

Separators for Planar Graphs

R. J. Lipton, R. E. Tarjan
1979/80

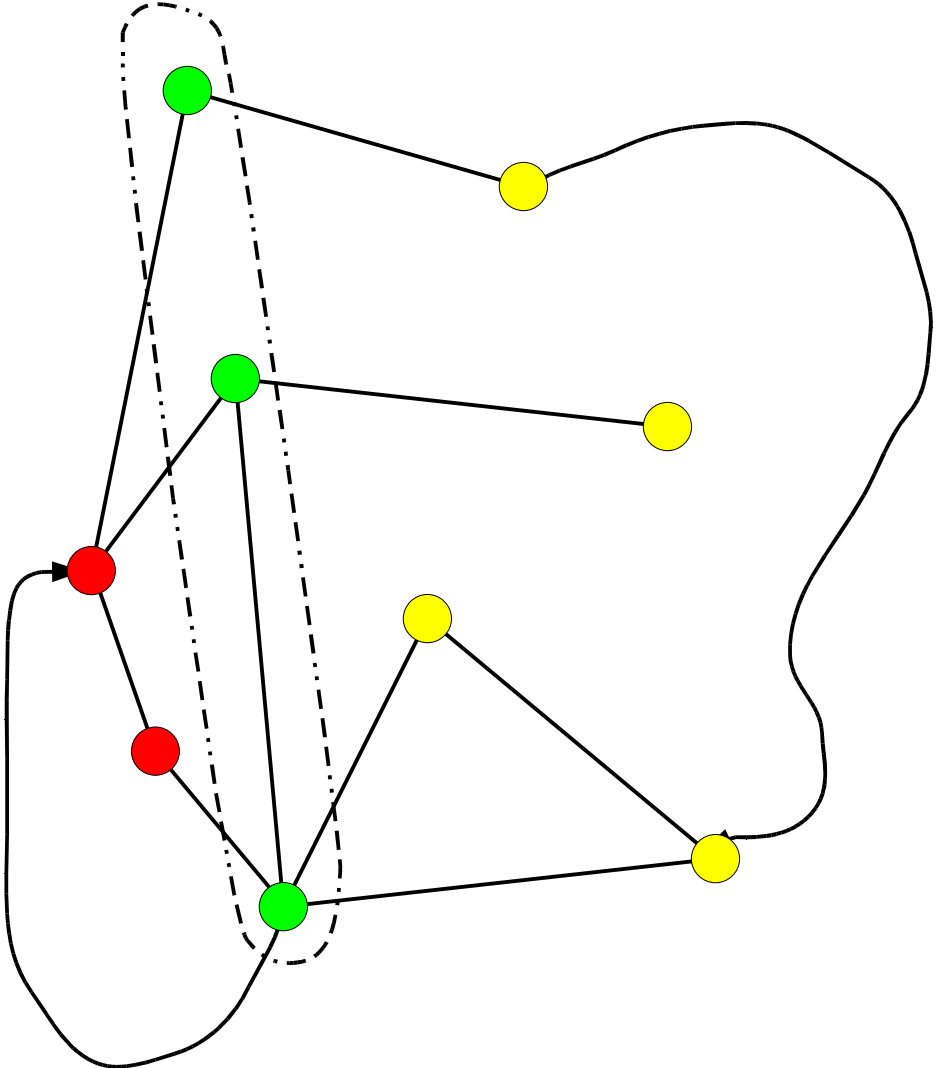
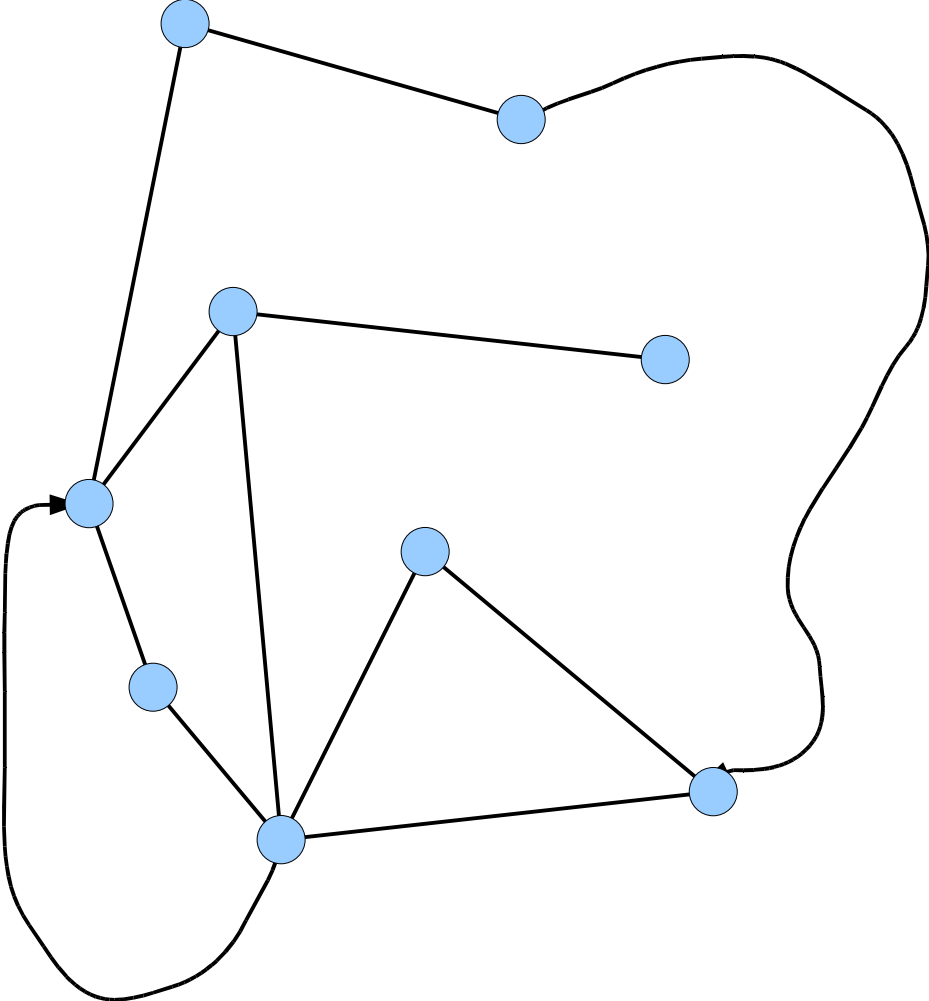
Goal

Theorem (\sqrt{n} -separator theorem):

Let G be any n -vertex planar graph. The vertices of G can be partitioned into three sets A , B , C such that

- no edge joins a vertex in A to a vertex in B
- neither A nor B contains more than $2n/3$ vertices
- C contains no more than $2\sqrt{2}\sqrt{n}$ vertices

Illustration



A



B



C

Generalization

More general setting:

- Each vertex v has a *nonnegative cost* $\mathbf{C}(v)$.
- A set of vertices S has a *total cost* $\mathbf{C}(S)$, defined as

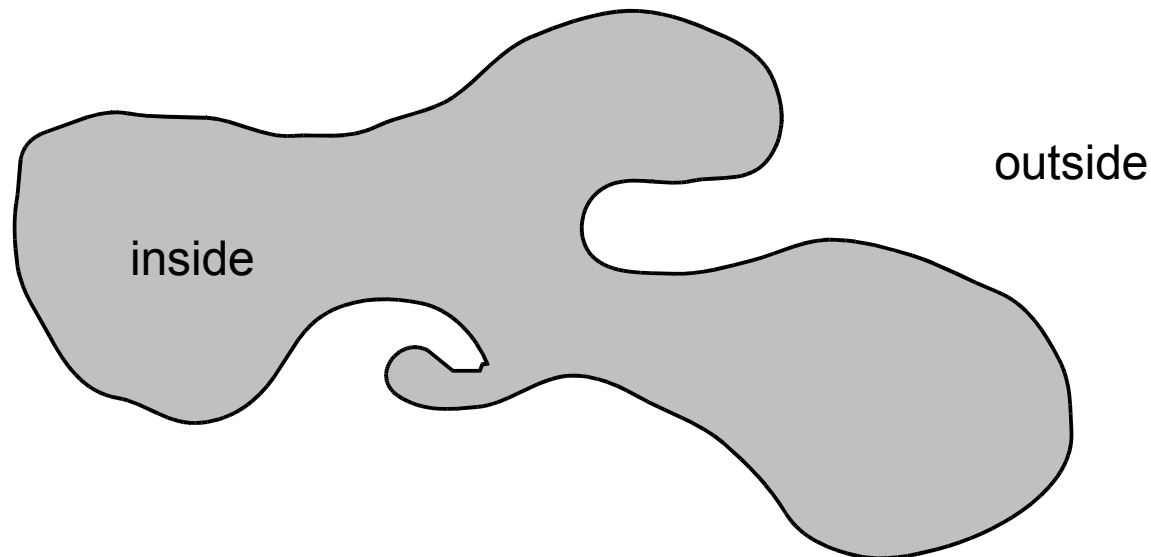
$$\mathbf{C}(S) := \sum_{v \in S} \mathbf{C}(v)$$

- A good separator will create two blocks of roughly equal cost.
- The specific case with vertex counts is obtained with equal cost vertices.

Basic Result 1

Jordan Curve Theorem

A simple closed curve in the plane divides it into exactly two connected regions – the “inside” and the “outside”.



Basic Result 2

Planar Graph Size

An n -vertex planar graph with $n \geq 3$ contains at most $3n - 6$ edges.

Implication: An iteration over the set of all edges takes $O(n)$ time.

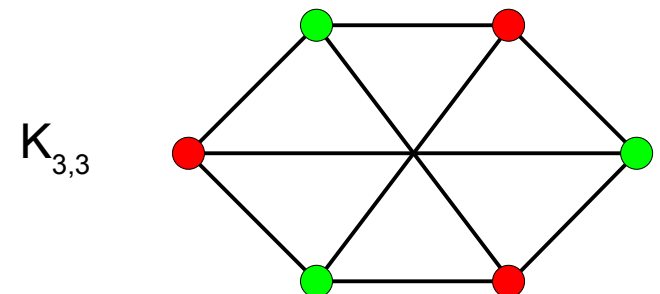
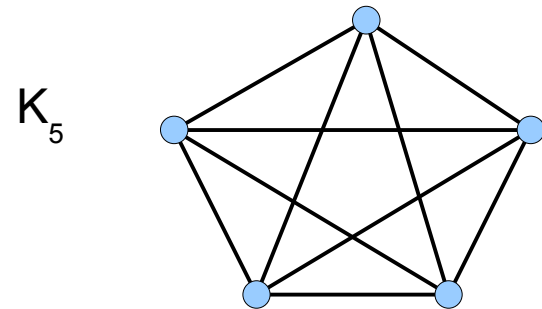
Basic Result 3

Kuratowski's Theorem

A graph is planar iff it contains neither K_5 or $K_{3,3}$ as a generalized subgraph.

Corollary

Shrinking an edge (and by induction any connected subgraph) of a planar graph to a single vertex preserves planarity.



Modus Operandi

1. **Partition Lemma**: show that a connected graph has a separator of size linear in radius of spanning tree.
2. **Levels Lemma**: construct a separator linear in sizes of two particular “levels” of the spanning tree, using (1).
3. **\sqrt{n} -Separator Theorem (\sqrt{n} -ST)**: show that the particular levels required in (2) exist and have combined size $O(\sqrt{n})$. Also extend the result to disconnected graphs.

Partition Lemma

Let G be any planar graph with nonnegative vertex costs summing to no more than 1.

Suppose G has a spanning tree of radius r . Then the vertices of G can be partitioned into three sets A , B , C such that

- no edge joins a vertex of A to a vertex of B
- neither A nor B has total cost $> 2/3$
- C has at most $2r + 1$ vertices including the root of the tree.

Proof of Partition Lemma

- If some vertex v has cost $> 1/3$, take $A = V - \{v\}$, $B = \emptyset$, $C = \{v\}$ and **we are done**.
- If all vertices have cost $\leq 1/3$:
 - Embed G in the plane
 - Add edges to form a triangulation.
 - **Any nontree edge e forms a simple cycle $\text{Cycle}(e)$ with some of the tree edges**, of length at most **$2r + 1$** (including root) or **$2r - 1$** .
 - The cycle splits the plane into the “inside” and the “outside” (Jordan). The total cost of vertices inside and outside $\text{Cycle}(e)$ are $C_i(e)$ and $C_o(e)$ respectively.

Proof of Partition Lemma

- **Balanced Split Lemma**

- At least one such cycle separates the graph so that neither the inside nor the outside contains vertices of total cost $> 2/3$.

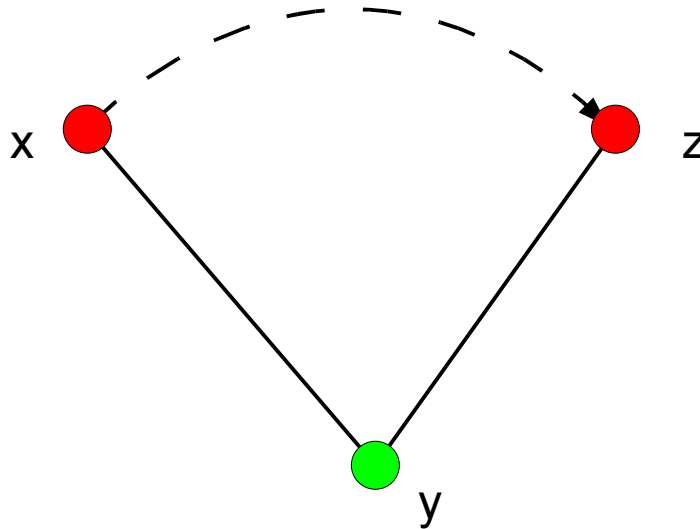
- **Proof**

- $(x, z) :=$ non-tree edge that **minimizes** the **maximum** cost inside or outside the cycle (“fairest split edge”). Break ties by choosing the edge whose cycle has fewest faces on the side of greater cost. Choose arbitrarily if ties remain.
- Assume, wlog, that $C_i(x, z) \geq C_o(x, z)$.
- If $C_i(x, z) \leq 2/3$, **we are done**.

Proof of Partition Lemma

– Proof of Balanced Split Lemma (contd)

- We'll show that $C_i(x, z) > 2/3$ is impossible by case analysis. Let y be the third vertex of the (triangular) face with edge (x, z) , lying inside $\text{Cycle}(x, z)$.
- Case 1:

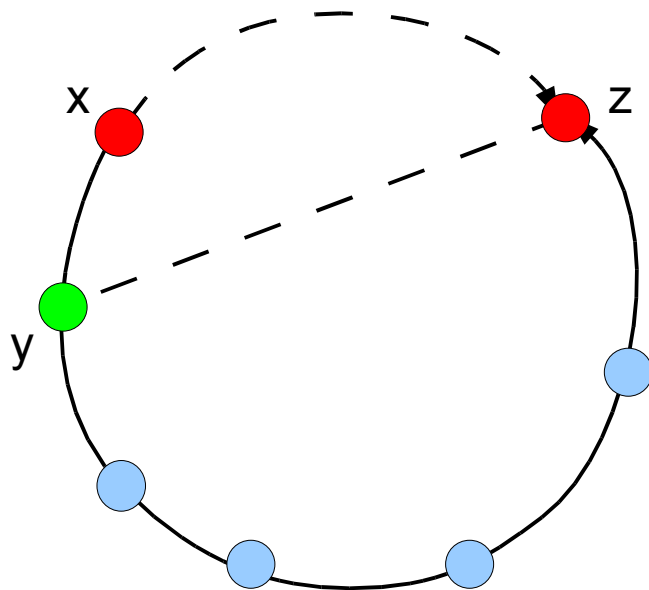


- The face *is* the cycle. Impossible, since vertices must lie in the cycle.

Proof of Partition Lemma

– Proof of Balanced Split Lemma (contd)

- Case 2:

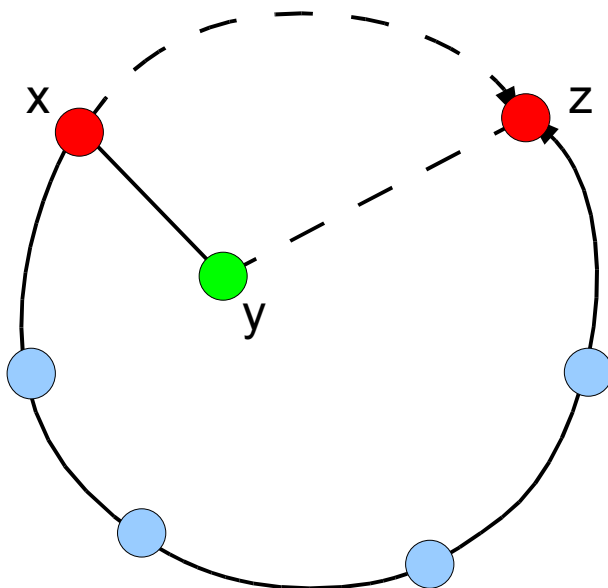


(y, z) is a non-tree edge defining a cycle containing the same vertices as $\text{Cycle}(x, z)$ but one less face. This contradicts our choice of (x, z) .

Proof of Partition Lemma

– Proof of Balanced Split Lemma (contd)

- Case 3:



- Cycle(y, z) has one less vertex and face than Cycle(x, z), and no greater cost.

- If $C_i(y, z) > C_o(y, z)$, we would have chosen (y, z) over (x, z).

- If $C_i(y, z) \leq C_o(y, z)$,

$$C_o(y, z) = C_o(x, z) + C(y)$$

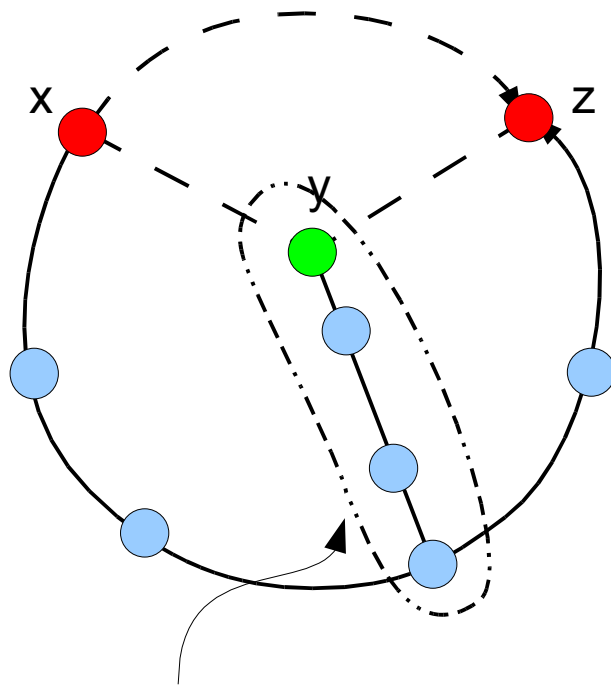
$$< 1/3 + 1/3 = 2/3 < C_i(x, z)$$

So again (y, z) would have been chosen over (x, z).

Proof of Partition Lemma

– Proof of Balanced Split Lemma (contd)

- Case 4:



may be single vertex y

- Every vertex inside $\text{Cycle}(x, z)$ is inside $\text{Cycle}(x, y)$, inside $\text{Cycle}(y, z)$, or on the boundary of both.
- Assume $C_i(x, y) \geq C_i(y, z)$.
- If $C_i(x, y) \geq C_o(x, y)$, we would have chosen (x, y) over (x, z) (fewer internal faces).
- If $C_i(x, y) < C_o(x, y)$,

$$C_i(x, y) + C(\text{Cycle}(x, y)) > C_i(x, z) / 2 > 1/3$$

$$\Rightarrow C_o(x, y) < 2/3 < C_i(x, z)$$
 So again (y, z) would have been chosen over (x, z) .

QED (Balanced Split Lemma)

Proof of Partition Lemma

- Now consider the cycle of the **Balanced Split Lemma**
 - $A :=$ vertices inside the cycle
 - $B :=$ vertices outside the cycle
 - $C :=$ vertices of the cycle

QED (Partition Lemma)

Levels Lemma

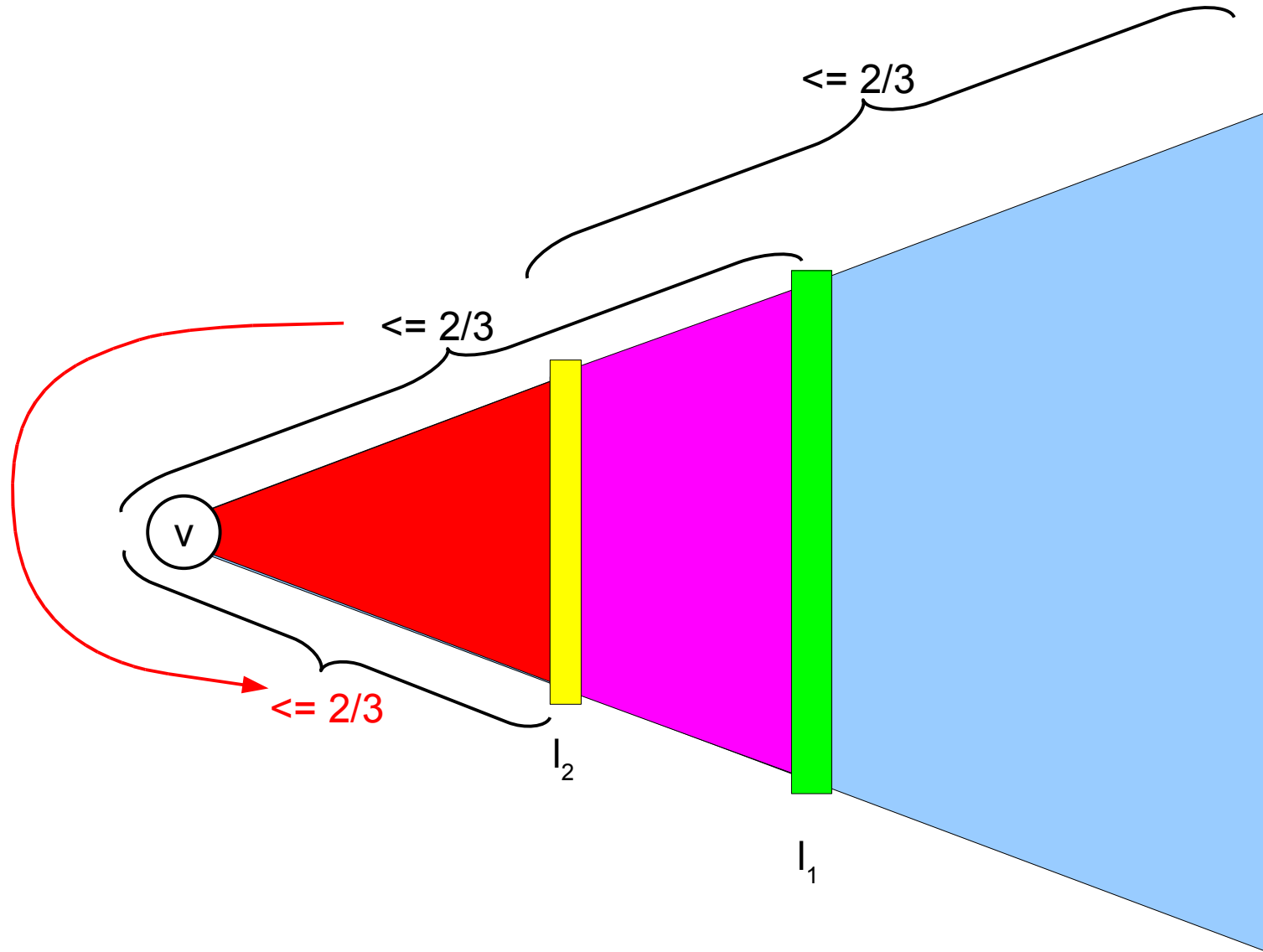
Let G be any n -vertex connected planar graph with nonnegative vertex costs totalling at most 1.

- Consider a bfs tree T of G with root v and radius r . T divides the $V(G)$ into “levels” based on distance from v (add an empty level $r + 1$). Let $L(i)$ be the number of vertices at level (distance) i .
- Given: l_1, l_2 s.t. levels 0 to $l_1 - 1$ have total cost $\leq 2/3$ and levels $l_2 + 1$ to $r + 1$ have total cost $\leq 2/3$.
- We can find a partition A, B, C of $V(G)$ s.t. no pair in $A \times B$ is adjacent, neither A or B has total cost $> 2/3$ and $|C| \leq L(l_1) + L(l_2) + \max\{0, 2(l_2 - l_1 - 1)\}$.

Proof of Levels Lemma

- **Note:** only vertices on the same level or on adjacent levels are adjacent.
- If $l_1 \geq l_2$,
 - $A :=$ all vertices on levels $0 \dots l_1 - 1$.
 - $B :=$ all vertices on levels $l_1 + 1 \dots r$ (subset of vertices on levels $l_2 + 1 \dots r$).
 - $C :=$ all vertices on level l_1 .
 - This defines the necessary partition and **we are done**.

Proof of Levels Lemma



Proof of Levels Lemma

- If $l_1 < l_2$,
 - $X :=$ vertices on levels $0 \dots l_1 - 1$ (cost $\leq 2/3$)
 - $L_1 :=$ vertices on level l_1
 - $Y :=$ vertices on levels $l_1 + 1 \dots l_2 - 1$ (cost = ?)
 - $L_2 :=$ vertices on level l_2
 - $Z :=$ vertices on levels $l_2 + 1 \dots r$ (cost $\leq 2/3$)
- If $C(Y) \leq 2/3$,
 - $A :=$ most costly of X, Y, Z (cost $\leq 2/3$)
 - $B :=$ other two parts (easy to show cost $\leq 2/3$)
 - $C := L_1 \cup L_2$, and **we are done**.

Proof of Levels Lemma

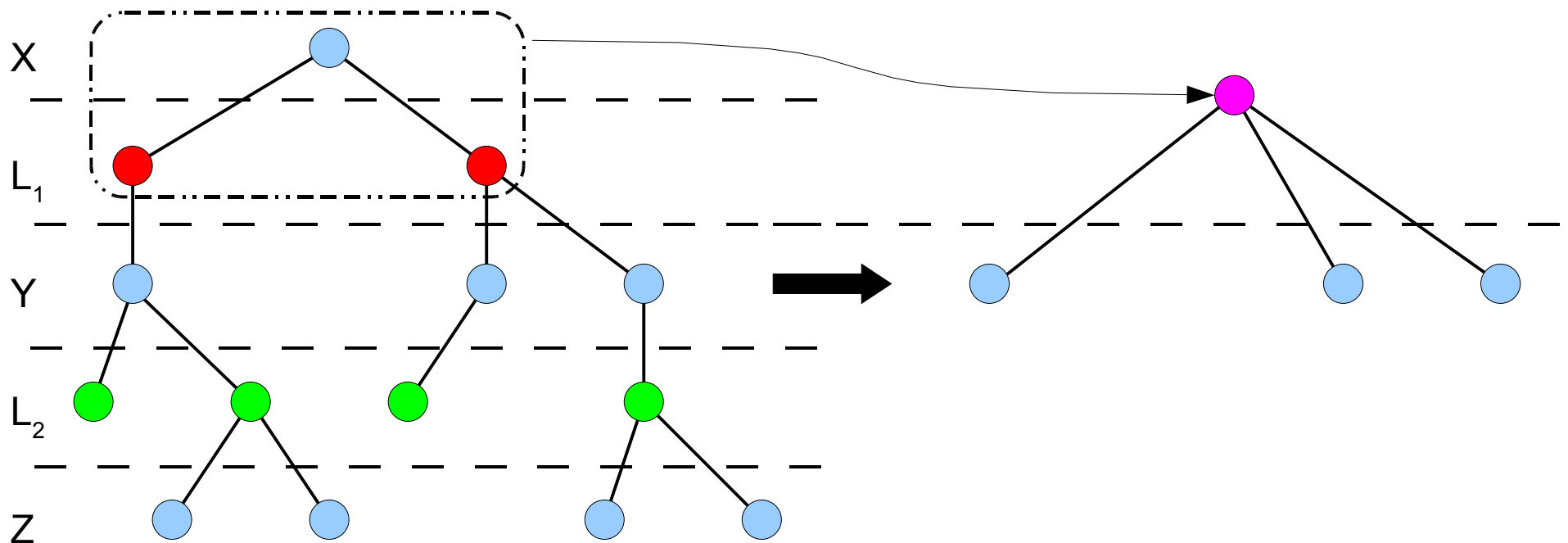
– If $C(Y) > 2/3$,

• Form a new graph G' as follows:

– Delete $Z \cup L_2$

– Shrink $X \cup L_1$ to a single vertex x of cost 0.

(these operations preserve planarity)



Proof of Levels Lemma

- G' has a spanning tree of radius $l_2 - l_1 - 1$ and root x .
- Apply the **Partition Lemma** to G' , and let A^* , B^* , C^* be the resulting partition.
 - $A :=$ set among A^* , B^* with greater cost
 - $C := (C^* - \{x\}) \cup L_1 \cup L_2$
 - $B :=$ remaining vertices of G
- Easy to verify that C separates A and B .
- By the Partition Lemma, $C(A) \leq 2/3$
- $C(A \cup C) \geq C(A \cup C^*) = C(A) + C(C^*) \geq C(Y) / 2 > 1/3$, therefore $C(B) = 1 - C(A \cup C) < 2/3$.
- $|C| \leq L(l_1) + L(l_2) + 2(l_2 - l_1 - 1)$

QED (Levels Lemma)

General \sqrt{n} -Separator Theorem

Let G be any n -vertex planar graph with nonnegative vertex costs totalling at most 1. The vertices of G can be partitioned into three sets A , B , C such that

- no edge joins a vertex in A to a vertex in B
- the total costs of A and B are each $< 2/3$
- C contains no more than $2\sqrt{2}\sqrt{n}$ vertices

Proof of Gen. \sqrt{n} -Separator Thm.

- If G is connected,
 - Assume levels $0 \dots r$ based on a bfs tree, add dummy levels -1 and $r + 1$.
 - $l_1 :=$ level s.t. total cost in levels $0 \dots l_1 - 1 < 1/2$ but total cost in levels $0 \dots l_1 \geq 1/2$.
 - If l_1 doesn't exist, the total cost of all vertices $< 1/2$, and $A = V(G)$, $B = C = \emptyset$ is a suitable partition.
 - $k :=$ number of vertices in levels $0 \dots l_1$.

Proof of Gen. \sqrt{n} -Separator Thm.

– $l_0 :=$ level s.t.

- $l_0 \leq l_1$ and $|L(l_0)| + 2(l_1 - l_0) \leq 2\sqrt{k}$

– $l_2 :=$ level s.t.

- $l_1 + 1 \leq l_2$ and $|L(l_2)| + 2(l_2 - l_1 - 1) \leq 2\sqrt{(n - k)}$

Proof of Gen. \sqrt{n} -Separator Thm.

– If l_0 does not exist,

- for $i \leq l_1$, $L(i) \geq 2\sqrt{k} - 2(l_1 - i)$

- $L(0) = 1$, $1 \geq 2\sqrt{k} - 2l_1$

$$\Rightarrow l_1 + 1/2 \geq \sqrt{k}$$

$$\Rightarrow l_1 = l_1 + \lfloor 1/2 \rfloor \geq \lfloor \sqrt{k} \rfloor$$

- $k = \sum_{i=0}^{l_1} L(i)$

$$\geq \sum_{i=l_1 - \lfloor \sqrt{k} \rfloor}^{l_1} 2\sqrt{k} - 2(l_1 - i)$$

$$\geq (4\sqrt{k} - 2\lfloor \sqrt{k} \rfloor) (\lfloor \sqrt{k} \rfloor + 1)$$

$$\geq \sqrt{k} (\lfloor \sqrt{k} \rfloor + 1) > k, \text{ which is a contradiction.}$$

– Similarly, l_2 must exist.

Proof of Gen. \sqrt{n} -Separator Thm.

- By **Levels Lemma**, $V(G)$ can be partitioned into sets A, B, C s.t. no pair in $A \times B$ is adjacent, neither A nor B has cost $> 2/3$, and C contains no more than $2(\sqrt{k} + \sqrt{(n - k)})$ vertices (from inequalities in choice of l_0 and l_2).
- $2(\sqrt{k} + \sqrt{(n - k)}) \leq 2(\sqrt{(n/2)} + \sqrt{(n/2)}) = 2\sqrt{2}\sqrt{n}$
- Hence **we are done** for connected graphs.

Proof of Gen. \sqrt{n} -Separator Thm.

- If G is not connected,
 - Let it have connected components G_1, \dots, G_k , with vertex sets V_1, \dots, V_k respectively.
 - If no G_i has total cost $> 1/3$,
 - Let i be min. index s.t. $C(V_1 \cup \dots \cup V_i) > 1/3$.
 - $A := V_1 \cup \dots \cup V_i$ (cost $< 2/3$ since $C(V_i) < 1/3$ and i is min. index)
 - $B := V_{i+1} \cup \dots \cup V_k$ (cost $< 2/3$ since $C(A) > 1/3$)
 - $C := \emptyset$

Proof of Gen. \sqrt{n} -Separator Thm.

- If some connected component, say G_i , has total cost in $[1/3, 2/3]$,
 - $A := V_i$
 - $B := V_1 \cup \dots \cup V_{i-1} \cup V_{i+1} \cup \dots \cup V_k$ (cost $< 2/3$ since $C(A) > 1/3$)
 - $C := \emptyset$

Proof of Gen. \sqrt{n} -Separator Thm.

- If some connected component, say G_i , has total cost $> 2/3$,
 - Apply argument for connected graphs to G_i to obtain partition A^*, B^*, C^* of $V(G_i)$.
 - $A :=$ set among A^*, B^* with greater cost
 - $C := C^*$
 - $B :=$ remaining vertices of G
 - Easy to show A and B have size $< 2/3$.

QED (General \sqrt{n} -Separator Theorem)

\sqrt{n} -Separator Theorem is special case of general result with all costs = $1/n$. QED (\sqrt{n} -Separator Theorem)

Partitioning Algorithm

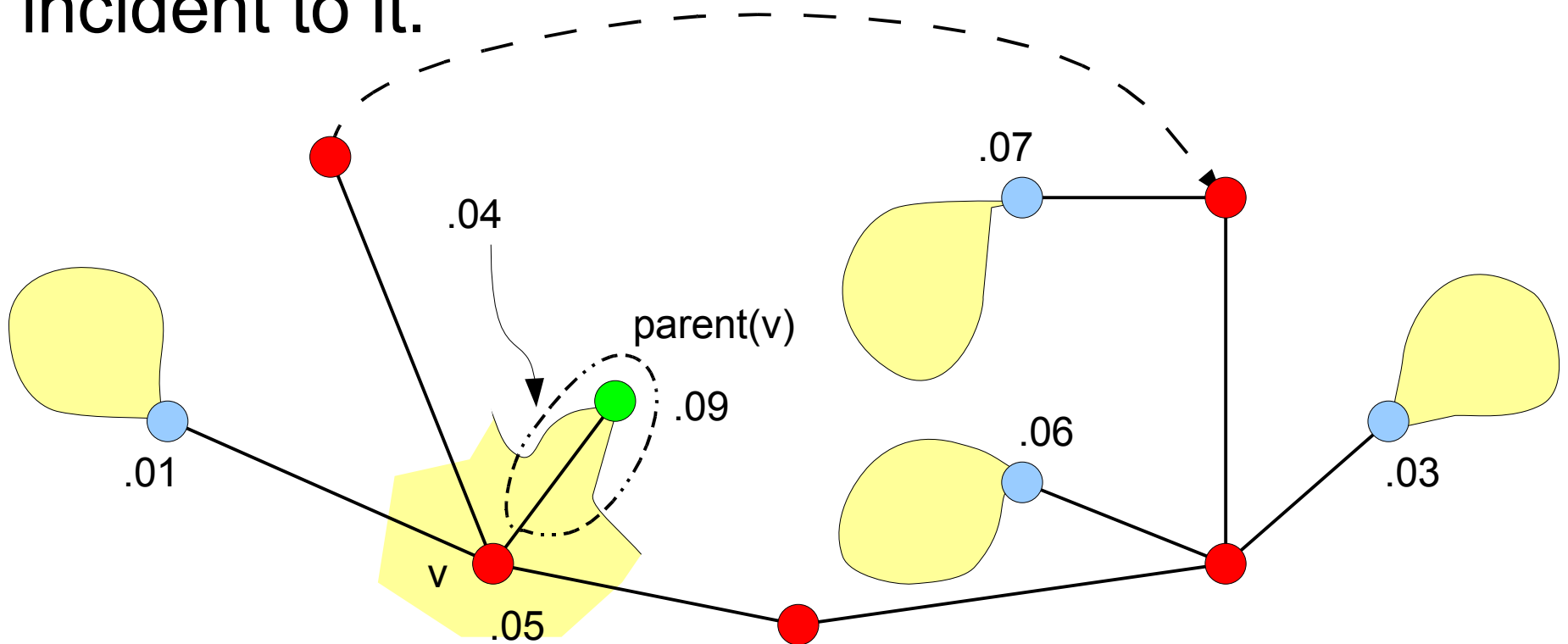
1. Find a planar embedding of G . [Hopcroft-Tarjan]
2. Find connected components of G (dfs) and determine cost of each one.
 - If no component has cost $> 2/3$, find partition (proof of General \sqrt{n} -Separator Theorem) and stop.
3. Find a bfs spanning tree T of the most costly component. Compute level of each vertex and number of vertices $L(i)$ in each level i .
4. Find l_1 and k . (proof of $G\sqrt{n}$ -ST)
5. Find highest l_0 and lowest l_2 . (proof of $G\sqrt{n}$ -ST)

Partitioning Algorithm

6. Shrink the graph to G' by eliminating vertices in levels above l_2 and collapsing levels l_0 and below to a single vertex x .
 - Scan levels 0 to l_0 using T , updating edges incident to these levels to connect to x .
7. Find a bfs spanning tree T' of G' (modify T).
 - At each vertex v , store parent and cost of subtree rooted at v .
 - Add edges to make G' a triangulation.

Partitioning Algorithm

8. (start of iteration) Pick a non-tree edge (v_1, w_1) and locate $\text{Cycle}(v_1, w_1)$. Compute total costs on both sides of cycle by scanning tree edges incident to it.



Partitioning Algorithm

9. (iteration step) Let (v_i, w_i) be current candidate edge, and call the side of $\text{Cycle}(v_i, w_i)$ with greater cost the “inside”.
- If $C_i(v_i, w_i) < 2/3$, stop.
 - Else
 - locate third vertex y of triangle inside the cycle with edge (v_i, w_i) .
 - If either (v_i, y) or (y, w_i) is a tree edge, let (v_{i+1}, w_{i+1}) be the non-tree edge. Compute cost inside it from $C(\text{Cycle}(v_i, w_i))$, $C(v_i)$, $C(w_i)$ and $C(y)$.

Partitioning Algorithm

- Else
 - Find tree path from y to closest parent z on $\text{Cycle}(v_i, w_i)$.
 - Compute cost on this path (excluding z).
 - Scan tree edges inside and incident to $\text{Cycle}(v_i, y)$ and $\text{Cycle}(y, w_i)$ **alternately**, accumulating subtree costs, until **one** of the subcycles has been covered.
 - Use previously computed costs to find cost of other cycle.
 - $(v_{i+1}, w_{i+1}) :=$ edge among (v_i, y) and (y, w_i) whose cycle has greater cost inside it.
- Repeat until a satisfactory edge is found.

10. Use cycle obtained in Step 9 and levels in Step 4 to construct a partition of the connected component ([proof of Levels Lemma](#)). Extend to entire graph ([proof of \$G\sqrt{n}\$ -ST](#)).

Construction Time Bound

- All steps easily shown to be $O(n)$ except Step 9, which requires some explanation:
 - **Proof** that Step 9 takes linear time
 - Each iteration of Step 9 deletes a face from the inside of the current cycle, so at most $O(n)$ iterations.
 - Each iteration takes $O(1) + O(\text{length of } y\text{-}z \text{ tree path}) + O(\# \text{ of edges scanned inside } (v_i, y) \text{ and } (y, w_i) \text{ subcycles})$.
 - Each vertex on $y\text{-}z$ tree path is inside the current cycle but outside the interior of all subsequent cycles.
 - One of every two edges scanned while accumulating subcycle costs is outside all subsequent cycles.
 - Hence time = $O(1) + O(\#\text{vertices}) + O(\#\text{edges}) = O(n)$.

Some Applications

Divide-and-conquer philosophy on planar graphs

- Approximation algorithms for NP-complete problems
 - Maximum Independent Set
- Embedding data structures
- Maximum matching

Maximum Independent Set

- **Problem:** Find the largest set of pairwise non-adjacent vertices in a graph.
- **Our special case:**
 - Planar graph
 - Approximate solution (error decreases as n increases)

Maximum Independent Set

- **Splitting Theorem:** Let G be an n -vertex planar graph with nonnegative vertex costs totalling at most 1 and let $0 \leq e \leq 1$. Then there is a set C of $O(\sqrt{n} / \sqrt{e})$ vertices whose removal leaves G with no connected component of cost $> e$.

The set C can be found in $O(n \log n)$ time.

Proof of Splitting Theorem

- **Proof:**

- If $e < 1/n$, $C = V(G)$, which obviously has size $O(\sqrt{n} / \sqrt{e}) = O(n)$.
- Else, recurse as follows:
 - Base: $C = \emptyset$
 - Induction:
 - If $G - C$ has no connected component of cost $> e$, **stop**.
 - Find a connected component K in $G - C$ of cost $> e$.
 - Apply **$G\sqrt{n}$ -ST** to K to obtain A^* , B^* , C^* .
 - Update $C \leftarrow C \cup C^*$.
 - *Level* of each component K found during recursion:
 - 0 if it exists when algorithm terminates
 - Else, $1 + \max$ level of children formed by splitting K (induction step).

Proof of Splitting Theorem

- **Note:** two components on the same level are vertex-disjoint.
- Level 1 component is split into level 0 components, so it has cost $> e$.
- In general, level i component has cost $> (3/2)^{i-1}e$.
- Total cost of $G \leq 1$, so number of components at each level $\leq (2/3)^{k-1} / e \leq n (3/2)^{k-1}$:

$$k \leq \log_{3/2} n + 1$$

- Time to split component = $O(\text{size of component})$, and they are vertex-disjoint, so running time = $O(\text{size of level} * \text{number of levels}) = \mathbf{O(n \log n)}$.

Proof of Splitting Theorem

– Bound on size of C:

- Let K_1, K_2, \dots, K_t be components, of sizes n_1, n_2, \dots, n_t respectively, at level i .
- # of vertices added to C by splitting K_i 's $\leq 2\sqrt{2} \sum_{j=1}^t \sqrt{n_j}$.
- $t \leq (2/3)^{i-1} / e$, $\sum_{j=1}^t n_j \leq n$, so a simple maximization gives
$$\sum_{j=1}^t \sqrt{n_j} \leq \sqrt{(nt)} \leq \sqrt{(n/e) / (2/3)^{(i-1)/2}}$$
- Summing over all levels, we have $|C| = O(\sqrt{n} / \sqrt{e})$.

QED (Splitting Theorem)

Maximum Independent Set

- **Algorithm**

- **Step 1:** Apply **Splitting Theorem** to G with $e = k(n)/n$ ($k(n)$ to be fixed later) and all vertex costs equal (to $1/n$). Yields C of size $O(n / \sqrt{k(n)})$ which partitions G into components of size $\leq k(N)$.
- **Step 2:** Find MIS in each component by brute force and take their union as the required approximation.

Maximum Independent Set

- **Approximation accuracy**

- Let I^* be an MIS of G . By separator property of C , I^* when restricted to a component left after removal of C is same as I when restricted to the same component.
 - Hence, $|I^*| - |I| = O(n / \sqrt{k(n)})$.
- G is planar and hence four-colourable, so
 - $|I^*| \geq n / 4$.
- Relative error $:= (|I^*| - |I|) / |I^*| = O(1 / \sqrt{k(n)})$.

Maximum Independent Set

- **Time Bound**

- **Step 1:** $O(n \log n)$ by [Splitting Theorem](#).
- **Step 2:** $O(n_i 2^{n_i})$ on a connected component of n vertices. So total time is

$$O\left(\max \left\{ \sum_{i=1}^n n_i 2^{n_i} \mid \sum n_i = n \text{ and } 0 \leq n_i \leq k(n) \right\}\right) \\ = O(n 2^{k(n)})$$

- Choose $k(n) = \log n$, get time $O(n^2)$, relative error $O(1 / \sqrt{\log n})$.
- Choose $k(n) = \log \log n$, get time $O(n \log n)$, relative error $O(1 / \sqrt{\log \log n})$.

Embedding of Data Structures

- Graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$
- An *embedding* of G_1 in G_2 is a one-one map $\varphi: V_1 \rightarrow V_2$.
- *Average proximity* of the embedding is
$$\sum \{d_2(\varphi(v), \varphi(w)) \mid \{v, w\} \in E_1\} / |E_1|$$
i.e. average increase in distance of adjacent vertices.

Embedding of Data Structures

- **Embedding Theorem:** Any planar graph of maximum degree k can be embedded in a binary tree with average proximity $O(k)$.
- **Proof:** Let G be an n -vertex planar graph. Embed G in binary tree T as follows:
 - If G has one vertex, $T := G$
 - Else, use \sqrt{n} -ST to split $V(G)$ into A, B, C .
 - $v :=$ any vertex in C (or in A if C is empty)
 - Embed subgraphs G_1 and G_2 induced by $A \cup C - \{v\}$ and B in binary trees T_1 and T_2 recursively.
 - $T :=$ tree with root v , subtrees T_1 and T_2

Embedding of Data Structures

- **Note:** T has exactly n vertices.
- $h(n) := \text{max depth of T}$
 - $h(n) \leq 100$ if $n < 100$
 - $h(n) \leq h(2n/3 + 2\sqrt{2}\sqrt{n} - 1)$ if $n \geq 100$
 $\leq h(29n / 30) + 1$
 - Solving, $h(n) = O(\log n)$.
- Define *total proximity function*
 - $s(G) := \sum \{d_2(\varphi(v), \varphi(w)) \mid \{v, w\} \in E\}$
 - $s(G) := 0$ if $n = 1$

Embedding of Data Structures

- Any edge in G not in G_1 or G_2 must be incident to a vertex of C (separator property).
- # of edges adjacent to vertices in $C \leq k |C|$.
- Any two vertices of the tree are at most $2h(n)$ distant from each other.
- So $s(G) \leq s(G_1) + s(G_2) + 2k |C| h(n)$ for $n > 1$.
- Let $s(n) = \max s(G)$ over all n -vertex graphs G .
 - $s(1) = 0$
 - $s(n) \leq \max \{ s(i) + s(n - i - 1) + ck\sqrt{n} \log n \mid n/3 - 2\sqrt{2}\sqrt{n} \leq i \leq 2n/3 + 2\sqrt{2}\sqrt{n} \}$
- Solving, $s(n) = O(kn)$.

Embedding of Data Structures

- If G is connected, average proximity = $O(kn) / n = O(k)$.
- If G is not connected,
 - Embed each component, of size n_i , separately.
 - Average proximity of each component = $O(kn_i) / n_i = O(k)$.
 - Since components are not connected, overall proximity = $\max_i \{ \text{avg. proximity of } i^{\text{th}} \text{ component} \} = O(k)$.

Embedding of Data Structures

- **Theorem (separate→embed):** Let S be any class of graphs of maximum degree k closed under the subgraph relation. Suppose S satisfies an $n / (\log n)^{2 + \epsilon}$ separator theorem. Then any graph in S can be embedded in a binary tree with $O(k)$ average proximity.
- **Theorem (embed→separate):** Let $G = (V, E)$ be any graph of n vertices and m edges which is embeddable in a binary tree T with average proximity p . Then V can be partitioned into three sets A, B, C s.t. $|A|, |B| \leq 2n / 3$, $|C| \leq cmp / \log n$ for some positive constant c , and no pair in $A \times B$ is adjacent.
- From a result of Erdos, Graham and Szemerédi and the second theorem, “almost all” graphs of cn edges (for a large enough c) require $\Omega(\log n)$ average proximity when embedded in binary trees.

Maximum Matching

- $G = (V, E)$ is an undirected graph.
- A *matching* M is a set of edges of G no two of which have a common endpoint.
- *Maximum cardinality matching*: $\max |M|$.
- *Maximum weight matching*:
 - assign real-valued weight to each edge
 - maximize sum of weights over M
- *Unmatched vertex*: not incident to M .

Maximum Matching

- $G = (V, E)$ is an undirected graph.
- A *matching* M is a set of edges of G no two of which have a common endpoint.
- *Maximum cardinality matching*: $\max |M|$.
- *Maximum weight matching*:
 - assign real-valued weight to each edge
 - maximize sum of weights over M
- *Unmatched vertex*: not incident to M .

Maximum Matching

- If P is an augmenting path w.r.t. M , $M \Delta P$ is a matching of greater weight.
 - Δ is the symmetric difference: $A \Delta B = A \cup B - (A \cap B)$.
- **Augmentation Theorem:** A matching in a graph has maximum weight iff there is no augmenting path.

Maximum Matching

- **Increment Lemma:** In weighted, undirected graph $G = (V, E)$, let $v \in V$, and let $G - v$ be the subgraph of G induced by $G - \{v\}$. Suppose M is a max weight matching in $G - v$.
 - If G has no augmenting path (w.r.t. M) with v as an endpoint, then M is a max weight matching of G .
 - Else, let P be the edge set of the augmenting path of max net weight. $M \Delta P$ is a max weight matching of G .
- Given M , a max weight matching in G can be found in time $O(m \log n)$ ($O(m)$ if max cardinality matching).
[\[Gabow\]](#)
- Planar graph has $|E| = O(n) \Rightarrow$ time $O(n \log n)$.

Maximum Matching

- **Algorithm**

1. If $|V| \leq 1$, return \emptyset

2. Else,

- Apply \sqrt{n} -ST to G to obtain A, B, C .

- Let G_A, G_B be the subgraphs of G induced by A and B .

- Apply algorithm recursively to find maximum weight matchings M_A and M_B in G_A and G_B respectively.

- $M := M_A \cup M_B, S := A \cup B$

3. Add C one vertex at a time to S , applying the **Increment Lemma** at each step to update the matching. Stop when $S = V$.

Maximum Matching

- **Correctness**

- Step 2 returns max weight matching of $G_{A \cup B'}$
- Step 3 extends it to max weight matching of G by Increment Lemma.

- **Time:** Let $t(n)$ be time taken on n -vertex graph.

- $t(1) = c_1$

- $t(n) \leq \max \{t(n_1) + t(n_2) + c_2 n^{3/2} \log n$
| $n_1 + n_2 \leq n; n_1, n_2 \leq 2n/3\}$ if $n > 1$

- since $|C| = O(\sqrt{n})$ and we apply the Increment Lemma to each vertex of C .

- Inductively, $t(n) = \mathbf{O(n^{3/2} \log n)}$ ($O(n^{3/2})$ for max cardinality matching.)