3/16/98 day 18 “We should stop.”

1 Max flow

1.1 definitions

- Maximal flow = a flow which can't be improved without backtracking.
- Maximum flow = a flow which can't be improved.

1.2 Level graph

A level graph is one which can have all of it's nodes arranged in levels s.t. nodes at level $l+1$ don't connect to nodes in level l. A level graph is a DAG (directed acyclic graph).

1.3 Dinic

To find the maximal flow (phase) do a depth first search through the DAG. Dinic came up with an $O(mn)$ per phase algorithm. $\#phases \leq n$.

1.4 MPM algorithm $O(n^3)$

- $c(v) = min\{\sum_{u \in V} c(u,v), \sum_{u \in V} c(v,u)\}$ = the capacity of a vertex.

Within a phase, saturate a node, then push the flow into the other nodes.

1.4.1 code

1. find min capacity vertex
2. if capacity(v) = u then delete v
3. push flow to sink and to source.

1.4.2 data structure

We meed a data structure to operate with:

	$\#/phase$	$cost$	$total$
$delete$	n	$log(n)$	$n * log(n)$
min	n	1	n
$decrement$	n^2	1	n^2

Actually, arrays work just as well

1.4.3 timing analyses

If an edges is saturated associate the cost of saturation with the edge. Since there are m edges which will only saturate at most once, the cost of all saturated edges $\leq m \leq n^2 \leq n^3$.

2 Example problems

2.1 Maximum matching problem

There are two groups 'boys' and 'girls'. Each boy-girl pair has some 'compatibility rating'. The goal is to match boys to girls, maximizing overall 'compatibility'. The problem can be turned into a maximum flow problem. Make a source connect to all boys, the boys connect to the girls, and the girls connect to the sink. All edges have capacity 1, which means each 'boy' will connect to 1 'girl'.

2.1.1 Maximum matching

The algorithm can be specialized to $O(m\sqrt{n})$.

- Cost per phase is $O(m)$
- #phases is $O(\sqrt{n})$

Let M = matches, M^* = maximal match.

$M \bigoplus M^* = (M - M^*) \cup (M^* - M)$ = an augmenting path.

All the augmenting paths are disjoint in the maximum matching problem. Do depth first search. When you encounter the sink, pop the stack and remove the searched edges then start over. Each edges is only searched over once so the cost per phase is $O(m)$. With $\sqrt{n}$ phases completed, $\#levels \geq \sqrt{n}$. $M \bigoplus M^* =$ K vertex disjoint paths from s to t. Length $\geq \sqrt{n}$ which implies that $K \leq \sqrt{n}$.

2.2 Maximum number of disjoint paths

Find the maximum number of edges disjoint paths from a source to a sink. Setup each edge in the graph to have capacity one and solve for the max flow, which will be composed of some number of disjoint paths.

2.3 Maximum number of vertex disjoint paths

Find the maximum number of vertex disjoint paths. Split all vertices into 2 nodes. The first node has all the incoming edges and connects to the other node. The other node has all the outgoing edges of the original vertex. Now, any flow must go between the 2 nodes and only one flow can go between them which means flows are vertex disjoint.