

Disclaimer: *These are rough notes with some exercises from the instructor, who provides them with few promises.*

The notes have been remembered or inspired by or taken from various sources. Apologies to those who were plagiarized and/or not cited. We are covering the Lipton-Tarjan theorem, and Leiserson's VLSI area result. Search the internet for citations for now.

2.1 Testing.

Question: Did you do the reading?

Hopefully.

Question: What is a DAG?

A directed acyclic graph.

Question: How do you topologically sort a DAG using DFS?

Inverse ordering of post ordering number.

Question: Why?

For any edge (u, v) , $post(u) > post(v)$. Either v is explored first, and is thus popped first. Or, u is explored first and is not popped until v is explored and popped.

Question: What is the analogy of this post number property for strongly connected components?

For any two components C and C' where there is an edge from C to C' the highest post numbered node of C is higher than the highest one in C'

Question: Where is the highest post number?

In a source component.

Question: Where is the highest post number, after deleting all the nodes in a source component?

In a source component of the remaining graph.

Question: How do you compute components strongly connected components?

Compute DFS in reverse graph, then compute DFS in decreasing order of post ordering, specifying a new component, whenever stack is empty.

Exercise 1: Problem 3.31. (d)-(h).

2.2 Separators.

Question: Direct edge toward bigger side (break ties by point toward node 1.) How many nodes have all the nodes directed inward?

Only one outedge from any node. If one is directed outward, it has more nodes than any other subtree and the edge from the subtree must be directed toward this node. If two places, then two outedges on path between those two places.

Question: Divide a tree into pieces where none is bigger than a constant by removing a node?

Do the previous. Remove the node with all edges directed inward. Every subtree has at most $n/2$ nodes.

Question: If the maximum degree of the tree is 3, how small can we make the largest connected component, after removing one edge?

The sink node has three incoming arcs. At least one of them has $x = \lceil (n-1)/3 \rceil$ nodes in it. Removing that edge leaves two components. One with x nodes and the other with two have at most $n-x \leq 2(n-1)/3 + 1$ nodes.

Question: Why would we want to divide a graph this way?

Recursive algorithms.

Question: How small should we make the pieces to have logarithmic depth?

Smaller by a constant factor.

Question: So, for convenience, is it reasonable to define the separator problem to finding a set of nodes whose removal leaves no piece that is larger than $2/3$ of the original graph?.

Sure.

Question: So trees have a size 1 separator? What about planar graphs?

Hmmm..

2.2.1 A lower bound.

Question: At least something?

Yes. At $\Omega(\sqrt{n})$. See a grid.

Question: Can you prove a grid has a large separator?

Hmmm...

Question: Can you prove the complete graph on n nodes (K_n) has a large separator?

Yes. At least $(n/3)(2n/3) = \Omega(n^2)$ edges must cross any $1/3 - 2/3$ partition of the vertices.

Question: Can you prove a grid has a large separator?

You asked that before.

Question: Oh, sorry. If a planar graph could efficiently communicate like the complete graph..hmmm. How could one map the complete graph into the grid?

Each node in the grid corresponds to a node complete graph to the grid by routing an edge from each (i, j) to each (i', j') in the grid using a path from (i, j) to (i', j) to (i', j') .

Question: Let's say the grid is $k \times k$, or that $n = k^2$. What is the most that any edges is used?

Now, each row edge supports at most n complete graph edges for each of k row nodes, or it is used $nk = n^{3/2}$. (Similarly for column edges.)

Question: Can you prove a grid has a large separator?

Any partition of the grid corresponds to a partition of the complete graph, thus the edges in the grid cut must contain at $\Omega(n^2)$ complete graph edges. Ah..I see.

Question: Can you do the calculation, now?

Since each edge is used by at most $O(n^{3/2})$ complete graph edges, there must be $\Omega(\sqrt{n})$ edges in the grid cut.

Question: How about a node separator?

Degree 4 nodes, so that can only help by a factor of four.

2.2.2 Algorithm for Planar graph.

Question: Do a breadth first search. What does a level of the Breadth first search tree do?

Hopefully. Given a node, all the other nodes are partitioned into sets V_1, V_2, \dots , such that edges go from V_i to V_{i+1} or between nodes in some V_i .

Question: Do you see a cut of size $O(\sqrt{n})$?

Any middle V_i . Is it balanced?

Question: Does this always work?

Maybe...but no... Binary tree. Level's are large.

Question: If we add edges and find a separator in the resulting graph, can we find a separator in the original graph?

Yes.

Question: Triangulate the planar graph. Take a breadth first spanning tree of the planar graph. Say it has depth r . What does each non-tree edge induce in the tree?

A cycle of length $2r + 1$. A set of nodes that cuts apart the graph, if there are nodes on both sides of this Jordan curve.

Question: Consider the dual. Recall that the non-tree edges, form a spanning tree of the dual. Direct the edges in the dual toward the side of the corresponding Jordan curve that has more nodes. Is there an edge where the number of nodes on one side or the other is at most $2n/3$?

Yes. This is the same argument as the tree argument.

Question: So, for a planar graph with diameter r , there is a $O(r)$ sized separator?

Yes. Lets call this a lemma.

Lemma 2.1 *For a planar graph with diameter r , there is a $O(r)$ sized separator?*

Question: Are we done? Why?

No. Consider a planar graph with higher than \sqrt{n} diameter.

Question: What about BFS? If the diameter of the graph is $\geq l = \Omega(\sqrt{n})$, what is average size of a level in the breadth first search tree?

$n/l = O(\sqrt{n})$.

Question: Can we always find a level that splits evenly?

No. Not necessarily. Consider a long line and then leaf to leaf binary trees and then a long line.

Question: In this case, what is a good cut?

Two cuts isolating the bulk of the graph, and then the graph in the middle has low diameter, and we can use Lemma 2.1 for the middle.

Question: What if the graph is more complicated, say a reasonably thin grid, a fat part, and then a thin grid. Do you see a cut?

Two levels of the BFS. Plus two paths across.

Question: Can you find the paths using Lemma 2.1?

Yes. Find the BFS levels. Add a new exterior node, with edges to nodes in highest level. Now, it is a low depth planar graph. Apply lemma 2.1.

Note, this is an algorithm that given two BFS levels produces a cut consisting of the two levels plus a set of nodes of size at most $i + j$.

It divides the graph into the pieces consisting of the nodes before the first BFS level, after the last, and divides the middle set into pieces where each contains at most $2/3$ of the nodes in the middle levels.

Let's call this the **middle cut algorithm**.

Question: What is the size of the cut?

Say the BFS levels are V_i and V_j . At most $|V_i| + |V_j| + 2|i + j| + 1$.

Question: Can we make it balanced?

If every piece is at most $2n/3$ nodes. Then, sort the pieces according to size, and some prefix should work.

Question: Can we choose levels such that $|V_i| + |V_j| + 2|i + j| = O(\sqrt{n})$ and the pieces will remain small.

Start at median level k , i.e., fewer than $n/2$ nodes before and after. Choose V_i to be the deepest level below k with fewer than \sqrt{n} nodes. Choose V_j to be the shallowest level below k with fewer than \sqrt{n} nodes. So, we have $|V_i| + |V_j| = O(\sqrt{n})$.

Question: Is $|j - i| = O(\sqrt{n})$?

Yes, otherwise we run out of nodes.

Question: If we run the middle cut algorithm, are we done?

The number of nodes in the cut is $O(\sqrt{n})$. The size of each piece is at most $2n/3$. So, we now have a $O(\sqrt{n})$ sized separator.

Exercise 2: Do one of the following.

Exercise 2.1: Give an algorithm with separator size $2\sqrt{2n}$

Exercise 2.2: Give an algorithm that finds a set of nodes of size $O(\sqrt{n})$ whose removal leaves a set of connected components that can be divided into two sets where each has no more than $n/2$ nodes in it.

2.2.3 VLSI layout.

Question: Say you have a graph of degree four. How would one lay it out as a circuit?

A grid. Each node is put on the plane. Each edge is mapped to a path in the grid. No paths should use the same edge. This is called the “Thompson grid model.”

Question: Ugh. Can you show me how to route say K_4 ?

We’ll do something called slicing to add each edge. Add a row next to one endpoint and a column next to another. Then, use the row-column routing.

Question: Can you do this for any graph?

Sure.

Question: How big is the grid?

Each edge adds a row and column, so it is an $|E| \times |E|$ grid, and has $O(|E|^2) = O(n^2)$ area.

Question: Can you do better with a planar graph. Does the planar embedding have better layout?

No. Think of a set of nested diamonds. The area is $O(n^2)$?

Question: What about divide and conquer?

Ok. Divide the planar graph into pieces of size $n/4$ using an $O(\sqrt{n})$ node separator. Recursively, layout each piece.

Question: How do you put them together?

Arrange the recursive pieces in a two by two square, with the separator nodes in a square in the middle. Add the edges back using slicing.

Question: What is the recurrence for this? Perhaps, make it for the height, rather than the area.

$$H(n) = 2H(n/4) + O(\sqrt{n}).$$

Question: What is the solution of this recurrence?

$$H(n) = O(\sqrt{n} \log n).$$

Question: Why?

Recursive tree has depth $\log n$. Each level has cost \sqrt{n} . (Top is \sqrt{n} , next is $2\sqrt{n/4} = \sqrt{n}$, next has

$4\sqrt{n/16} = \sqrt{n}$...) Master's theorem also works here.

Question: What is the area?

Same recurrence for width, and thus we get $H(n)^2 = O(n \log^2 n)$.

Question: Recall the other algorithm, are we better?

Yes, much better. It produced $O(n^2)$ area layouts, where as this is nearly linear in the number of nodes.