CS3230 Finals Notes

Chapter 7 - Network Flow

Define a s-t cut to be a partition (A,B) with the source in A and the sink in B. The capacity of the cut will be the **sum of the capacities of all the edges out of A**.

Flow value lemma: If (A,B) is an s-t cut, then the net flow of the cut is equal to the amount leaving s, which is equal to the sum of the flow out of A - sum of the flow into A.

Weak duality: The value of the flow across a s-t cut is at most equal to the capacity of the cut.

If f is any flow and (A,B) is any cut, if v(f) = cap(A,B) then f is a max flow and (A, B) is a min cut.

FordFulkerson Algorithm: Add a flow, then reverse the edge, until it is not possible to add any flow into the graph anymore. It will output the maxflow. FF is not a polynomial time algo since it can potentially depend on the size of the input.

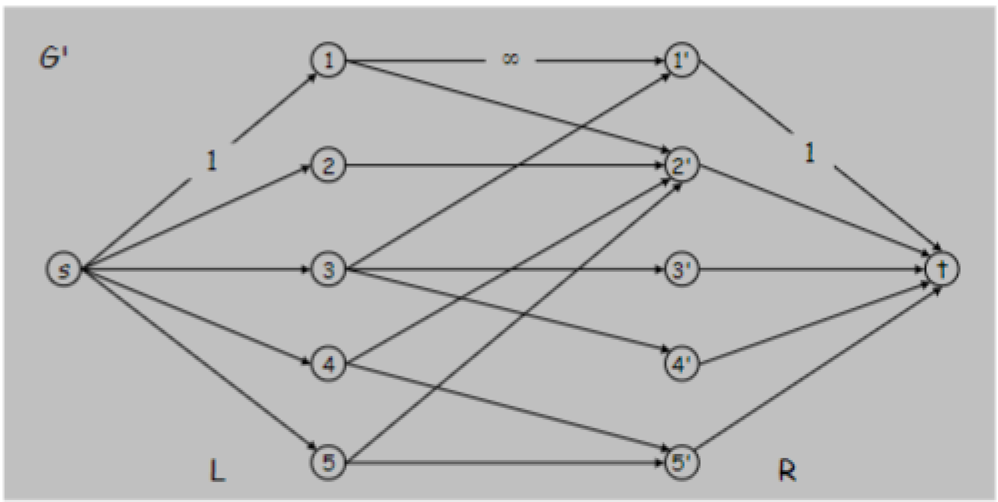
The algorithm terminates at at most nC iterations, if each augmentation increases the maxflow value by at least 1. O(mn) time.

Saturated edge: when the flow through the edge is equivalent to its capacity, and you cannot push anymore flow.

Modelling bipartite (max cardinality) matching as a maxflow problem

Max flow formulation.

- Create digraph $G' = (L \cup R \cup \{s, t\}, E')$.
- Direct all edges from L to R, and assign infinite (or unit) capacity.
- Add source s, and unit capacity edges from s to each node in L.
- Add sink t, and unit capacity edges from each node in R to t.



Max cardinality matching in G = value of max flow in G′

(since each node in L and R participate in at most one edge in M, the set of edges with flow f = 1 after running the Ford Fulkerson algo)

If a perfect matching exists, then the size of set R, |R| is larger than or equal to |L|, the size of the set L. By marriage theorem, iff for any subset S of L, the size of N(S) is larger or equal to S.

Maxflow Applications

We can model problems, by adding an edge between L and R nodes if the path between them in the problem satisfies a certain requirement (Lpath < N).

Edge Disjoint Path

If we set the capacity of every edge to unit capacity, then the max disjoint path will be equivalent to the max flow.

Chapter 8 - NP Completeness

An algorithm is said to run in polynomial time if the runtime is polynomial in **length of input.** But it depends on what n means in the context of the problem!

i.e. if we want to check if an element 'a' is the largest element in a n sized array of n elements. then length of input is **n, number of elements.**

p refers to the set of all decision problems with a polynomial runtime.

Polynomial time reductions

- Let A and B be decision problems
- Definition: We say that $A \leq_p B$ to mean that there exists a polynomial time algorithm which does the following:
  - Input: a, an instance of A
  - Output: b, an instance of B
  - The correct answer to a and b are the same (i.e. both yes or both no)
- Theorem: if B in P, and $A \leq_p B$, then A in P
  - Given an instance of $a \in A$, we can generate $b \in B$, solve b and then use that as our answer to a

If A reduces to B, that means if B is solvable in polynomial time, then A is also solvable in polynomial time (since we can just get the answer in B and it will be the answer for A)

Contrapositive: if A reduces to B, if A is NP hard it simply means that B is NP hard as well (since if there was a solution to B there would have been a solution to A).

BUT if A reduces to B and A is in p, does not mean that B is in p!! It could be possible that B is not solvable in polynomial time but A is.2

np refers to the class of decision problems that can be verified in polynomial time with an appropriate certificate.

Examples of these problems: Does there exist a ....

if yes, then show the example (explicitly) but if no then it is impossible to fool the algorithm

Not an example: Find a example of .... , because there are potentially infinite configurations where you can check and it is not possible to verify all of them

a problem b is np complete if

- for all A in NP, A reduces to B
- B is in np

to show that a problem b is np complete, we find a np complete problem a and show that a reduces to b

Chapter 11 - Approximation Algorithms

A p-approximation algorithm is an algorithm that:

- Is guaranteed to run in poly time
- Guaranteed to solve an arbitrary instance of the problem
- Guaranteed to find a solution within ratio **p** of true optimum

Chapter 12 - Randomized Algorithms

Expectation = Summation of the (probability that X = j) * j, for all possible values of X

Just see notes lol