11 April 2018 PRELIM 2 REVIEW.

Prelim 2 is 7:30–9:00pm tomorrow (April 12) night. Stay tuned (Piazza + email) for room assignments.

Review sheet (review2.pdf) on Piazza covers most of this material.

Prelim covers: Ch 5 (Divide & Conquer)
Ch 7 (Flow)
Ch 8 (NP-completeness)
Coverage of 7 & 8 will greater than 5.

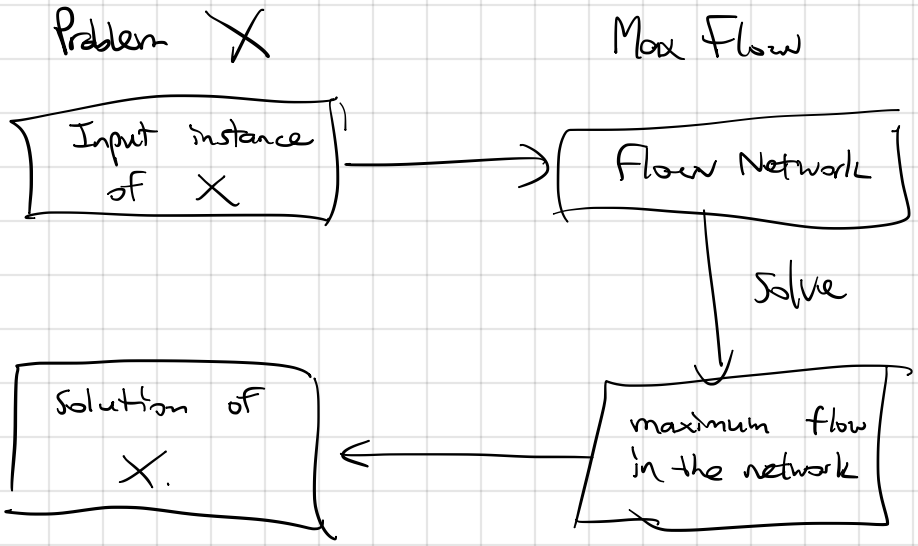
Div & Conq. Algorithm will be given to you. Just analyze it. (Correctness, running time)
Correctness proofs are always strong induction on size of input.
Running time analysis always boils down to setting up and solving a recurrence such as
$$T(n) = T(\tfrac{n}{2}) + O(n) \Longrightarrow T(n) = O(n).$$
(reviewing & memorizing the "Master Theorem" may help.)
Don't bother memorizing Ch. 5 algorithms and learning to run them step by step.

Flow. Knowledge of basic notions
flows, cuts, augmenting paths, residual graphs
Ford-Fulkerson algorithm
Max-flow min-cut theorem.
Running times of flow algs...
n = # of vertices,
m = # of edges, $n = O(m)$
C = combined capacity of edges leaving s
(all capacities are integers)

Ford-Fulkerson	$O(mC)$
Edmonds-Karp #1	$O(m^2 \log n \log C)$
Edmonds-Karp #2	$O(m^2 n)$
Preflow-Push (§7.4)	$O(n^3)$

} special cases of Ford-Fulkerson

Flow reductions. If Prelim 2 asks you to design an algorithm, it's probably asking you to do this.



Describing such an algorithm boils down to describing $\longrightarrow$ and $\longleftarrow$.

When you create the network, what are its vertices and edges? What are their capacities?

When you translate a max-flow into a solution of X, what is the translation? I.e. how should flow values on edges be interpreted as a solution of X?

This is the step most often skipped in incomplete solutions.

Running time: incorporates the running time of all 3 steps. Almost always, solving max-flow will be the slowest step.

Most common mistake in this step: forgetting to substitute the parameters that describe the size of the input to problem X.

E.g. Problem X deals with assigning p profs to c committees.

You create flow network with $n = p \cdot c$ $m = 6 \cdot p \cdot c$ and solve using E.K. #2.

Don't write running time as $O(m^2 n)$, but as $O(p^3 c^3)$.

Proof of correctness. Show a bidirectional transformation $\{\text{solutions of } X\} \longleftrightarrow \{\text{integer flows in } G(X)\}$.

NP-Completeness. Typical 5-step solution.

[1] verifier. (show membership in NP)
[2] reduce from some other problem that is known to be NP-Complete (e.g. 3SAT) to Problem X.
[3] reduction is poly time.
[4] Answer to 3SAT is yes $\Longrightarrow$ Answer to X is yes.
[5] Answer to X is yes $\Longrightarrow$ Answer to 3SAT is yes.

Can be very brief. Just explain how the verifier works.

Can be very brief.

Potential starting points for NPC reductions...

3SAT
IND SET
VTX COV
CLIQUE
HAM CYCLE
SUBSET SUM

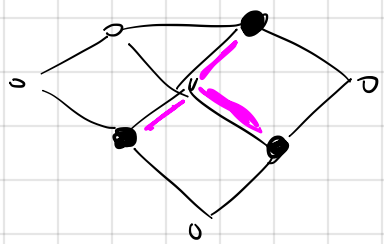
Also eligible: SET COVER, SET PACKING, 3-COLORING, 3-D MATCHING, TRAVELING SALESMAN, KNAPSACK.

NP-Completeness simple: Graphical Steiner Tree.

Input: Undir graph G,
Subset of vertices, K.
Positive #, t.

Question: Does G contain a subtree with $\leq t$ edges such that every vertex in K belongs to subtree?

E.g.

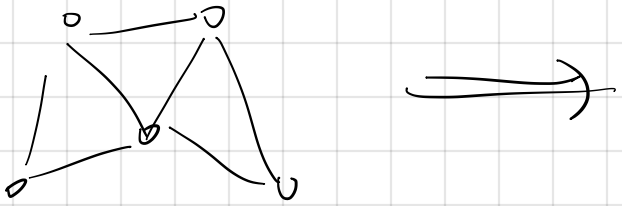


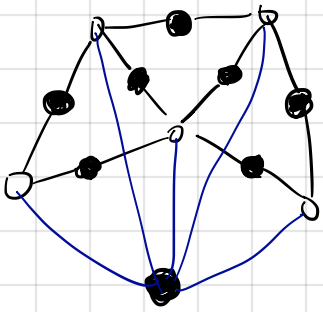
K = black nodes
$t = 3$.
Answer: YES.

In vertex cover:
Choose vertex set ~~> Covers every edge

In Steiner tree
Choose tree ~~> Connect every node of K.

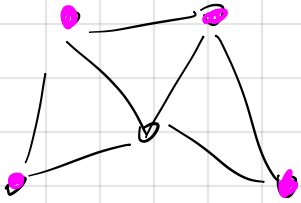
Vertex Cover Problem

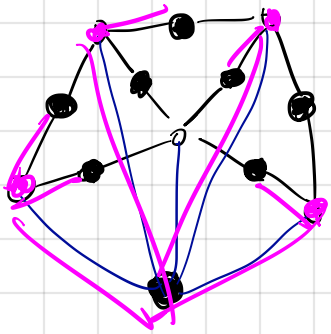




K = black nodes

Vertex set that covers every edge ⟹ Steiner tree





Size s ⟹ tree with s+m edges.

- from root
- to the remaining m elements of K.

... therefore reduction should set $t = k+m$.

The diagram above shows that a "yes" instance of VERTEX COVER transform to a "yes" instance of GRAPHICAL STEINER TREE.

To prove the converse: a subtree T with $t = k+m$ edges has $k+m+1$ vertices. If K is a subset of those vertices, since $|K| = m+1$, there must be k other vertices in the tree. The corresponding vertices of G must cover every edge, because for every black node corresponding to an edge e, there must be an edge of T that contains that black node. The other endpoint of that edge of T is a white node corresponding to a vertex of G which is one of the endpoints of e. That vertex is included in our vertex cover, so edge e is covered.