjoint edges are separated by a straight line parallel to λ_1, λ_2, or λ_3.

THEOREM 1.2 Each plane graph with $n \geq 3$ vertices has a straight-line embedding on the $n-2$ by $n-2$ grid.

Barycentric representations

A barycentric representation of a graph G is an injective function $v \in V(G) \longrightarrow (v_1, v_2, v_3) \in \mathbb{R}^3$ that satisfies the conditions:

(1) $v_1 + v_2 + v_3 = 1$ for all vertices v

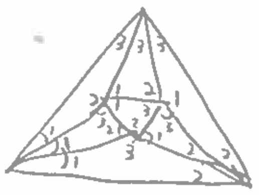
(2) For each edge $\{x, y\}$ and each vertex $z \notin \{x, y\}$, there is some $k \in \{1, 2, 3\}$ such that $x_k < z_k$ and $y_k < z_k$.

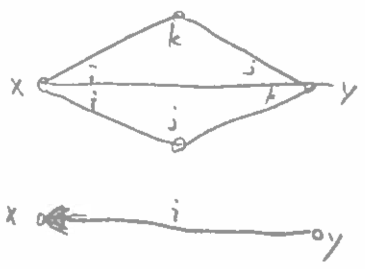
normal labeling

A normal labeling of a triangular graph G is a labeling of the angles of G with the labels 1, 2, 3 satisfying the conditions

(1) Each elementary triangle of G has an angle labeled 1, an angle labeled 2, and an angle labeled 3. The corresponding vertices appear in counterclockwise order.

(2) The labels of the angles of each interior vertex v of G form, in counterclockwise order, a nonempty interval of 1's followed by a nonempty interval of 2's by a nonempty interval of 3's.



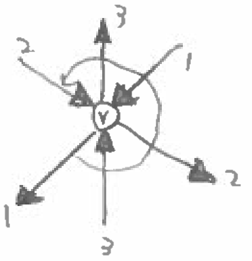


THEOREM 4.2. Each triangular graph has a normal labeling.

realizer A realizer of a triangular graph G is a partition of the interior edges of G in three sets T_1, T_2, T_3 of directed edges such that for each interior vertex v there holds:

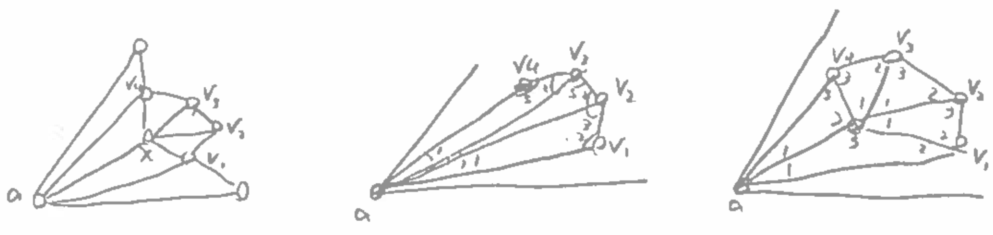
(1) v has outdegree one in each of T_1, T_2, T_3

(2) The counterclockwise order of the edges incident on v is: leaving T_1, entering T_3, leaving in T_2, entering in T_1, leaving in T_3, entering in T_2.



The edge {x,y} is contractible if x and y have exactly two common neigbours

Prove THEOREM 4.2 by induction.



THEOREM 4.5

Let G be a triangular graph with at least four vertices and let T_1, T_2, T_3 be a realizer of G. Then each T_i is a tree including all interior vertices and exactly one exterior vertex and all edges of T_i are directed toward this exterior vertex The exterior vertices belonging to T_1, T_2, T_3 are distinct and appear in counterclockwise order.

THEOREM 4.6

If T_1, T_2, T_3 is a realizer of a triangular graph, then for i=1,2,3 the relation $T_i \cup T_{i+1}^{-1} \cup T_{i+2}^{-1}$ has no directed cycle (indices are modulo 3).

THEOREM 6.1

The function $f: v \in V(G) \longrightarrow \frac{1}{2n-5}(v_1, v_2, v_3)$ is a barycentric representation of G and the labeling of G that is induced by f is identical to the given labeling of G

COROLLARY 6.2

Let a, b and c denote the roots of T_1, T_2, T_3. Then for any choice of noncolinear positions of a, b and c the mapping

$$f: v \rightarrow \frac{1}{2n-5}(v_1 a + v_2 b + v_3 c)$$

is a straight line embedding of G in the plane spanned by a, b, c.

PROPOSITION 6.3

The mapping $v \in V(G) \longrightarrow (v_1, v_2)$ is a straight line embedding of G on the 2n-5 by 2n-5 grid.

HOW TO DRAW A GRAPH By W.T. TUTTE

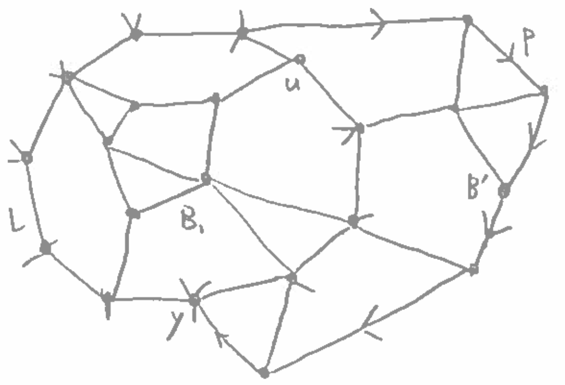
3-connected: A graph G is 3-connected (nodally 3-connected) if it is simple and non-separable and satisfies the following condition; if G is the union of two proper subgraphs H and K such that $H \cap K$ consists solely of two vertices u and v, then one of H and K is a link-graph (arc-graph) with ends u and v.

peripheral polygon: Let J be a polygon of G and let $\beta(J)$ denote the number of bridges of J in G. If $\beta(J) \leq 1$ we call J a periphral polygon of G.

(2.1) Let G be a nodally 3-connected graph, Let J be a polygon of G and B any bridge of J in G. Then either J is a peripheral or J has another bridge B' which does not avoid B.

avoid: If one of residual arc-graphs of B in J includes all the vertices of attachment of a second bridge B' of J in G we say that B' avoids B. Then B avoids B'.

(2.2) Let G be a nodally 3-connected graph. Let K_1 be a polygon of G, B_1 a bridge of K_1 in G, C a subgraph of B_1, and L a branch of G in K_1. Then we can find a peripheral polygon J of G such that $L \subset J$ and $J \cap C \subseteq K_1 \cap C$.



(2.3) Let G be a nodally 3-connected graph which is not a polygon or a link-graph, and let L be a branch of G. Then we can find two peripheral polygons J_1 and J_2 of G such that $J_1 \cap J_2 = L$.

(2.4) Let G be a nodally 3-connected graph, K a polygon of G, B a bridge of K in G, and L a branch of G contained in K. Let J_1 and J_2 be peripheral polygons of G such that $L \subseteq J_1 \cap J_2$ and neither $B \cap J_1$ nor $B \cap J_2$ is a subgraph of K. Then we can find a peripheral polygon J_3, distinct from J_1 and J_2, such that $L \subset J_3$.

(2.5) Let G be a nodally 3-connected non-null graph Then we can find a set of $p_1(G)$ independent peripheral cycles of G.

peripheral cycle : Consider the set of cycles of a connected graph G The rank of this set, the maximum number of cycles independent under mod-2 addition, is

$$p_1(G) = \alpha_1(G) - \alpha_0(G) + 1$$

We refer to the elementary cycles associated with a peripheral polygon as a peripheral cycle.

(2.6) Let G be a nodally 3-connected non-null graph. with at least two edges, which is not a polygon. Suppose that no edge of G belongs to more than two distinct peripheral polygons. Then G has just $p_1(G) + 1$ distinct peripheral cycles, and they constitute a planar mesh of G.

(2.7) A peripheral polygon K of a non-seperable graph G belongs to every planar mesh of G.

(2.8) If M is a planar mesh of a nodally 3-connected graph G, then each member of M is peripheral.

(2.7) + (2.8) show that a nodally 3-connected graph has at most one planar mesh.

representation : We call H a representation of G in $\overset{\text{(closed plane)}}{\Pi}$ if it satifies the following conditions

(i) No edge of H contains any vertex of H.

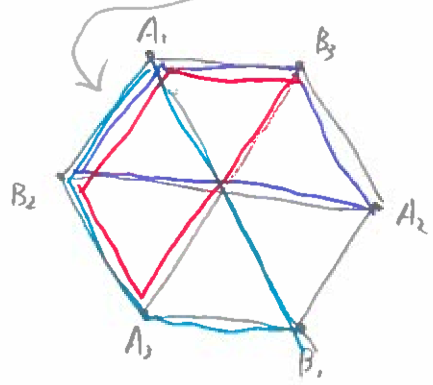
(ii) If e and e' are distinct edges of G, then f(e) and f(e') are disjoint.

A graph G is said to be planar if it has a representation in Π

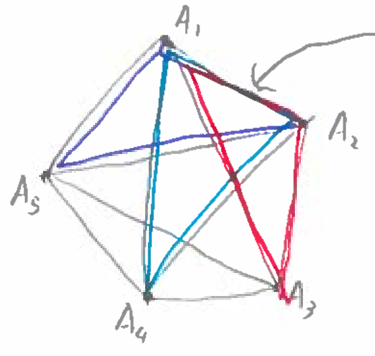
(3.1) Each peripheral polygon of H bounds a face of H.

(3.2) If a graph G has three distinct peripheral polygons with a common edge, then G is non-planar.

Kuratowski graph of Type I.



Kuratowski graph of Type II

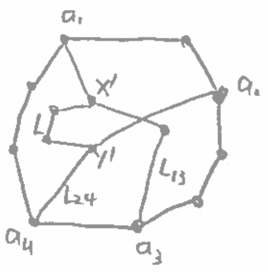


(4.1) Every Kuratowski graph is non-planar.

COROLLARY. Any graph having a Kuratowski subgraph is non-planar.

Crossing diagonals: Let J be a polygon of a graph G. Let a_1, a_2, a_3, a_4 be distinct vertices of J such that a_1 and a_3 separate a_2 from a_4 on J. Let L_{13} and L_{24} be disjoint arc-graphs of G spanning J. Then we say that L_{13} and L_{24} are crossing diagonals of J.

(5.1) Given a peripheral polygon of G with a pair of crossing diagonals we can find a Kuratowski subgraph of G of Type I.



($a_1, a_2, a_3, a_4, x', y'$ form a Kuratowski graph of Type I)

(5.2) Let J be a peripheral polygon of a graph G. Let $a, b,$ and c be distinct vertices of J. Let Y_1, Y_2 be Y-graphs of G, each with ends $a, b,$ and c, which spans J. Suppose further that $Y_1 \cap Y_2$ consists solely of the vertices $a, b,$ and c. Then we can find a Kuratowski subgraph of G.

Barycentric mappings

If $n < i \le m$ let $A(i)$ be the set of all vertices of G adjacent to v_i, that is joined to v_i by an edge. For each v_j in $A(i)$ let a unit mass m_j be placed at point $f(v_j)$. Then $f(v_i)$ is required to be the centroid of the masses m_j.

Denoting the coordinates of $f(v_i)$, $1 \le i \le m$, by (x_i, y_i).

Define a matrix $K(G) = \{C_{ij}\}$, $1 \le (i, j) \le m$, as follows.

If $i \ne j$ then C_{ij} is minus the number of edges joining v_i and v_j.

If $i = j$ then C_{ij} is the degree of v_i.

Then the foregoing requirement specifies the coordinates x_j and y_j, for $n < j \le m$, as the solutions of the equations

$$(5) \qquad \sum_{j=1}^{m} C_{ij} x_j = 0$$

$$(6) \qquad \sum_{j=1}^{m} C_{ij} y_j = 0.$$

where $n < i < m$.

$\varphi(i)$: Choose a line L in the plane and define $\varphi(i)$, $1 \leq i \leq m$, as the perpendicular distance of $f(v_i)$ from L, counted positive on one side of L and negative on the other.

φ-active: We call v_i φ-active if there is an adjacent vertex v_j of G such that $\varphi(j) \neq \varphi(i)$

positive φ-poles: The nodes v_i of J with the greatest value of $\varphi(i)$ are the positive φ-poles of G. The number of positive φ-poles is either 1 or 2.

rising (falling) φ-path: Let P be a simple path in G. We call P a rising (falling) φ-path if each vertex of P other than the last corresponds to a smaller (greater) value of the function $\varphi(i)$ than does the immediately succeeding vertex.

(6.1) Suppose that v_i, where $n < i \leq m$, is a φ-active vertex. Then it has adjacent vertices v_j and v_k such that $\varphi(v_j) < \varphi(v_i) < \varphi(v_k)$.

(6.2) Let v_i be a φ-active vertex. Then we can find a rising φ-path P from v_i to a positive φ-pole, and a falling φ-path P' from v_i to a negative φ-pole.

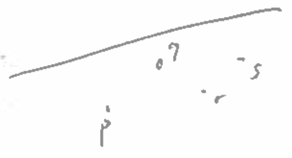
(7.1) Every node of G is φ-active.

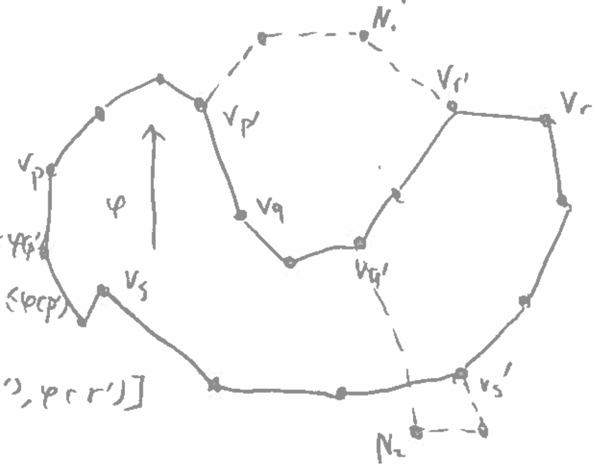
(7.2) Suppose $v_i \notin V(J)$. Then $f(v_i)$ is in the interior of Q.

(8.1) Let K be a peripheral polygon of G such that $V(K)$ includes just three nodes x, y, and z of G. Then $f(x)$, $f(y)$, and $f(z)$ are not collinear.

(8.2) Let K be a peripheral polygon of G. Let v_p, v_q, v_r, and v_s be nodes of G in $V(K)$ such that v_p and v_r separate v_q and v_s in K. Then it is not true that

(7) $$\varphi(p) \geq \varphi(q) \leq \varphi(r) \geq \varphi(s) \leq \varphi(p)$$





✓(8) $\varphi(p') \geq \varphi(q) \leq \varphi(r') \geq \varphi(s) \leq \varphi(p')$

✓(9) $\varphi(p') \geq \varphi(q') \leq \varphi(r') \geq \varphi(s') \leq \varphi(p')$

$\varphi(j) < \max[\varphi(q'), \varphi(s')] \leq \min[\varphi(p'), \varphi(r')]$

(8.3) The nodes of any peripheral polygon K of G are mapped by f onto distinct points of the plane, no three of which are collinear.

(8.4) Let L be a branch of G having just $t \geq 1$ internal vertices, and let its ends be a and b. Then $f(a)$ and $f(b)$ are distinct and f maps the internal vertices onto t distinct points of the segment $f(a)f(b)$ subdividing it into $t+1$ equal parts. Moreover, the order of the vertices from a to b in L agrees with that of their images in $f(a)f(b)$

(8.5) Let K be any peripheral polygon of G. Then f maps the nodes of G in K onto the vertices of a (geometrical) convex polygon Q_K so that the cyclic order of nodes in K agrees with that of vertices in Q_K.

(8.6) Let e be any edge of R. Then just two distinct peripheral polygons of G pass through e, and the two corresponding regions R_K lie on opposite sides of the segment $f(e)$.

Barycentric representations.

(9.1) $\underline{\delta(A)} = 1$ for each A in $\underline{S}$.

$\uparrow$ point

$\hookrightarrow$ $|H|$'s complentary set in the plane.

/ number of distinct peripheral polygons K of G such that $A \in R_K$.

(9.2) Let G be a nodally 3-connected graph having no Kuratowski subgraph. Let J be a peripheral polygon of G which includes just $n \geq 3$ nodes of G. Let Q be an n-sided convex polygon in the Euclidean plane. Then there is a unique barycentric representation of G on Q mapping the nodes of G occuring on J onto the vertices of Q in any arbitrary specified way preserving the cyclic order.

(9.3) Let G be a nodally 3-connected graph having at least one polygon. Then if G has no Kuratowski subgraph we can construct a convex representation of G.

Straight representations

(10.1) Let G be the union of two proper subgraphs H and K such that $H \cap K$ is either null or a vertex-graph. Let M_H and M_K be planar meshes of H and K respectively. Then $M_H \cup M_K$ is a planar mesh of G. Moreover, any planar mesh of G can be represented in this form.

(10.6) Let G be a graph having a planar mesh M. Then each subgraph of G has a planar mesh.

(10.7) If a graph has a planar mesh it has no Kuratowski subgraph.

(10.8) Let G be any simple graph having a planar mesh. Then by adding new links to G with ends in $V(G)$, we can construct a nodally 3-connected graph T having a planar mesh.

(10.9) If G is a simple graph having a planar mesh we can find a straight representation of G in this plan

{MIT 6.889} by Erik Demaine, Shay Mozes, Christian Sommer, Siamak Tazari

"Solve your favourite problems faster for graphs that matter!"

Survey:

Problems	general	vs. planar
- single-source shortest paths (arbitrary weights)	$O(nm)$ [Bellman-Ford]	$O(n \frac{\lg^2 n}{\lg\lg n})$ [Mozes & Wulff-Nilsen - ESA 2010]
- nonnegative weights	$O(n \lg n + m)$ [Dijkstra] + [Fredman & Tarjan] - JACM 1987	$O(n)$ [Henzinger, Klein, Rao, Subramanian - JCSS 1997]
- maximum flow	$O(nm \lg n)$ [Goldberg & Tarjan 1986] $O(m^{3/2} \lg n \lg u)$ [Goldberg & Rao 1997]	$O(n \lg n)$ [Borradaile & Klein - JACM 2009]
- undirected		$O(n \lg\lg n)$ [Italiano, Nussbaum, Sankowski, Wulff-Nilsen - STOC 2011]
- multiterminal		$O(n \lg^3 n)$ [Borradaile, Klein, Mozes, Nussbaum, Wulff-Nilsen - FOCS 2011]
- min spanning tree	$O(n)$ rand. [Karger, Klein, Tarjan 1985]	$O(n)$ det

« Algorithm Design » by Jon Kleinberg and Éva Tardos

§ Chapter 13 : Randomized Algorithms

When one thinks about random process, it is usually in one of two distinct ways.

1) average - case analysis (randomly generated input)

2) consider algorithms that behave randomly $\longrightarrow$ randomized algorithm

13.1 A First Application: Contention Resolution

Suppose n processes $P_1, P_2, \dots, P_n$, each competing for access to a single shared database. time as being divided into discrete rounds. Database can be accessed by at most one process in a single round. Processes can't communicate with one another at all.

Algorithm

each process will attempt to access the database in each round with probability p, independently of the decisions of the other processes.

Analyzing the Algorithm

let $\mathcal{A}[i,t]$ denote the event that P_i attempts to access database in round t. $Pr[\mathcal{A}[i,t]] = p$

let $\mathcal{S}[i,t]$ denote the event that P_i succeeds in accessing the database in round t. $Pr[\overline{\mathcal{A}[i,t]}] = 1-p$

$$Pr[\mathcal{S}[i,t]] = Pr[\mathcal{A}[i,t]] \cdot \prod_{j\neq i} Pr[\overline{\mathcal{A}[j,t]}] = p(1-p)^{n-1}$$

$$f(p) = p(1-p)^{n-1}$$

$$f'(p) = (1-p)^{n-1} - p\cdot(n-1)(1-p)^{n-2}$$

$$(1-p)^{n-1} - (n-1)\cdot p(1-p)^{n-2} \geq 0$$
$$(1-p) \geq (n-1)p$$
$$\frac{1}{p} - 1 \geq n-1$$
$$p \leq \frac{1}{n}$$

we set $p = \frac{1}{n}$, $\quad Pr[\mathcal{S}[i,t]] = \frac{1}{n}(1-\frac{1}{n})^{n-1}$

(13.1)
(a) The function $(1-\frac{1}{n})^n$ converges monotonically from $\frac{1}{4}$ up to $\frac{1}{e}$ as n increases from 2.
(b) The function $(1-\frac{1}{n})^{n-1}$ converges monotonically from $\frac{1}{2}$ down to $\frac{1}{e}$ as n increases from 2.

$$\therefore \quad \frac{1}{en} \leq \underbrace{Pr[\mathcal{S}[i,t]]}_{\Theta(\frac{1}{n})} \leq \frac{1}{2n}$$

let $\mathcal{F}[i,t]$ denote the "failure event" that process P_i does not succeed in any of the rounds 1 through t.

$$Pr[\mathcal{F}[i,t]] = Pr[\bigcap_{r=1}^{t} \overline{\mathcal{S}[i,r]}] = \left[1 - \frac{1}{n}(1-\frac{1}{n})^{n-1}\right]^t \leq (1-\frac{1}{en})^t \underset{\text{set } t = \lceil en \rceil}{\leq} (1-\frac{1}{en})^{en} \leq \frac{1}{e}$$

$t = \lceil en \rceil \cdot c \ln n$

$$Pr[F[i,t]] \le \left(1 - \frac{1}{en}\right)^t = \left(\left(1 - \frac{1}{en}\right)^{\lceil en \rceil}\right)^{c \ln n} \le e^{-c \ln n} = n^{-c}$$

Conclusion: After $\Theta(n)$ rounds ($t = \lceil en \rceil$), the probability that P_i has not succeeded is bounded by a constant ($\frac{1}{e}$); and between then and $\Theta(n \ln n)$, this probability drops to a quantity that is quite small, bounded by an inverse polynomial in n.

Waiting for All Processes to Get Though

$$\mathcal{F}_t = \bigcup_{i=1}^{n} F[i,t]$$

(13.2) (The Union Bound) Given events $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$, we have

$$Pr\left[\bigcup_{i=1}^{n} \mathcal{E}_i\right] \le \sum_{i=1}^{n} Pr[\mathcal{E}_i].$$

$$\therefore \quad Pr[\mathcal{F}_t] \le \sum_{i=1}^{n} Pr[F[i,t]]$$

choose $t = \lceil en \rceil (c \ln n)$, $Pr[F[i,t]] \le n^{-c}$

choose $t = 2\lceil en \rceil \ln n$

$$Pr[\mathcal{F}_t] \le \sum_{i=1}^{n} Pr[F[i,t]] \le n \cdot n^{-2} = n^{-1}$$

(13.3) With probability at least $1 - n^{-1}$, all processes succeed in accessing the database at least once within $t = 2\lceil en \rceil \ln n$ rounds.

13.2 Finding the Global Minimum Cut

undirected graph $G = (V, E)$, define a cut of G to be a partition of V into two nonempty sets A and B. For a cut (A, B), the size of (A, B) is the number of edges with one end in A and the other in B. A global minimum cut is a cut of minimum size.

(13.4) There is a polynomial-time algorithm to find a global min-cut in an undirected graph G.
convert to directed graph
fix s. for every $t \in V - \{s\}$, run push-relabel.
the best among these will be a global min-cut of G.

David Karger 1992.

Algorithm. (Contraction Algorithm)
└ works with connected multigraph.

The Contraction Algorithm applied to a multigraph $G = (V, E)$:

For each node v, we will record the set $S(v)$ of the nodes that have been contracted into v.

```
        Initially $S(v) = \{v\}$ for each $v$

    If $G$ has two nodes $v_1$ and $v_2$, then return the cut $(S(v_1), S(v_2))$.
    Else choose an edge $e = (u, v)$ of $G$ uniformly at random
        Let $G'$ be the graph resulting from the contraction of $e$,
            with a new node $z_{uv}$ replacing $u$ and $v$.
        Define $S(z_{uv}) = S(u) \cup S(v)$
        Apply the Contraction Algorithm recursively to $G'$
    End if
```

<u>Analyzing the Algorithm</u>

(13.5) The Contraction Algorithm returns a global min-cut of G with probability at least $\frac{1}{\binom{n}{2}}$

Suppose the global min-cut has size k, a set F of k edges with one end in A and the other in B.

$\hookrightarrow$ every node in G has degree at least k.

We want an upper bound on the probability that an edge in F is contracted, and for this we need a lower bound on the size of E.

$$|E| \geq \frac{1}{2} kn$$

Hence the probability than an edge in F is contructed is at most

$$\frac{k}{\frac{1}{2}kn} = \frac{2}{n}.$$

Consider the situation after j iterations, there are $n-j$ supernodes in G'.

Thus G' has at least $\frac{1}{2}k(n-j)$ edges. So the probability that an edge of F is contracted in the next iteration $j+1$ is at most.

$$\frac{k}{\frac{1}{2}k(n-j)} = \frac{2}{n-j}$$

We write $\mathcal{E}_j$ for the event that an edge of F is not contracted in iteration j, then we have shown $\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$ and $\Pr[\mathcal{E}_{j+1} \mid \mathcal{E}_1 \cap \mathcal{E}_2 \cap \cdots \cap \mathcal{E}_j] \geq 1 - \frac{2}{n-j}$

We are interested in lower-bounding the quality $\Pr[\mathcal{E}_1 \cap \mathcal{E}_2 \cdots \cap \mathcal{E}_{n-2}]$.

$$\Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 | \mathcal{E}_1] \cdots \Pr[\mathcal{E}_{j+1} | \mathcal{E}_1 \cap \mathcal{E}_2 \cdots \cap \mathcal{E}_j] \cdots \Pr[\mathcal{E}_{n-2} | \mathcal{E}_1 \cap \mathcal{E}_2 \cdots \cap \mathcal{E}_{n-3}]$$

$$\geq \left(1 - \frac{2}{n}\right) \cdot \left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{n-j}\right) \cdots \left(1 - \frac{2}{3}\right)$$

$$= \left(\frac{n-2}{n}\right) \cdot \left(\frac{n-3}{n-1}\right) \left(\frac{n-4}{n-2}\right) \cdots \left(\frac{3}{5}\right)\left(\frac{2}{4}\right)\left(\frac{1}{3}\right)$$

$$= \frac{2}{n(n-1)} = \binom{n}{2}^{-1}$$

So we know that a single run of the Contraction Algorithm fails to find a global min-cut with probability at most $(1 - 1/\binom{n}{2})$.

If we run the algorithm $\binom{n}{2}$ times,

$$(1 - 1/\binom{n}{2})^{\binom{n}{2}} \leq \frac{1}{e}$$

If we run the algorithm $\binom{n}{2} \ln n$ times

$$\left[(1 - 1/\binom{n}{2})^{\binom{n}{2}}\right]^{\ln n} \leq e^{-\ln n} = n^{-1}$$

Further Analysis: The Number of Global Minimum Cuts

Given an undirected graph $G = (V, E)$, what is the maximum number of global min-cuts it can have (as a function of n)?

undirected graph has $\binom{n}{2}$ global min-cuts.

(13.6) An undirected graph $G = (V, E)$ on n nodes has at most $\binom{n}{2}$ global min-cuts.

Let G be a graph, and let $C_1, \ldots, C_r$ denote all its global min-cuts.
Let $\mathcal{E}_i$ denote the event that C_i is returned by the Contraction Algorithm,
let $\mathcal{E} = \bigcup_{i=1}^{r} \mathcal{E}_i$ denote the event that the algorithm returns any global min-cut.

$$Pr[\mathcal{E}] = Pr[\bigcup_{i=1}^{r} \mathcal{E}_i] = \sum_{i=1}^{r} Pr[\mathcal{E}_i] \geq r / \binom{n}{2}$$

$$Pr[\mathcal{E}] \leq 1 \Rightarrow r \leq \binom{n}{2}$$

13.3 Random Variables and Their Expectations

random variable: Given a probability space, a random variable X is a function from the underlying sample space to the natural numbers, such that for each natural number j, the set $X^{-1}(j)$ of all sample points taking value j is an event.
$Pr[X = j]$ as loose shorthand for $Pr[X^{-1}(j)]$.

expectation: the "average value" assumed by X.

$$E[X] = \sum_{j=0}^{\infty} j \cdot Pr[X = j].$$

$$S = \sum_{j=1}^{\infty} j (1-p)^j = 1 \cdot (1-p)^1 + 2(1-p)^2 + 3(1-p)^3 + \cdots$$

$$(1-p)S = 1(1-p)^2 + 2(1-p)^3 + 3(1-p)^4 + \cdots$$

$$S - (1-p)S = (1-p)^1 + (1-p)^2 + (1-p)^3 + \cdots = \frac{(1-p)}{p}$$

$$pS = \frac{(1-p)}{p} \Rightarrow S = \frac{(1-p)}{p^2}$$

$$E[X] = \sum_{j=0}^{\infty} j \cdot Pr[X = j] = \sum_{j=1}^{\infty} j (1-p)^{j-1} p = \frac{p}{1-p} \sum_{j=1}^{\infty} j (1-p)^j$$

$$= \frac{p}{1-p} \cdot \frac{(1-p)}{p^2} = \frac{1}{p}$$

(13.8) Linearity of Expectation. Given two random variables X and Y defined over the same probability space, we can define $X+Y$ to be the random variable equal to $X(\omega) + Y(\omega)$ on a sample point ω. For any X and Y, we have

$$E[X+Y] = E[X] + E[Y].$$

(13.10) $H(n) = \sum_{i=1}^{n} \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ harmonic number $H(n)$.

$\int_1^{n+1} \frac{1}{x} dx = \ln(n+1)$

$\ln(n+1) < H(n) < 1 + \ln n$, and more loosely $H(n) = \Theta(\log n)$.

Conditional Expectation

Suppose we have a random variable X and an event $\mathcal{E}$ of positive probability. Then we define the conditional expectation of X, given $\mathcal{E}$, to be the expected value of X computed only over the part of sample space corresponding to $\mathcal{E}$.

$$E[X \mid \mathcal{E}] = \sum_{j=0}^{\infty} j \cdot Pr[X = j \mid \mathcal{E}].$$

13.4 A Randomized Approximation Algorithm for MAX 3-SAT

Algorithm: set each variable $x_1, \dots, x_n$ independently to 0 or 1 with probability $\frac{1}{2}$ each.

let $Z_i = 1$ if clause C_i is satisfied, and 0 otherwise.

Thus $Z = Z_1 + Z_2 + \cdots + Z_k$. $E[Z_i]$ is equal to the probability that C_i is satisfied.

In order for C_i not to be satisfied, each of its three variables must be assigned the value that fails to make it true; $(\frac{1}{2})^3 = \frac{1}{8}$ so $E[Z_i] = \frac{7}{8}$.

$$E[Z] = E[Z_1] + E[Z_2] + \cdots + E[Z_k] = \frac{7}{8}k.$$

(13.14) Consider a 3-SAT formula, where each clause has three different variables. The expected number of clauses satisfied by a random assignment is within an approximation factor $\frac{7}{8}$ of optimal.

(13.15) For every instance of 3-SAT, there is a truth assignment that satisfies at least a $\frac{7}{8}$ fraction of all clauses.

Cute application: Every instance of 3-SAT with at most seven clauses is satisfiable.

(13.16) There is a randomized algorithm with polynomial expected running time that is guaranteed to produce a truth assignment satisfying at least $\frac{7}{8}$ fraction of all clauses.

$P \geq \frac{1}{8k}$

$n \leq 8k$

Fibonacci Heaps [Introduction to Algorithms by CLRS]

dual purpose:

1) supports a set of operations that constitutes a "mergeable heap".
2) several operations run in constant amortized time.

mergeable heap:

- MAKE-HEAP()
- INSERT(H, x) inserts element x, whose key has already been filled in, into heap H.
- MINIMUM(H) returns a pointer to the element in heap H whose key is minimum.
- EXTRACT-MIN(H)
- UNION(H_1, H_2)

Fibonacci heaps also support:

+ DECREASE-KEY(H, x, k) assigns to element x within H the new key value k, which we assume to be no greater than its current key value.
+ DELETE(H, x)

Procedure	Binary heap (worst-case)	Fibonacci heap (amortized)
MAKE-HEAP	$\Theta(1)$	$\Theta(1)$
INSERT	$\Theta(\lg n)$	$\Theta(1)$
MINIMUM	$\Theta(1)$	$\Theta(1)$
EXTRACT-MIN	$\Theta(\lg n)$	$O(\lg n)$
UNION	$\Theta(n)$	$\Theta(1)$
DECREASE-KEY	$\Theta(\lg n)$	$\Theta(1)$
DELETE	$\Theta(\lg n)$	$O(\lg n)$

desirable when number of these operations are small

application:

1) counting minimum spanning trees
2) single-source shortest paths.

drawbacks:

- large constant factors
- programming complexity

Fibonacci heap:

A Fibonacci heap is a collection of rooted trees that are min-heap ordered.

min-heap property:

the key of a node is greater than or equal to the key of its parent

Potential function

$t(H)$ the number of trees in the root list
$m(H)$ the number of marked nodes in H.

$\Phi(H) = t(H) + 2m(H)$

- assume that a unit of potential can pay for a constant amount of work

Maximum degree

assume that we know an upper bound $D(n)$ on the maximum degree of any node in an n-node Fibonacci heap.

$D(n) \leq \lfloor \lg n \rfloor$

when we support DECREASE-KEY and DELETE, $D(n) = O(\lg n)$.

Inserting a node

- just add it to the root list
- the increase in potential $t(H') = t(H)+1$, $m(H') = m(H)$ is 1,
  actual cost $O(1)$, amortized cost $O(1)+1 = O(1)$

Uniting two Fibonacci heaps

- change in potential

$\Phi(H) - (\Phi(H_1) + \Phi(H_2))$
$= (t(H) + 2m(H)) - ((t(H_1) + 2m(H_1)) + (t(H_2) + 2m(H_2)))$
$= 0$

The amortized cost of FIB-HEAP-UNION is equal to its $O(1)$ actual cost.

Extracting the minimum node

```
FIB-HEAP-EXTRACT-MIN(H)
    $z = H.min$
    if $z \neq NIL$
        for each child $x$ of $z$
            add $x$ to the root list of $H$
            $x.p = NIL$
        remove $z$ from the root list of $H$
        if $z == z.right$
            $H.min = NIL$
        else $H.min = z.right$    ⟵ not necessarily going to be the new minimum node.
            CONSOLIDATE(H)
        $H.n = H.n - 1$
    return z
```

```
CONSOLIDATE(H)                    need to know upper bound.
    let A[0 .. D(H.n)] be a new array  // keep track of roots according to their degrees
    for i = 0 to D(H.n)
        A[i] = NIL
    for each node w in the root list of H
        x = w
        d = x.degree
        while A[d] ≠ NIL
            y = A[d]          // another node with the same degree as x
            if x.key > y.key
                exchange x with y
            FIB-HEAP-LINK(H, y, x)
            A[d] = NIL
            d = d+1
        A[d] = x
    H.min = NIL
    for i = 0 to D(H.n)
        if A[i] ≠ NIL
            if H.min == NIL
                create a root list of H containing just A[i]
                H.min = A[i]
            else insert A[i] into H's root list
                if A[i].key < H.min.key
                    H.min = A[i]
```

loop invariant: At the start of each iteration of the while loop $d = x.degree$

② the total amount of work in the for loop is at most proportional to $D(n) + t(H)$

```
FIB-HEAP-LINK(H, y, x)
1. remove y from the root list of H
2. make y a child of x, incrementing x.degree
3. y.mark = FALSE
```

amortized cost : try to show it is $O(D(n))$.

① $O(D(n))$ contribution comes from FIB-HEAP-EXTRACT-MIN processing at most $D(n)$ children of the minimum node,

Thus, the total actual work is ①② in $O(D(n) + t(H))$.

potential before extracting minimum $t(H) + 2m(H)$

potential afterwards is at most $(D(n)+1) + 2m(H)$

can scale up the units of potential to dominate the constant.

$O(D(n) + t(H)) + ((D(n)+1) + 2m(H)) - (t(H) + 2m(H)) = O(D(n)) + \underline{O(t(H)) - t(H)} = D(D(n))$

Bounding the maximum degree

to show the upper bound of $D(n)$ is $O(\lg n)$.

In particular, $D(n) \le \lfloor \log_\phi n \rfloor$

$\phi = \frac{1+\sqrt{5}}{2}$

<u>Lemma 19.1</u>

Let x be any node in a Fibonacci heap, and suppose that $x.degree = k$. Let $y_1, y_2, \dots, y_k$ denote the children of x in the order in which they were linked to x from the earliest to the latest. Then $y_1.degree \ge 0$ and $y_i.degree \ge i-2$ for $i = 2, 3, \dots k$.

<u>Lemma 19.2</u>

For all integers $k \ge 0$,

$$F_{k+2} = 1 + \sum_{i=0}^{k} F_i \qquad F_k = \begin{cases} 0 & , k=0 \\ 1 & , k=1 \\ F_{k-1} + F_{k-2} & , k \ge 2 \end{cases}$$

<u>Lemma 19.3</u>

For all integers $k \ge 0$, the $(k+2)$nd Fibonacci number satisfies $F_{k+2} \ge \phi^k$

inductive step:
$$\begin{aligned} F_{k+2} &= F_{k+1} + F_k \\ &\ge \phi^{k-1} + \phi^{k-2} \quad \text{(by the inductive hypothesis)} \\ &= \phi^{k-2}(\phi + 1) \\ &= \phi^{k-2} \cdot \phi^2 \quad (\phi \text{ is the positive root of equation } x^2 = x+1) \\ &= \phi^k \end{aligned}$$

<u>Lemma 19.4</u>

Let x be any node in a Fibonacci heap, and let $k = x.degree$. Then $size(x) \ge F_{k+2} \ge \phi^k$, where $\phi = \frac{1+\sqrt{5}}{2}$

Proof. let s_k denote minimum possible size of any node of degree k in any Fibonacci heap.

$s_k \le size(x)$

$$\begin{aligned} size(x) &\ge s_k \\ &\ge 2 + \sum_{i=2}^{k} s_{y_i.degree} \\ &\ge 2 + \sum_{i=2}^{k} s_{i-2} \\ &\ge 2 + \sum_{i=2}^{k} F_i \\ &= 1 + \sum_{i=0}^{k} F_i \\ &= F_{k+2} \quad \text{(by Lemma 19.2)} \\ &\ge \phi^k \quad \text{(by Lemma 19.3)} \end{aligned}$$

$\therefore size(x) \ge s_k \ge F_{k+2} \ge \phi^k$.

<u>Corollary 19.5</u>

The maximum degree $D(n)$ of any node in an n-node Fibonacci heap is $O(\lg n)$

$n \ge size(x) \ge \phi^k$ where $k = x.degree$

$k \le \lfloor \log_\phi n \rfloor$

Maximum Flows and Parametric Shortest Paths in Planar Graphs

by Jeff Erickson jeffe@cs.uiuc.edu

let $G=(V,E)$ be a directed plane graph. s,t be vertices of G
let $c: E \longrightarrow \mathbb{R}$ be a nonnegative capacity function

Goal: compute a maximum (s,t)-flow in G.

Assume WLOG the reversal of any directed edge in G is also an edge in G
$\longrightarrow$ implies that both G and its dual G^* are strongly connected.

Venkatesan's Reduction

Idea: compute a feasible (s,t)-flow with fixed value λ, or correctly report that no such flow exists, by reduction to a single source shortest path problem in an appropriately weighted dual graph G^*

$\pi(e)$: Fixed an arbitrary directed path P from s to t, and let $\pi: E \to \mathbb{R}$ denote the unit flow through P:

$$\pi(e) := \begin{cases} 1 & \text{if } e \in P \\ -1 & \text{if } rev(e) \in P \\ 0 & \text{otherwise} \end{cases}$$

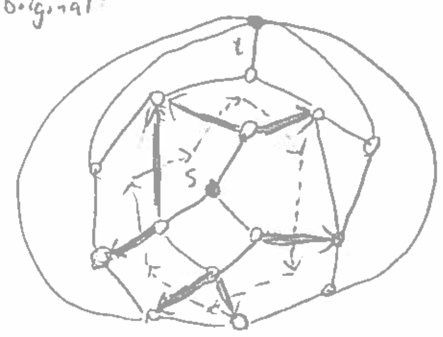
For any subset $E' \subseteq E$, let $\pi(E') = \sum_{e \in E'} \pi(e)$.

cocycle: A subgraph C of G is called a cocycle if the corresponding dual subgraph C^* is a simple directed cycle in G^*.

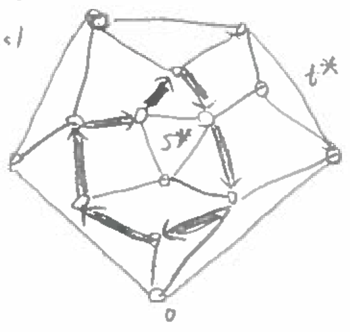
crossing number: For any cycle C^* in G^*, we call $\pi(C)$ the crossing number of C^*.

Lemma 2.1. $\pi(C) \in \{-1, 0, 1\}$ for any cocycle C. Moreover, $\pi(C) = 1$ if and only if C is an (s,t)-cut

original:



dual



consider the flow $\lambda \cdot \pi$, which assigns value $\begin{cases} \lambda & \text{to every directed edge in path } P \\ -\lambda & \text{" " " " rev}(P) \\ 0 & \text{to every other edge.} \end{cases}$

Let $G_\lambda := G_{\lambda \cdot \pi}$ denote the residual network of this flow

capacity function: $c(\lambda, e) := c(e) - \lambda \cdot \pi(e)$.

$\therefore$ $\lambda \cdot \pi$ is feasible if and only if $c(\lambda, e) \geq 0$ for every edge e in G.

Let G_λ^* denote the dual residual network, which is the directed dual graph G^* where every edge e^* has a cost $c(\lambda, e^*) = \underline{c(\lambda, e)}$.

Lemma 2.2 There is a feasible (s,t)-flow in G with value λ if and only if the residual network G_λ^* does not contain a negative cycle.

let $dist(\lambda, p)$ denote the shortest path distance in G_λ^* from o to p. (arbitrary dual vertex called origin)

define $\quad \phi(\lambda, e) := dist(\lambda, head(e^*)) - dist(\lambda, tail(e^*)) + \lambda \cdot \pi(e)$.

because the duals of the edges leaving v define a directed cycle in G^*, all the $dist(\lambda, \cdot)$ terms in the sum cancel out.

$\therefore$ $\phi(\lambda, \cdot)$ is a valid (s,t)-flow with value λ.

define the slack of each dual edge e^*:

$slack(\lambda, e^*) := dist(\lambda, tail(e^*)) + c(\lambda, e) - dist(\lambda, head(e^*))$

so $slack(\lambda, e^*) = c(e) - \phi(\lambda, e)$

Parametric Shortest Paths

Let λ_{max} denote the largest value of λ for which shortest paths in G_λ^* are well-defined. Lemma 2.2 implies that λ_{max} is also the value of the maximum flow.

For any particular value of λ, let $\underline{T_\lambda}$ denote the single-source shortest path tree in G_λ^* rooted at o.

High level algorithm

```
PLANARMAXFLOW(G, c, s, t):
Compute $T_0$
Maintain $T_\lambda$ as $\lambda$ increases continuously from 0 to $\lambda_{max}$
Compute $\phi(\lambda_{max}, \cdot)$ from $T_{\lambda_{max}}$
```

Genericity assumption

we assume that the capacity function is generic.

1) Our genericity assumption implies that T_λ is uniquely defined for all λ between 0 and λ_{max}, except for a finite set of critical values

2) Our genericity assumption implies that exactly one non-tree edge becomes tense at each critical value of λ.

Lemma 2.3 λ_{max} is the first critical value of λ whose pivot introduces a directed cycle into T_λ.

pivot At each critical value of λ, some non-tree dual edge $p \to q$ becomes tense and enters T_λ, replacing the previous edge $pred(\lambda, q) \to q$.

tense call a dual edge e^* tense if $slack(\lambda, e^*) = 0$.

Lemma 2.4 λ_{max} is the smallest critical value of λ whose pivot disconnects L_λ

loose call a primal edge e loose at λ if neither its dual e^* nor its reversed dual $rev(e^*)$ is tense at λ, and let L_λ be the graph of all loose edges.

active A dual edge is active at λ if its slack at λ is decreasing

LP_λ The primal spanning tree L_λ contains a unique directed path from s to t; call this loose path LP_λ.

Lemma 2.5 A dual edge e^* is active at λ if and only if e is an edge of LP_λ.

☆ PLANAR MAX FLOW (G, c, s, t):

```
Initialize the spanning tree $L$, predecessors and slacks
while $s$ and $t$ are in the same component of $L$
        $LP \leftarrow$ the path from $s$ to $t$
        $p \to q \leftarrow$ the edge in $LP^*$ with minimum slack
         $\Delta \leftarrow slack(p \to q)$
        for every edge $e$ in $LP$
              $slack(e^*) \leftarrow slack(e^*) - \Delta$
              $slack(rev(e^*)) \leftarrow slack(rev(e^*)) + \Delta$
        delete $(p \to q)^*$ from $L$
        if $q \neq 0$ << that is, if $pred(q) \neq \emptyset$ >>
            insert $(pred(q) \to q)^*$ into $L$
        $pred(q) \leftarrow p$
for each edge $e$
        $\phi(e) \leftarrow c(e) - slack(e^*)$
```

} maintain T_λ as λ increases by pivoting until λ_{max}

} Compute the flow

Force-Directed Methods [Chapter 10: Graph Drawing by Di Battista, Eades, Tamassia, Tollis]

- use a physical analogy to draw graphs
- the algorithm seeks a configuration of the bodies with locally minimal energy (the sum of the forces on each body is zero) $\longrightarrow$ equilibrium configuration

forced-directed methods in general have two parts:

- The model: A force system defined by the vertices and edges, which provides a physical model for the graph
- The algorithm: A technique for finding an equilibrium state of the force system that is a position for each vertex, such that the total force on every vertex is zero. This state defines a drawing of the graph.

10.1 Springs and Electrical Forces

- edges are modeled as springs
- vertices are equally charged particles which repel each other

the force on vertex v is:

$$F(v) = \underbrace{\sum_{(u,v)\in E} f_{uv}}_{\text{spring}} + \underbrace{\sum_{(u,v)\in V\times V} g_{uv}}_{\text{electrical repulsion}}$$

spring $\xrightarrow{\text{follows}}$ Hooke's law

electrical repulsion $\xrightarrow{\text{follows}}$ inverse square law

- $d(p,q)$: Euclidean distance between points p and q
- $P_v(x_v, y_v)$: the position of vertex v

So

x component of the force $F(v)$

$$= \sum_{(u,v)\in E} k^{(1)}_{uv}\left(d(p_u,p_v) - l_{uv}\right)\frac{x_v - x_u}{d(p_u,p_v)} + \sum_{(u,v)\in V\times V} \frac{k^{(2)}_{uv}}{(d(p_u,p_v))^2}\cdot\frac{x_v - x_u}{d(p_u,p_v)}$$

- l_{uv} : the natural (zero energy) length of the string between u and v is l_{uv}.
- $k^{(1)}_{uv}$: the stiffnes of the string between u and v
- $k^{(2)}_{uv}$: the strength of the electrical repulsion between u and v.

benefits (aims to satisfy important aesthetics)

- The spring force is aimed to ensure that the distance between adjacent vertices u and v is approximately equal to l_{uv}.
- The electrical force aims to ensure that vertices should not be close together
- Under certain assumptions the drawing tends to be symmetric

If we use <u>logarithmic</u> springs rather than Hooke's law springs, then

$$x \text{ component of } f_{uv} = k^{(1)}_{uv} \log\left(\frac{d(p_u, p_v)}{l_{uv}}\right) \frac{x_v - x_u}{d(p_u, p_v)}$$

$\boxed{\text{"follow your nose" algorithm}}$

Vertices are initially placed at random locations.
At each iteration, the force $F(v)$ on each vertex is computed, and each vertex v is moved in the direction of $F(v)$ by a small amount proportional to the magnitude of $F(v)$

10.2 The Barycenter Method.

Tutte's algorithm: $l_{uv} = 0$. $k^{(1)}_{uv} = 1$. $k^{(2)}_{uv} = 0$

Thus force $F(v) = \sum_{(u,v)\in E} (p_u - p_v)$

- problem. trivial solution $p_v = 0$ for all v.
- to avoid: the vertex set V is partitioned into two sets, a set of at least three fixed vertices, and a set of free vertices

We choose p_v so that $F(v) = 0$ for each free vertex v:

$$\begin{cases} \sum_{(u,v)\in E} (x_u - x_v) = 0 \\ \sum_{(u,v)\in E} (y_u - y_v) = 0 \end{cases}$$

Let $N_0(v)$ denote the set of fixed neighbours of v
Let $N_1(v)$ denote the set of free neighbours of v.

Solving them amounts to placing each free vertex at the <u>barycenter of its neighbours</u>

$$\begin{cases} \deg(v)\, x_v - \sum_{u\in N_1(v)} x_u = \sum_{w\in N_0(v)} x_w^* \\ \deg(v)\, y_v - \sum_{u\in N_1(v)} y_u = \sum_{w\in N_0(v)} y_w^* \end{cases}$$

(x_w^*, y_w^*) is a position of a fixed vertex

$\deg(v)$ degree of v.

- The matrix is diagonally dominant. In practice, a simple Newton-Raphson iteration converges quickly.
- For planar graphs, the matrix is sparse and it is possible to solve the equations in $O(n^{1.5})$ [LRT79]

$\boxed{\text{Algorithm 10.1 Barycenter-Draw}}$

```
Input: graph $G=(V,E)$; a partition $V = V_0 \cup V_1$ of $V$ into a set $V_0$ of
       at least 3 fixed vertices and a set $V_1$ of free vertices; a strictly
       convex polygon $P$ with $|V_0|$ vertices
Output: a position $p_v$ for each vertex of $V$, such that the fixed vertices
        form a convex polygon $P$.

1. Place each fixed vertex $u \in V_0$ at a vertex of $P$, and each free vertex
   at the origin
2. repeat
       foreach free vertex $v$ do
           $x_v = \frac{1}{\deg(v)} \sum_{(u,v) \in E} x_u$
           $y_v = \frac{1}{\deg(v)} \sum_{(u,v) \in E} y_u$
   until $x_v$ and $y_v$ converge for all free vertices $v$.
```

Theorem 10.1 Suppose that G is a triconnected planar graph, f is a face in a planar embedding of G, and P is a strictly convex planar drawing of f. Then applying the barycenter algorithm, with the vertices of f fixed and positioned according to P, yields a convex planar drawing of G.

- the resolution is poor ?
- for every $n > 1$ there is a graph G s.t. the barycentric method outputs a drawing exponential area for any resolution rule.
- can be generalized to drawings obtained with a more complex energy function. For Barycenter-Draw, the energy of a drawing is the sum of the squares of the lengths of the edges.
- more generally, we can define the energy of a drawing as the sum of pth powers of the edge lengths.

10.3 Forces Simulating Graph Theoretic Distances

[KS80] J.B. Kruskal and J.B. Seery, Designing Network Diagrams
[KK89] T. Kamada and S. Kawai, An algorithm for Drawing General Undirected Graphs

graph theoretic distance $\delta(u,v)$

$G=(V,E)$ is a connected graph, $u,v \in V$, the graph theoretic distance, denoted by $\delta(u,v)$, is the number of edges on a shortest path between u and v

aim : find a drawing in which, for each pair u,v of vertices, the Euclidean distance $d(p_u,p_v)$ between u and v is approximately proportional to $\delta(u,v)$ between all pairs u and v of G.
Thus the system has a force proportional to $d(p_u,p_v) - \delta(u,v)$ between vertices u and v

Kamada and Kawai

take an energy view of this intuition
The potential energy in the spring between u and v is the intergral of the force that the spring exerts, that is

$$\frac{1}{2} \underset{\text{stiffness parameter}}{\underset{\uparrow}{k_{uv}}} \left(d(p_u,p_v) - \delta(u,v)\right)^2.$$

Kamada chooses $k_{uv} = \dfrac{k}{\delta(u,v)^2}$ for a constant k.

energy in (u,v) is $\quad \eta = \dfrac{k}{2}\left(\dfrac{d(p_u,p_v)}{\delta(u,v)} - 1\right)^2$

The energy η in the whole drawing is the sum of these individual energies,

$$\eta = \frac{k}{2} \sum_{u \neq v \in V} \left(\frac{d(p_u,p_v)}{\delta(u,v)} - 1\right)^2$$

minima occur when the partial derivatives of η, with respect to x_v and y_v are zero.
This gives a set of $2|V|$ equations

$$\underbrace{\frac{\partial \eta}{\partial x_v} = 0, \quad \frac{\partial \eta}{\partial y_v} = 0, \quad v \in V}_{\text{nonlinear}}$$

An iterative approach may be used to solve them.
At each step, a vertex is moved to a position that minimizes energy, while all other vertices remain fixed.
The vertex to be moved is chosen as the one that has the largest force acting on it, that is, $\sqrt{\left(\frac{\partial\eta}{\partial x_v}\right)^2+\left(\frac{\partial\eta}{\partial y_v}\right)^2}$ is maximized over all $v \in V$.

10.4 Magnetic Fields

Sugiyama and Misue [SM95a, SM95b]
"Graph Drawing by Magnetic-Spring Model"
"A Simple and Unified Method for Drawing Graphs: Magnetic-Spring Algorithm"

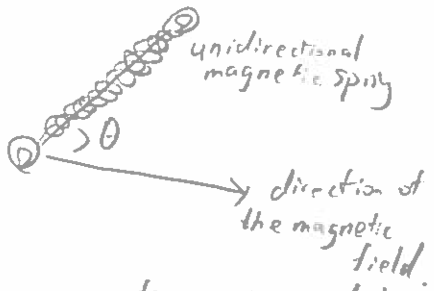
A model in which some or all of the springs are magnetized and there is a global magnetic field that acts on the springs.

three basic types of magnetic field:

- Parallel: All magnetic forces operate in the same direction
- Radial. The forces operate radially outward from a point
- Concentric: The forces operate in concentric circles.

the strings can be magneticed in 2 ways.
- Unidirectional
- Bidirectional

furthermore, a string may not be magnetized at all.


unidirectional magnetic spring
θ
direction of the magnetic field

for a unidirectionally magnetized spring representing the edge (u,v), the force is proportional to $d(p_u, p_v)^{\alpha}\theta^{\beta}$, α, β are constants.

$*$ the magnetic spring model is able to handle directed graphs ← arcs point downward / outward / counterclockwise
- the method has also been applied with some success to orthogonal drawings (close to an hv drawing)

mixed graphs : graphs with both directed and undirected edges.

10.5 General Energy Functions

- including discrete energy functions

for example, for each drawing, we can define:

- the number of crossings
- the number of horizontal and vertical edges
- the number of bends in edges

main problem:

* it may be computationally expensive to find a minimum energy state.

we must resort to very general optimization methods such as simulated annealing and genetic algorithms.?

aesthetics conflict with each other, we cannot expect to optimize several criteria simultaneously

we can use an energy function that linearly combines a number of measures:

$$\eta = \lambda_1 \eta_1 + \lambda_2 \eta_2 + \dots + \lambda_k \eta_k$$

η measures the "ugliness" of the drawing and a drawing of minimum energy has maximum beauty.

Davidson and Harel [DH96]: Drawing Graphics Nicely Using Simulated Annealing

$$\eta = \lambda_1 \eta_1 + \lambda_2 \eta_2 + \lambda_3 \eta_3 + \lambda_4 \eta_4$$

$\eta_1 = \sum_{u,v \in V} \frac{1}{d(p_u,p_v)^2}$: similar electrical repulsion

aims to ensure the vertices do not come too close together

$\eta_2 = \sum_{u \in V} \left(\frac{1}{r_u^2} + \frac{1}{l_u^2} + \frac{1}{t_u^2} + \frac{1}{b_u^2} \right)$

r_u, l_u, t_u, b_u are the Euclidean distances between vertex u and the four sidelines of the rectangular frame in which the graph is drawn.

ensures vertices do not come to close to the borders of the screen.

$\eta_3 = \sum_{(u,v) \in E} d(p_u,p_v)^2$ edges do not become too long

$\eta_4 =$ the number of edge crossings in the drawing ? (How to minimize?).

The flexibility of general energy function methods allows a variety of aesthetics to be used by adjusting the coefficients λ_i.

[BBS 97] J. Branke, F. Bucher and H. Schmeck.

Using genetic algorithms for drawing undirected graphs.

10.6 Constraints

Force-directed methods can handle:

- Position constraints
- Fixed-subgraph constraints
- Constraints that can be expressed by forces or energy functions

- Iterative methods can confine the movement of vertices to the prescribed region at each iteration.

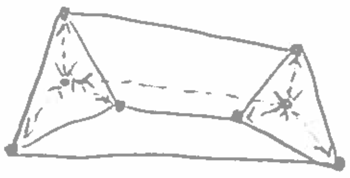
- the barycenter method may be seen as a force-directed method that constraints a set of vertices to a polygon shape.

Constraints that can be expressed by forces include:

- Orientaton of directed edges in a given direction. e.g., horizontal or vertical
- Geometric clustering of specidied sets vertices
- Alignment of vertices

Achieve clustering:

1. For each set C of vertices which need to be clustered, add to the graph a dummy "attractor" vertex v_C.
2. Add attractive forces between an attractor v_C and each vertex in C.
3. Add repulsive forces between pairs of attractors and between attractors and vertices not in any cluster.



[ECH97] P. Eades, R. Cohen and M. Huang. Online animated graph drawing for web navigation.

10.7 Remarks

[Ost 96] 1) Equations describing the minimal energy states are stiff? for some graphs of low connectivity.
2) Replacing the cliques of a graph by stars can improve the speed of spring algorithms for dense graphs.

Force-directed algorithms are heuristics which are best analyzed empirically.

[Ost 96] D. Ostry. Drawing Graphs on Convex Surfaces. Master's thesis.

Linear Algebra Background Knowledge.

determinant: $\det(A)$, $\det A$, $|A|$

Geometrically, it can be viewed as the scaling factor of the linear transformation described by the matrix.

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

$$|A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

The determinant gives the signed n-dimensional volume of this parallelotope, $\det(A) = \pm \mathrm{vol}(P$
In particular, if the determinant is zero, then this parallelotope has volume zero and is not fully n-dimensional, which indicates that the dimension of the image of A is less than n.
This means that A produces a linear transformation which is neither onto nor one-to-one, and so is not invertible.

$$\begin{vmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{vmatrix} = a\begin{vmatrix} f & g & h \\ j & k & l \\ n & o & p \end{vmatrix} - b\begin{vmatrix} e & g & h \\ i & k & l \\ m & o & p \end{vmatrix} + c\begin{vmatrix} e & f & h \\ i & j & l \\ m & n & p \end{vmatrix} - d\begin{vmatrix} e & f & g \\ i & j & k \\ m & n & o \end{vmatrix}$$

characteristic polynomial:

The characteristic polynomial of a square matrix is a polynomial which is invariant under matrix similarity and has eigenvalues as roots.

$$p_A(t) = \det(tI - A)$$

invertible = nonsingular = nondegenerate

Singular: A square matrix A that is not invertible is called singular or degenerate.
A square matrix is singular if and only if its determinant is 0.

multiplicity: the multiplicity of a member of a multiset is the number of times it appears in the multiset.

Spectral Graph Theory by Daniel A. Spielman

{Lecture 1}

- The hypercube on 2^k vertices. The vertices are elements of $\{0,1\}^k$. Edges exist between vertices that differ in only one coordinate.

Matrices for Graphs

adjacency matrix, M_G, whose entries $M_G(a,b)$ are given by

$$M_G(a,b) = \begin{cases} 1 & \text{if } (a,b) \in E \\ 0 & \text{otherwise} \end{cases}$$

* index the rows and columns of the matrix by vertices, rather than by number.

view M as an operator:

the most natural operator associated with a graph G is probably its diffusion operator.
This operator describes the diffusion of stuff among the vertices of a graph and how random walks behave.

use M to define a quadratic form:

the most natural quadratic form associated with a graph is defined in terms of its Laplacian matrix.

$$L_G \overset{def}{=} D_G - M_G.$$

D_G is the diagonal matrix in which $D_G(a,a)$ is the degree of vertex a. $d(a)$.

In a weighted graph, we use the weighted degree: the sum of the weights of edges attached to the vertex a.

Given a function on the vertices, $x \in \mathbb{R}^V$, the Laplacian quadratic form is:

$$x^T L_G x = \sum_{(a,b)\in E} (x(a) - x(b))^2$$

This form measures the smoothness of the function x.
It will be small if the function x does not jump too much over any edge.
$x(a)$ denotes the coordinate of vector x corresponding to vertex a

a vector ψ is an eigenvector of a Matrix M with eigenvalue λ if

$$M\psi = \lambda\psi$$

λ is an eigenvalue if and only if $\lambda I - M$ is a singular matrix ?

Thus, the eigenvalues are the roots of the characteristic polynomial of M:

$$\det(xI - M)$$

Theorem 1.6.1 [The Spectral Theorem] If M is an n-by-n, real, symmetric matrix, then there exist real numbers $\lambda_1, \ldots, \lambda_n$ and n mutually orthogonal unit vectors $\psi_1, \ldots, \psi_n$ and such that ψ_i is an eigenvector of M of eigenvalue λ_i, for each i.

- If the matrix is not symmetric, it might not have n eigenvalues.
- If the eigenvectors are orthogonal, then the matrix is symmetric ?

Fact 1.6.2 The Laplacian matrix of a graph is positive semidefinite. That is, all its eigenvalues are nonnegative.

proof Let ψ be a unit eigenvector of L of eigenvalue λ. Then,

$$\psi^T L \psi = \psi^T \lambda \psi = \lambda = \sum_{(a,b)\in E} (\psi(a) - \psi(b))^2 \geq 0$$

we always number the eigenvalues of the Laplacian from smallest to largest.

Thus, $\lambda_1 = 0$. ? $\longrightarrow$ has a constant eigenvector

We will refer to λ_2, and in general λ_k for small k, as low-frequency eigenvalues.

λ_n is a high-frequency eigenvalue.

$\downarrow$

the curves they traces out resemble the low-frequency modes of vibration of a string

We will relate low-frequency eigenvalues to connectivity.

We will relate high-frequency eigenvalues to problems of graph coloring and finding independent sets.

1.7.2 Spectral Graph Drawing

We can often use the low-frequency eigenvalues to obtain a nice drawing of a graph.

That's a great way to draw a graph if you start out knowing nothing about it.

1.7.3 Graph Isomorphism

If we permute the vertices then the eigenvectors are similarly permuted. That is, if P is a permutation matrix, then

$$L\psi = \lambda\psi \quad \text{if and only if} \quad (PLP^T)(P\psi) = \lambda(P\psi)$$

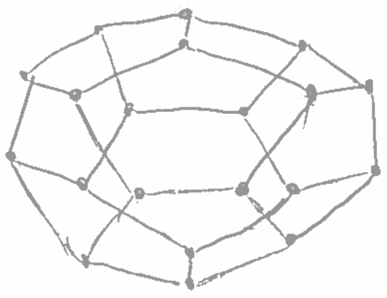
because $P^T P = I$.

"Graph Isomorphism Testing Problem"

First, check if the two graphs have the same sets of eigenvalues.
If they don't, they they are not isomorphic.
If they do, and the eigenvalues have multiplicity one then draw the pictures. If the pictures are the same, up to horizontal or vertical flips, and no vertex is mapped to the same location as another, then by lining up the pictures we can recover the permutation.

1.7.4 Platonic Solids

dodecahedron



We really shouldn't be drawing this picture in two dimensions: the smallest non-zero eigenvalue of the Laplacian has multiplicity three. So we can't reasonably choose just two eigenvectors. We should be choosing three that span the eigenspace

1.7.5 The Fiedler Value

The second-smallest eigenvalue of the Laplacian matrix of a graph is zero if and only if the graph is disconnected.

If G is disconnected, then we can partition it into two graphs G_1 and G_2 with no edges between them.

$$L_G = \begin{pmatrix} L_{G_1} & 0 \\ 0 & L_{G_2} \end{pmatrix}$$

Fiedler suggested that think of λ_2 as a measure of how well connected the graph is. He called it "Algebraic Connectivity" of a graph, and we call it the "Fiedler value".

Fiedler proved that the further λ_2 is from 0, the better connected the graph is.

Cheeger's inequality.

If λ_2 is small, then for some t, the set of vertices.

$$S_i \overset{\text{def}}{=} \{ i : v_2(i) < t \}$$

may be removed by cutting much less than $|S_i|$ edges.

The smallest eigenvalue of the diffusion matrix is zero if and only if the graph is bipartite.

1.7.7 Planar Graphs

We will prove that graphs that can be drawn nicely must have small Fiedler value.

1.7.9 Expanders

Roughly speaking, expanders are sparse graphs (say a number of edges linear in the number of vertices), in which λ_2 is bounded away from zero by a constant.

2.1 Eigenvalues and Optimization {Lecture 2}

Eigenvalues arise as the solution to natural optimization problems.

Theorem 2.1.1 Let M be a symmetric matrix and let x be a non-zero vector that maximizes the Rayleigh quotient with respect to M:

$$\frac{x^T M x}{x^T x}$$

Then, x is an eigenvector of M with eigenvalue equal to the Rayleigh quotient. Moreover, this eigenvalue is the largest eigenvalue of M.

Proof. We recall that the gradient of a function at its maximum must be the zero vector. Let's compute the gradient.

We have
$$\nabla x^T x = 2x$$
and
$$\nabla x^T M x = 2Mx$$

So,
$$\nabla \frac{x^T M x}{x^T x} = \frac{(x^T x)(2Mx) - (x^T M x)(2x)}{(x^T x)^2}$$

In order for this to be zero, we must have

$$Mx = \frac{x^T M x}{x^T x} x$$

That is, if and only if x is an eigenvector of M with eigenvalue equal to its Rayleigh quotient. $\square$

Theorem 2.1.2 (Courant-Fischer Theorem). Let L be a symmetric matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Then

$$\lambda_k = \min_{\substack{S \subseteq \mathbb{R}^n \\ \dim(S)=k}} \max_{x \in S} \frac{x^T L x}{x^T x} = \max_{\substack{T \subseteq \mathbb{R}^n \\ \dim(T)=n-k+1}} \min_{x \in T} \frac{x^T L x}{x^T x}.$$

For example, consider $k=1$, S is just the span ψ_1 and T is all of $\mathbb{R}^n$.

Lemma 2.1.3 Let M be a symmetric matrix with eigenvalues $\mu_1, \ldots, \mu_n$ and a corresponding orthonormal basis of eigenvectors $\psi_1, \ldots, \psi_n$. Let x be a vector and expand x in the eigenbasis as

$$x = \sum_{i=1}^{n} c_i \psi_i.$$

Then,

$$x^T M x = \sum_{i=1}^{n} c_i^2 \lambda_i.$$

You should check for yourself (or recall) that $c_i = x^T \psi_i$

Proof. Compute:

$$\begin{aligned} x^T M x &= \Big(\sum_i c_i \psi_i\Big)^T M \Big(\sum_j c_j \psi_j\Big) \\ &= \Big(\sum_i c_i \psi_i\Big)^T \Big(\sum_j c_j \lambda_j \psi_j\Big) \\ &= \sum_{i,j} c_i c_j \lambda_j \psi_i^T \psi_j \\ &= \sum_i c_i^2 \lambda_i \end{aligned}$$

as $\psi_i^T \psi_j = 0$ for $i \neq j$ □

Proof of 2.1.2 Let $\psi_1, \ldots, \psi_n$ be an orthonormal set of eigenvectors of L corresponding to $\lambda_1, \ldots, \lambda_n$. We will just verify the first characterization of λ_k. The other is similar

First, let's verify that λ_k is achievable. Let S_k be the span of $\psi_1, \ldots, \psi_k$. We can expand every $x \in S_k$ as

$$x = \sum_{i=1}^{k} c_i \psi_i$$

Applying Lemma 2.1.3 we obtain

$$\frac{x^T L x}{x^T x} = \frac{\sum_{i=1}^k \lambda_i c_i^2}{\sum_{i=1}^k c_i^2} \leq \frac{\sum_{i=1}^k \lambda_k c_i^2}{\sum_{i=1}^k c_i^2} = \lambda_k$$

To show that this is in fact the maximum, we will prove that for all subspaces S of dimension k,

$$\max_{x \in S} \frac{x^T L x}{x^T x} \geq \lambda_k.$$

Let T_k be the span of $\psi_k, \dots, \psi_n$. As T_k has dimension $n-k+1$, every S of dimension k has an intersection with T_k of dimension at least 1. So,

$$\max_{x \in S} \frac{x^T L x}{x^T x} \geq \max_{x \in S \cap T_k} \frac{x^T L x}{x^T x}$$

Any such x may be expressed as

$$x = \sum_{i=k}^{n} c_i \psi_i.$$

and so

$$\frac{x^T L x}{x^T x} = \frac{\sum_{i=k}^{n} \lambda_i c_i^2}{\sum_{i=k}^{n} c_i^2} \geq \frac{\sum_{i=k}^{n} \lambda_k c_i^2}{\sum_{i=k}^{n} c_i^2} = \lambda_k$$

<u>Theorem 2.1.4</u> Let L be an $n \times n$ symmetric matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ with corresponding eigenvectors $\psi_1, \dots, \psi_n$. Then

$$\lambda_i = \min_{x \underset{?}{\perp} \psi_1, \dots, \psi_{i-1}} \frac{x^T L x}{x^T x},$$

and the eigenvectors satisfy

$$\psi_i = \arg \min_{x \underset{?}{\perp} \psi_1, \dots, \psi_{i-1}} \frac{x^T L x}{x^T x}.$$

2.2 Drawing with Laplacian Eigenvalues

'drawing a graph on a line, that is, mapping each vertex to a real number. Let $x \in \mathbb{R}^V$ be the vector that describes the assignment of a real number to each vertex. We would like most pairs of vertices that are neighbours to be close to one another. So Hall suggested that we choose an x minimizing

$$x^T L x \qquad (2.1)$$

Unless we place restrictions on x, the solution will be degenerate.
To avoid this, and to fix the scale of the embedding overall, we require

$$\sum_{a \in V} x(a)^2 = \|x\|^2 = 1. \qquad (2.2)$$

Even with this restriction, another degenerate solution is possible; every vertex maps to $\frac{1}{\sqrt{n}}$.
To prevent this from happening, we add the additional restriction that

$$\sum_{a} x(a) = 1^T x = 0 \qquad (2.3)$$

As 1 is the eigenvector of the 0 eigenvalue of the Laplacian, the nonzero vectors that minimize (2.1) subject to (2.2) and (2.3) are the unit eigenvectors of the Laplacian of eigenvalue λ_2.

Of course, we really want to draw a graph in two dimensions. So, we will assign two coordinates to each vertex given by x and y. As opposed to minimizing (2.1), we will minimize

$$\sum_{(a,b) \in E} \left\| \begin{pmatrix} x(a) \\ y(a) \end{pmatrix} - \begin{pmatrix} x(b) \\ y(b) \end{pmatrix} \right\|^2$$

This turns out not to be so different from minimizing (2.1), as it equals

$$\sum_{(a,b) \in E} (x(a) - x(b))^2 + (y(a) - y(b))^2 = x^T L x + y^T L x.$$

as before, we impose the scale conditions

$$\|x\|^2 = 1 \quad \text{and} \quad \|y\|^2 = 1,$$

and the centering constraints

$$1^T x = 0 \quad \text{and} \quad 1^T y = 0.$$

However, this still leaves us with the degenerate solution $x = y = \psi_2$.
To ensure that the two coordinates are different, Hall introduced the restriction that x be orthogonal to y. One can use the spectral theorem to prove that the solution is given by setting $x = \psi_2$ and $y = \psi_3$, or by taking a rotation of this solution.

2.3 Isoperimetry and λ_2

Let S be a subset of the vertices of a graph. One way of measuring how well S can be separated from the graph is to count the number of edges connecting S to the rest of the graph. These edges are called the boundary of S, which we formally define by

$$\partial(S) \overset{def}{=} \{ (a,b) \in E : a \in S, b \notin S \}$$

We are less interested in the total number of edges on the boundary than in the ratio of this number to the size of S itself.

We will call this ratio the isoperimetric ratio of S, and define it by

$$\theta(S) \overset{def}{=} \frac{|\partial(S)|}{|S|}$$

The isoperimetric number of a graph is the minimum isoperimetric number over all sets of at most half the vertices:

$$\theta_G \overset{def}{=} \min_{|S| \leq n/2} \theta(S).$$

We will now derive a lower bound on θ_G in terms of λ_2.

Theorem 2.3.1 For every $S \subset V$

$$\theta(S) \geq \lambda_2 (1-s)$$

where $s = \frac{|S|}{|V|}$. In particular

$$\theta_G \geq \frac{\lambda_2}{2}$$

Proof. As

$$\lambda_2 = \min_{x : x^T 1 = 0} \frac{x^T L_G x}{x^T x}$$

for every non-zero x orthogonal to 1 we know that

$$x^T L_G x \geq \lambda_2 x^T x$$

To exploit this inequality, we need a vector related to the set S.
A natural choice is χ_S, the characteristic vector of S,

$$\chi_S(a) = \begin{cases} 1 & \text{if } a \in S \\ 0 & \text{otherwise} \end{cases}$$

We find

$$\chi_S^T L_G \chi_S = \sum_{(a,b)\in E} (\chi_S(a) - \chi_S(b))^2 = |\partial(S)|$$

However, χ_S is not orthogonal to $\mathbf{1}$. To fix this, use

$$x = \chi_S - s\mathbf{1}.$$

so

$$x(a) = \begin{cases} 1-s & \text{for } a \in S, \text{ and} \\ -s & \text{otherwise} \end{cases}$$

We have $x^T \mathbf{1} = 0$, and

$$x^T L_G x = \sum_{(a,b)\in E} ((\chi_S(a) - s) - (\chi_S(b) - s))^2 = |\partial(S)|.$$

To finish the proof, we compute

$$x^T x = |S|(1-s)^2 + (|V| - |S|)s^2 = |S|(1-2s+s^2) + |S|s - |S|s^2 = |S|(1-s)$$

This gives

$$\lambda_2 \le \frac{\chi_S^T L_G \chi_S}{\chi_S^T \chi_S} = \frac{|\partial(S)|}{|S|(1-s)}$$

This theorem says if λ_2 is big, then G is very well connected.

<u>Claim 2.3.2</u> Let $S \subseteq V$ have size $s|V|$. Then

$$\| \chi_S - s\mathbf{1} \|^2 = s(1-s)|V|.$$

Computational Geometry Lecture Notes by David M. Mount.

{Lecture 11}

Voronoi Diagrams.

A Voronoi diagram encodes proximity information, that is, what is close to what.

let $P = \{p_1, p_2, \dots, p_n\}$ be a set of points in the plane, or more generally, which we call sites.

Let $\|pq\| = (\sum_{i=j}^{d} (p_j - q_j)^2)^2$ denot the Euclidean distance between two points p and q

Define $V(p_i)$, the Voronoi cell for p_i, to be the set of points q in the plane that are closer to p_i than to any other site, that is

$$V(p_i) = \{ q \in \mathbb{R}^d : \|p_i q\| < \|p_j q\|, \forall j \neq i \}.$$

the Voronoi cells of two distinct points of P are disjoint.

The union of the closure of the Voronoi cells defines a cell complex, which is called the Voronoi diagram of P, and is denoted $Vor(P)$.

The cells of the Voronoi diagram are convex polyhedra.

Applications:

- Nearest neighbour queries
- Computational morphology and shape analysis (medial axis)
- Center-based clustering
- Neighbours and interpolation (natural neighbour interpolation)

Properties of the Voronoi diagram (in the plane):

☆

- Empty circle properties:

  Each point on an edge of the Voronoi diagram is equidistant from its two nearest neighbours p_i and p_j. Thus, there is a circle centered at any such point where p_i and p_j lie on this circle, and no other site is interior to the circle.

- Voronoi vertices: the vertex at which three Voronoi cells $V(p_i)$, $V(p_j)$, $V(p_k)$ intersect
called a Voronoi vertex is equidistant from all sites.
Thus it is the center of the circle passing through the sites, and this circle contains no other sites in its interior.

convex hull. A cell of the Voronoi diagram is unbounded if and only if the corresponding site lies on the convex hull.

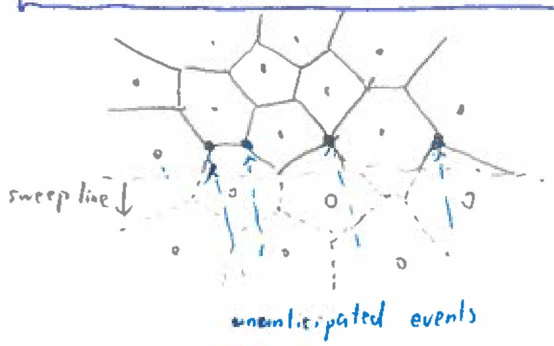
size: Let n denote the number of sites, then the Voronoi diagram has exactly n faces.
It follows from Euler's formula $v - e + f = 2$ The number of Voronoi vertices is roughly $2n$ and the number of edges is roughly $3n$

$3V = 2e$
$V - \frac{3}{2}V + n = 2$
$\frac{-1}{2}V = n - 2$
$V = 2n - 4$
$e = 3n - 6$

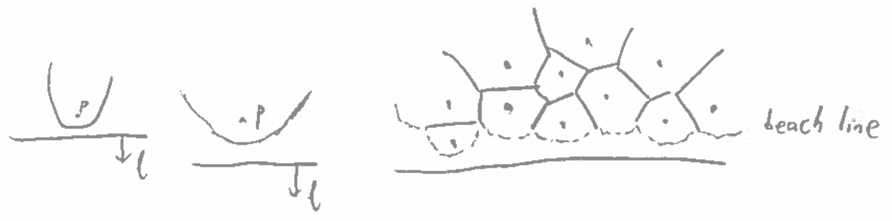
$\boxed{\text{Fortune's Algorithm (Steven Fortune)}}$

$O(n \log n)$ runtime

$O(n)$ space



beach line: The set of points q that are equidistant from the sweep line to their nearest site above the sweep line is called the beach line.



The set of points that are equidistant from a point (a site) and a line (the sweep line) is a parabola.
The beach line consists of the lower envelope of these parabolas, one for each site.
Because the parabolas are x-monotone, so is the beach line.

The point where two arcs of the beach line intersect is called a breakpoint.

Lemma The beach line is an x-monotone curve made up of parabolic arcs. The breakpoints of the beach line line on Voronoi edges of the final diagram.

Sweep-line Status:

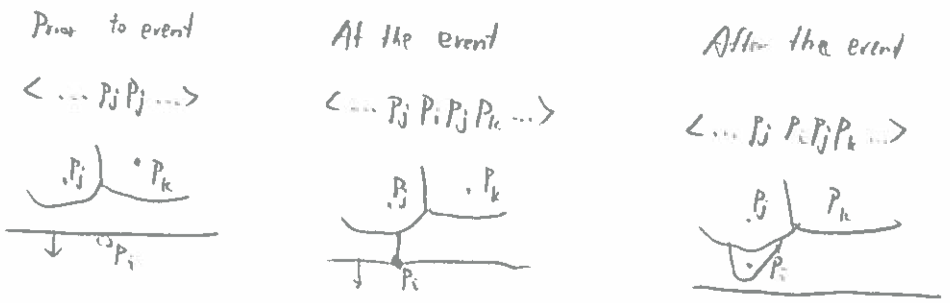
The algoritm maintains the current location (y-coordinate) of the sweep line.
It stores, in left-to-right order, the sequence of sites that define the beach line.

Events:

Site events: When sweep line passes over a new site a new porabolic arc will be inserted into beach line.

Voronoi vertex events: (circle events)
When the length of an arc of the beach line shrinks to zero, the arc disappears and a new Voronoi vertex will be created at this point.

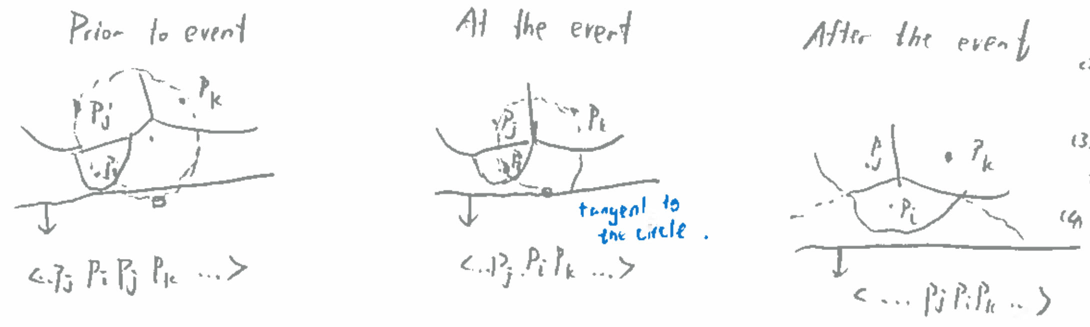
Site events:


Prior to event
$< \dots p_j p_j \dots >$
p_j
p_k
p_i
At the event
$< \dots p_j p_i p_j p_k \dots >$
p_j
p_k
p_i
After the event
$< \dots p_j p_i p_j p_k \dots >$
p_j
p_k
p_i

(1) advance sweep line
(2) replace
$< \dots p_j \dots >$ with $< \dots p_j, p_i, p_j >$
(3) create a new (dangling) edge in the Voronoi diagram, which lies on the bisector between p_i and p_j
(4) some old triples that involved p_j may be deleted and new triples involving p_i may be inserted

The maximum number of arcs on the beach line can be at most $2n-1$. (each site creates 2 new arcs except the last one)
The sites can be presorted by the y-coordinates and inserted as a batch into the event priority queue.

Voronoi vertex events:



(1) delete entry for p_j from beach line status
(2) create a new vertex in Voronoi diagram
(3) create a new (dangling) edge for the bisector between p_i and p_k
(4) delete any events that arose from triples involving the arc of p_j, and generate new events corresponding to consecutive triples involving p_i and p_k.

Sweep-line algorithm:

- Voronoi diagram: The partial Voronoi diagram that has been constructed so far will be stored in any reasonable data structure for storing planar subdivisions e.g. a doubly-connected edge list.

- Beach line: The beach line consists of the sorted sequence of sites whose arcs form the beach line.

- Event queue: The event queue is a priority queue with the ability both to insert and delete new events.

{Lecture 12}

Delaunay Triangulations

The Voronoi diagram of a set of sites in the plane is a planar subdivision, that is, a cell complex. The resulting dual graph is a triangulation of the sites called the Delaunay triangulation.

Properties:

1) Convex hull: The boundary of the exterior face of the Delaunay triangulation is the boundary of the convex hull of the point set

2) Circumcircle property: The circumcircle of any triangle in the Delaunay triangulation is empty. (contains no sites of P).

3) Empty circle property: Two sites p_i and p_j are connected by an edge in the Delaunay triangulation, iff. there is an empty circle passing through p_i and p_j.

4) Closest pair property: The closest pair of sites in P are neighbours in the Delaunay triangulation.

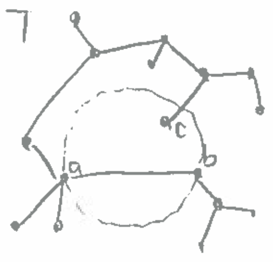
Given a point set P with n vertices where there are h on the convex hull, it is not ? hard to prove by Euler's formula that the Delaunay triangulation has $2n-2-h$ triangles and $3n-3-h$ edges.

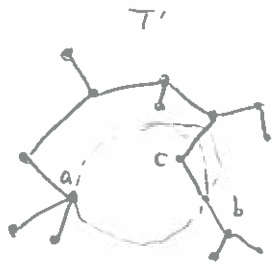
$v+f-e=2$

$n+f-\frac{3}{2}e=2$

f

Theorem The minimum spanning tree of a set of points P (in any dimension) is a subgraph of the Delaunay triangulation.





$w(T') = w(T) + \|bc\| - \|ab\| < w(T)$

Does the Delaunay triangulation minimize the total edge length?

The answer is <u>NO</u>.

The triangulation that minimizes total edge weight is called <u>minimum weight triangulation</u>. In 2008 it was proved that this problem is NP-hard.

<u>Spanner Properties:</u>

Consider any point set P and a straight-line graph G whose vertices are the points of P. For any two points $p, q \in P$, let $\delta_G(p,q)$ denote the length of the shortest path from p to q in G, where the weight of each edge is its Euclidean length.

Given any parameter $t \geq 1$, we say that G is a <u>t-spanner</u> if any two points $p, q \in P$, the shortest path between p and q in G is at most a factor t longer than the Euclidean distance between these points, that is

$$\delta_G(p,q) \leq t \, \|pq\|$$

<u>Theorem</u> (Keil and Gutwin)

Given a set of points P in the space, the Delaunay triangulation of P is a t-spanner for $t = 4\pi\sqrt{3}/9 \approx 2.418$.

<u>Maximizing Angles and Edge Flipping:</u>

<u>Theorem</u> Among all triangulations of a given planar point set, the Delaunay triangulation has the lexicographically largest angle sequence, and in particular it maximizes the minimum angle.

{Lecture 13} <u>Delaunay Triangulations: Incremental Construction.</u>

A simple randomized algorithm for constructing the Delaunay triangulation of a set of n sites in the plane. Its expected running time is $O(n \log n)$.

The input consists of a set $P = \{p_1, \ldots, p_n\}$ of points in the plane.

The idea is to insert sites in random order, one at a time, and update the triangulation with each new addition. After each insertion, the expected number of structural changes in the diagram is $O(1)$.

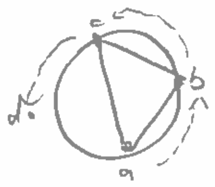
We will store each uninserted sites in a bucket according to the triangle in the current triangulation that contains it. We will show that the expected number of times that a site is rebucketed is $O(\log n)$.

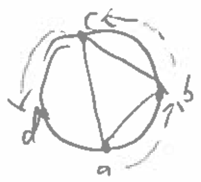
Incircle Test:

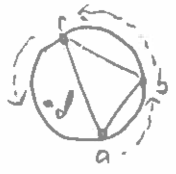
We claim that d lies in the circumcircle determined by the Δabc if and only if the following determinant is positive.

$$\text{inCircle}(a,b,c,d) = \det \begin{pmatrix} a_x & a_y & a_x^2 + a_y^2 & 1 \\ b_x & b_y & b_x^2 + b_y^2 & 1 \\ c_x & c_y & c_x^2 + c_y^2 & 1 \\ d_x & d_y & d_x^2 + d_y^2 & 1 \end{pmatrix} > 0$$

inCircle(a,b,c,d) < 0 inCircle(a,b,c,d) = 0 inCircle(a,b,c,d) > 0







proof. If four points are cocircular then $\exists$ center point $q = (q_x, q_y)$, and radius r, s.t.

$$(a_x - q_x)^2 + (a_y - q_y)^2 = r^2$$

$$\Rightarrow \quad 0 = (-2q_x)\,a_x + (-2q_y)\,a_y + 1\cdot(a_x^2 + a_y^2) + (q_x^2 + q_y^2 - r^2)\cdot 1$$

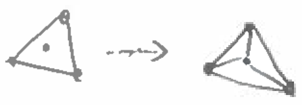
$$\begin{pmatrix} a_x & a_y & a_x^2 + a_y^2 & 1 \\ b_x & b_y & b_x^2 + b_y^2 & 1 \\ c_x & c_y & c_x^2 + c_y^2 & 1 \\ d_x & d_y & d_x^2 + d_y^2 & 1 \end{pmatrix} \begin{pmatrix} -2q_x \\ -2q_y \\ 1 \\ q_x^2 + q_y^2 - r^2 \end{pmatrix} = 0$$

Incremental update:

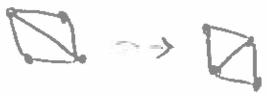
The algorithm begins by inserting three points that are a huge distance from the other points, such that this massive triangle encloses all the other points.

We then insert sites of P one by one. The basic changes are.

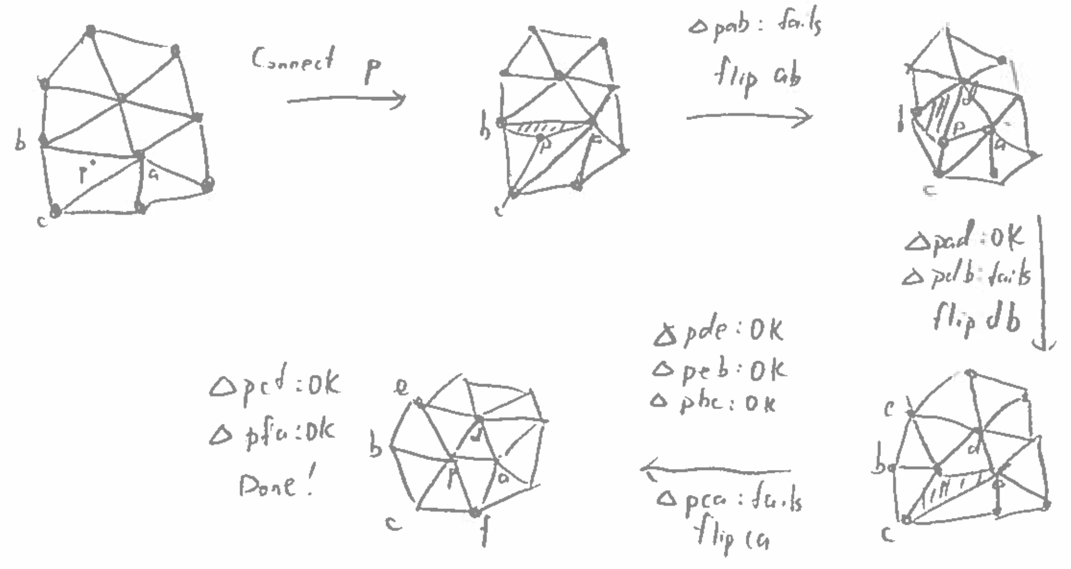
(a) Join a site in the interior of some triangle to the triangle's vertices.



(b) Perform an edge flip.



Both operations can be performed in $O(1)$ time assuming that the triangulation is maintained in any reasonable way, say, as a double-connected edge list. ?



For each of the triangles that have been added, we check the vertex of the triangle that lies on the opposite side of the edge that does not include p.

* Randomized Incremental Delaunay Triangulation Algorithm *

```
Insert (p) {
    Find the triangle △abc containing p;
    Insert edges pa, pb, pc into triangulation;
    SwapTest (ab);                    // check/fix the surrounding edges.
    SwapTest (bc);
    SwapTest (ca);
}

SwapTest (ab) {
    if (ab is an edge on the exterior face) return;
    Let d be the vertex to the right of edge ab;
    if (inCircle (p, a, b, d)) {                  // d violates the incircle test
            Flip edge ab for pd;
            SwapTest (ad);                        // check/fix new suspect edges.
            SwapTest (db);
    }
}
```

- A triangulation is <u>locally Delaunay</u> if for each triangle the vertices lying on the opposite side of each edge of the (up to) three neighbouring triangles satisfy the empty circle condition.
- A triangulation is <u>globally Delaunay</u> means that empty circle condition is satisfied for all triangles and all points of P.

Structial Changes: the expected number of edge changes with each inserton is $O(1)$

The total number of changes made in the triangulation for the insertion of p is proportional to the degree of p after the insertion is complete.

Consider the situation after the insertion of the ith site. The expected time to insert ith site is equal to the average degree of a vertex in the triangulation of i sites.

By Euler's formula we know the average degree of a vertex in any planar graph is at most 6.

(n vertices can have at most $3n$ edges, the sum of vertex degrees is equal to twice the number of edges)

Thus, the expected number of edge changes is $O(1)$.

Summing over all n insertions, the total number of structial changes is $O(n)$. Each structial change can be performed in $O(1)$ time.

Rebucketing: the total expected time spent in rebucketing points is $O(n \log n)$.

We will show that the expected number of times that any site is rebucketed is $O(\log n)$.

Let us fix a site $q \in P$. Consider the situation just after the insertion of the ith site

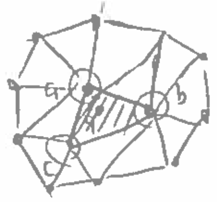
We assert that the probability that q was rebucketed as a result of the last insertion is a most $\frac{3}{i}$.

Let Δ be the triangle containing q after the ith insertion. Δ would have come into existence as a result of the last insertion if and only if one of its three vertices was the last to be added. Thus, the probability that q required rebucketing after the last insertion is at most $3/i$.

After each stage there are $n-i$ points that might subject to rebucketing, and each has probability $\frac{3}{i}$ of being rebucketed. Thus, the expected number of points that require rebucketing as part of the last insertion is at most $(n-i)\frac{3}{i}$.

$$\sum_{i=1}^{n} \frac{3}{i}(n-i) \leq \sum_{i=1}^{n} \frac{3}{i} n = 3n \sum_{i=1}^{n} \frac{1}{i} = 3n \ln n + O(1)$$

Thus, the total expected time spent in rebucketing is $O(n \log n)$.



q would have been rebucketed only if one of a, b, or c was the last to be inserted.

Computational Geometry by David Mount

{ Lecture 3 : Convex Hulls in the Plane }

For any $d \geq 1$, let $\mathbb{R}^d$ denote real d-dimensional space, that is, the set of d-dimensional vectors over the real numbers.

Convexity. A set $K \subseteq \mathbb{R}^d$ is convex if given any points $p, q \in K$, the line segment $\overline{pq}$ is entirely contained within K.

Open / Closed: A set in $\mathbb{R}^d$ is said to be open if it does not include its boundary. A set that includes its boundary is said to be closed.

Boundedness: A convex set is bounded if it can be enclosed with a sphere of a fixed radius. Otherwise, it is unbounded.

Convex body. A closed, bounded convex set is called a convex body.

Support line / hyperplane: An important property of any convex set K in the plane is that at every point p on the boundary of K there exists at least one line ℓ that passes through p such that K lies entirely in one of the closed half planes defined by ℓ

Convex hull. The convex hull of any set P is the intersection of all convex sets that contains P, or more intuitively, the smallest convex set that contains P. We will denote this conv(P).

Convex Hull Problem.

The (planar) convex hull problem is, given a set of n points P in the plane, output a representation of P's convex hull. The convex hull is a closed convex polygon, the simplest representation is a counterclockwise enumeration of the vertices of the convex hull

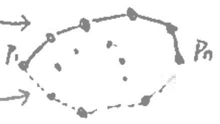
Graham's scan. $O(n \log n)$
(1970s)

incremental construction: add points in increasing order of x-coordinate.
(assume no duplicate x-coordinates)

represent the boundary of the convex hull as two polygon chains

— upper hull

— lower hull



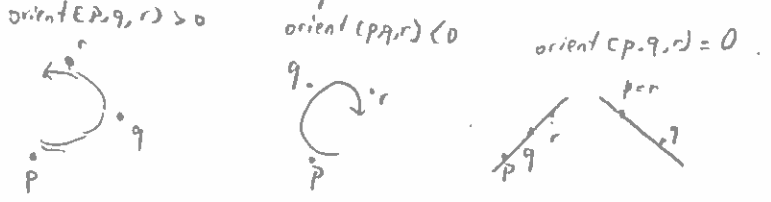
Let $\langle p_1, \dots, p_n \rangle$ denote sorted sequence (sort by x).

For i ranging from 1 to n. we will store the vertices of the current upper hull on a stack S, where the top of the stack corresponds to the most recently added point of P, and the bottom of the stack will be p_1.

Turning and orientations: "left-hand turn"

Given an ordered triple of points $\langle p, q, r \rangle$ in the plane, we say that they have positive orientation if they define a counterclockwise oriented triangle.
negative orientation if they define a clockwise oriented triangle.
zero orientation if they are collinear

Note that orientation depends on the order in which the points are given.



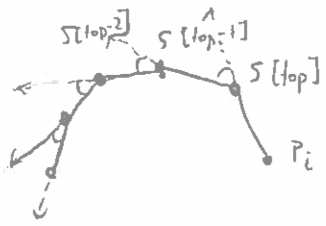
$$\text{Orient}(p, q, r) = \det \begin{pmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{pmatrix}$$

Orientation is formally defined as the sign of the determinant of the points given in homogeneous coordinates, that is, by prepending a 1 to each coordinate.

Given a sequence of three points p, q, r, we say that the sequence $\langle p, q, r \rangle$ makes a (strict) left-hand turn if $\text{Orient}(p, q, r) > 0$.

$\boxed{\text{Graham's Scan}}$

```
(1) Sort the points according to increasing order of their x-coordinates, denoted ⟨p_1, p_2, ..., p_n⟩
(2) push p_1 then p_2 onto S
(3) for i ← 3, ..., n do
      (a) while ( |S| ≥ 2 and Orient(p_i, S[top], S[top-1]) ≤ 0 ) pop S
      (b) push p_i onto S.
```



Let P_i denote the subsequence consisting of the first i points.

Claim: After the insertion of the point p_i, the contents of S (from top to bottom) consist of the vertices of the upper hull of P_i in right to left order.

Running-time analysis:

Let d_i denote the number of points that are popped on processing P_i.
Because each orientation test takes $O(1)$ time the amount of time spent processing P_i is $O(d_i+1)$.
(The extra +1 is for the last point tested, which is not deleted).
Thus, the total running time is proportional to

$$\sum_{i=1}^{n}(d_i+1) = n + \sum_{i=1}^{n} d_i$$

To bound $\sum_i d_i$, observe that each of the n points is pushed onto the stack once.
Once a point is deleted it can never be deleted again. $\sum_i d_i \le n$.

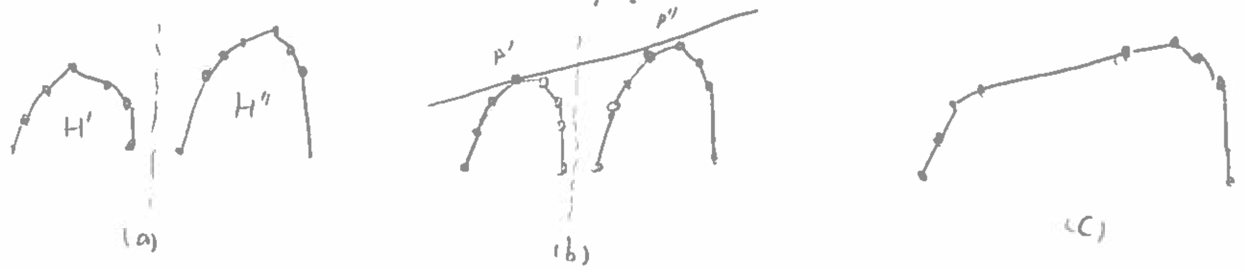
Thus after sorting, the total running time is $O(n)$
Since this is true for lower hull as well, to total time is $O(2n) = O(n)$.

Convex Hull by Divide-and-Conquer

The algorithm begins by sorting the points by their x-coordinate, in $O(n \log n)$ time.
It splits the point set in half at its median x-coordinate, computes the upper hulls of the left and right sets recursively, and then merges the two upper hulls into a single upper hull. This latter process involves computing a line, called the upper tangent that is a line of support for both hulls.

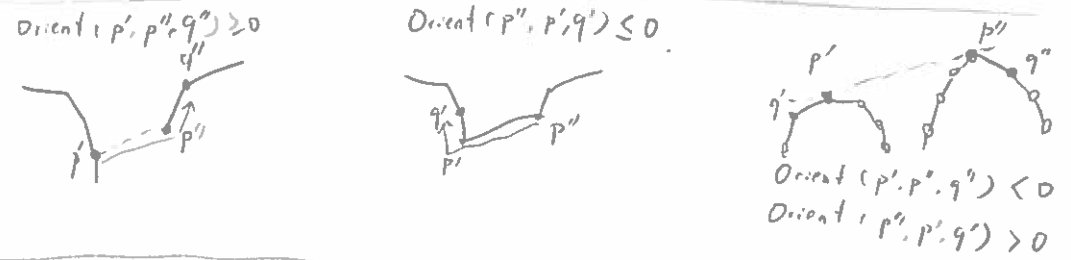
Divide-and-Conquer (Upper) Convex Hull

(1) If $|P| \le 3$, then compute the upper hull by brute force in $O(1)$ time and return.
(2) Otherwise, partition the point set P into two sets P' and P'' of roughly equal sizes by a vertical line.
(3) Recursively compute upper convex hulls of P' and P'', denoted H' and H'', respectively.
(4) Compute the upper tangent $l = \overline{p'p''}$
(5) Merge the two hulls into a single upper hull by discarding all the vertices of H' to the right of p' and the vertices of H' to the left of p''.



$\boxed{\text{Upper Tangent } (H', H''):}$

```
(1) let $p'$ be the rightmost point of $H'$, and let $q'$ be its predecessor.
(2) Let $p''$ be the leftmost point of $H''$, and let $q''$ be its successor
(3) Repeat the following until Orient $(p', p'', q'') < 0$ and Orient $(p'', p', q') > 0$:
    (a) while ( Orient $(p', p'', q'') \geq 0$ ) advance $p''$ and $q''$ to their successors on $H''$
    (b) while ( Orient $(p'', p', q') \leq 0$ ) advance $p'$ and $q'$ to their predecessors on $H'$
(4) return $(p', p'')$.
```



Orient $(p', p'', q'') \geq 0$
q''
p''
p'
Orient $(p'', p', q') \leq 0$
q'
p'
p''
p'
p''
q''
q'
Orient $(p', p'', q'') < 0$
Orient $(p'', p', q') > 0$

The running time is $O(n)$.

Running-time analysis: Given an input of size n.

$$T(n) = \begin{cases} 1 & \text{if } n \leq 3 \\ n + 2T(n/2) & \text{otherwise} \end{cases}$$

$T(n) \in O(n \log n)$.

{ Lecture 4: Convex Hulls: Lower Bounds and Output Sensitivity }

Sorting is polytime reducible to convex hull $\longrightarrow$ lower bound $\Omega(n \log n)$

The reduction leaves open two questions:

☆ 1) What if we don't require that the points be enumerated in cyclic order, just that they all be identified? (even counting the number of points has an $\Omega(n \log n)$ lower bound)

☆ 2) What if all the points of P do not lie on the convex hull?

An algorithm whose running time is expressed as a function of both its input size and output size is called output sensitive.

Gift-Wrapping and Jarvis's March:

?

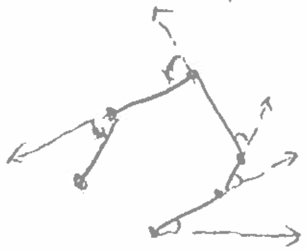
(can be seen as a variant on an $O(n^2)$ sorting algorithm called Selection Sort.

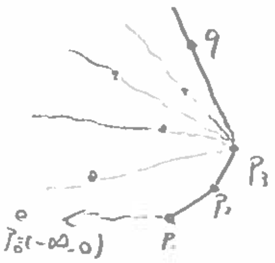
Jarvis's march:

Assuming that there are h vertices on the hull, this algorithm builds the hull in $O(nh)$ time by a process called "gift-wrapping".

It starts by adding one point of P that is guaranteed to be on the hull, say, the point with the smallest y-coordinate. It then repeatedly finds the "next" vertex on the hull in counterclockwise order.

Assuming that p_k and p_{k-1} were the last two points added to the hull, the algorithm finds the point $q \in P$ that maximizes $\angle p_{k-1} p_k q$. Clearly, we can find q in $O(n)$ time.



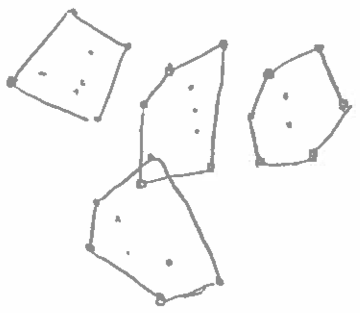


After repeating this h times, we will return back to the starting point and we are done. Thus, the overall running time is $O(nh)$. If $h = o(\log n)$ then this runs asymptotically faster than Graham's scan

☆ Chan's Algorithm: $O(n \log h)$

- It is a fast algorithm that is based on a combination of two slower algorithms Graham's and Jarvis's.
- It is based on "knowing" the final number of vertices on the convex hull. Since this number is not known, it adopts an interesting "guessing strategy" to determine its value (roughly). It is remarkable that the time to run the guess version is asymptotically as if you had known the number in advance.

Partition ($h^* = 8$) and mini-hulls



partition the set into n/h subsets, each of size h.

We can compute the convex hull of each subset in time $O(h \log h)$ by simply applying Graham's scan.

We call each of these a <u>mini-hull</u>

Total time: $O((\frac{n}{h}) \cdot h \log h) = O(n \log h)$

we had an estimate for h, call it h^*, where $h \le h^* \le h^2$

$$O(n \log h^*) = O(n \log h^2) = O(n \log h)$$

<u>Merging the minis:</u>

idea: run Jarvis's algorithm.

treat each mini-hull as if it is a "fat point".

At each step, compute the tangent lines of the current hull vertex to each of the mini-hulls including the mini-hull containing this vertex.

Among these tangents, we take the one that yields the smallest external angle.

Each mini-hull is a convex polygon

<u>Lemma</u>: Consider a convex polygon K in the plane and a point p that is external to K, such that the vertices of K are sorted in cyclic order in an array. Then the two tangents from p to K (more formally, the two supporting lines for K that pass through p) can each be computed in time $O(\log m)$, where m is the number of vertices in K.

```
RestrictedHull(P, h*):
(1) let $r \leftarrow \lceil n/h^* \rceil$.
(2) Partition $P$ into disjoint subsets $P_1, P_2, \ldots, P_r$, each of size at most $h^*$
(3) For ($i \leftarrow 1$ to $r$)
        compute Hull($P_i$) using Graham's scan and store the vertices in an ordered array
(4) let $p_0 \leftarrow (-\infty, 0)$ and let $p_1$ be the bottommost point of $P$.
(5) For ($k \leftarrow 1$ to $h^*$)
        (a) For ($i \leftarrow 1$ to $r$)
                compute point tangent $q_i \in$ Hull($P_i$), that is the vertex of Hull($P_i$) that maximizes the angle $\angle p_{k-1} p_k q_i$
        (b) Let $p_{k+1}$ be the point $q \in \{q_1, \ldots, q_r\}$ that maximizes the angle $\angle p_{k-1} p_k q$
        (c) If $p_{k+1} \leftarrow p_1$ then return $\langle p_1, \ldots, p_k \rangle$ (success).
(6) (Unable to complete the hull after $h^*$ iterations.) Return "Failure. $h^*$ is too small."
```

Guessing the Hull's Size:

$h^* = 1, 2, 3, \ldots, i \implies O(nh \log h)$

$h^* = 1, 2, 4, 8, \ldots, 2^i \implies O(n \log^2 h)$ doubling search

$h^* = 2, 4, 16, 64, \ldots, 2^{2^i} \implies O(n \log h)$ repeatedly squaring

we have $h \le h^* \le h^2$

$h_i^* = 2^{2^i}$ .. the ith trial takes time $O(n \log h_i^*) = O(n \log 2^{2^i}) = O(n 2^i)$

we succeed as soon as $h_i^* \ge h$. $i = \lceil \lg \lg h \rceil$

Thus, the algorithm's total running time is

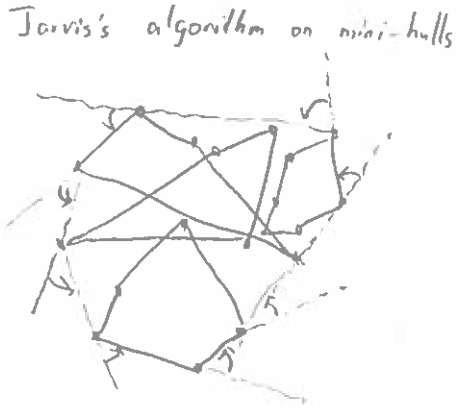
$$T(n,h) = \sum_{i=1}^{\lg\lg h} n 2^i = n \sum_{i=1}^{\lg\lg h} 2^i$$

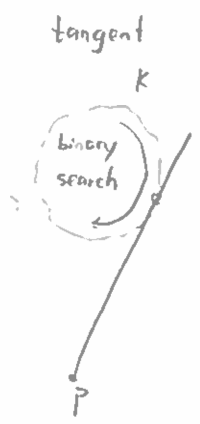
Note that $\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$

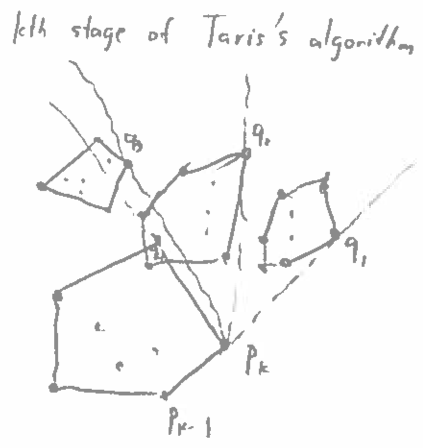
$$T(n,h) < n \cdot 2^{1+\lg\lg h} = 2n \cdot 2^{\lg\lg h} = 2n \lg h = O(n \log h)$$

$\boxed{\text{Hull}(P)}$

```
(1) $h^* \leftarrow 2$. $L \leftarrow$ fail
(2) while ($L$ = fail)
        (a) Let $h^* \leftarrow \min((h^*)^2, n)$
        (b) $L \leftarrow$ RestrictedHull($P, h^*$)
(3) Return $L$.
```







Lower Bound.

We will give an $\Omega(n \log h)$ lower bound on the following simpler decision problem.

Convex Hull Size Verification Problem (CHSV):

Given a point set P and integer h, does the convex hull of P have h distinct vertices?

Assume the algorithm is allowed to compute any algebraic function of the input coordinates. The result is called an algebraic decision tree.

The input to the CHSV problem is a sequence of $2n = N$ real numbers. These numbers form a vector in real N-dimensional space, $(z_1, z_2, \ldots, z_N) = \vec{z} \in \mathbb{R}^N$ which we call a configuration.

Each node of the decision tree is associated with a multivariate algebraic formula of degree at most d, where d is any fixed constant.

$\boxed{\text{(Ben-Or) Theorem}}$ Let $Y \in \mathbb{R}^N$ be any set and let T be any d-th order algebraic decision tree that determines membership in W. If W has M disjoint connected components, then T must have height at least $\Omega((\log M) - N)$.

Multiset Size Verification Problem (MSV): Given a multiset of n real numbers and an integer k, confirm that the multiset has exactly k distinct elements.

Lemma: The MSV problem requires $\Omega(n \log k)$ steps in the worst-case in the d-th order algebraic decision tree.

Consider all the tuples $(z_1, \ldots, z_n)$ with $z_1, \ldots, z_k$ set to the distinct integers from 1 to k. $(k!)$ and $z_{k+1}, \ldots, z_n$ each set to an arbitrary integer in the same range. (k^{n-k})

So there are at least $k! \, k^{n-k}$ different connected components.

MSV $\leq_p$ CHSV.

Let $Z = (z_1, \ldots, z_n)$ and k be an instance of MSV. We create a point set $\{p_1, \ldots, p_n\}$ in the plane where $p_i = (z_i, z_i^2)$, and set $h = k$.