CS218 Design and analysis of algorithms **Jan-Apr 2023**

Exercises: Network Flow (No submission)

1. For an s-t path, the bottleneck is defined to be the least capacity of an edge on that path. Can you design an efficient algorithm to find the s-t path with largest bottleneck? Or try this variant: given a threshold λ, is there an s-t path where every edge has capacity at least λ?

2. Construct a flow network with irrational capacities, where it is possible that the maximum flow algorithm takes infinite iterations. Note that there can be a clever way of choosing s-t paths. Here we are just saying that there is a possibility of choosing paths that leads to infinite iterations.

3. Try to design a scaling version of the max flow algorithm, that is, where the number of iterations is proportional to $\log(\sum_e c_e)$.

   Hint: take a parameter Δ, which will initially be large and will become smaller over time. First, ignore all the edges whose capacity is less than Δ, and run the max flow algorithm. When it is no longer possible to increase the flow, then reduce the value of Δ appropriately and repeat.

4. Use max flow algorithm to solve the following problem. Given a graph with a source vertex s and a destination vertex t, find the minimum number of vertices which can be removed to make s and t disconnected.

5. TA allocation: for each TA, we are given a list of suitable courses. For each course we are given the required number of TAs. We want to check whether a TA allocation is possible meeting all course demands, where a TA is only assigned to one of their suitable courses. Design an algorithm for this problem using network flow.

   Use max flow min cut theorem to prove the following: if the above TA allocation is not possible then there must be a subset S of courses whose total demand is larger than the total number suitable TAs for them. This is also known as Hall's theorem.

6. Recall the algorithm we discussed for partitioning a poset (partially ordered set) into minimum number of chains. The algorithm went via the bipartite matching algorithm. First, for the given poset P on n elements, we construct a bipartite graph G_P on $n+n$ vertices, say $\{\ell_1, \ell_2, \ldots, \ell_n\}$ and $\{r_1, r_2, \ldots, r_n\}$. We have an edge (r_i, ℓ_j) in G_P if and only if $i \leq j$ in the poset.

   - Prove that from a matching of size $n-k$ in G_P, we can get a partition of P into k chains. What happens with the unmatched vertices.
   - Prove that for every partition of P into k chains, there is a corresponding matching of size $n-k$ in G_P.
   - Prove the optimality of the obtained partition into chains.

7. Use the max flow min cut theorem to prove Dilworth's theorem. That is, for any partially ordered set, the minimum number of chains in which we can partition the set is equal to the maximum antichain (i.e., maximum number of mutually incomparable elements). You can use the above reduction to bipartite matching and then a reduction to max flow. Assume max flow is equal to s-t-minimum-cut. Then given an s-t cut with outgoing capacity of $n-k$, try to construct an antichain of size k.

8. **Project selection** We are given a set of projects, where some projects are pre-requisites for others. This can be represented by a direct acylic graph on projects. An edge from i to j represents that i is a pre-requisite for j. Clearly this is a transitive relation. A project can be done only if all its pre-requisite projects have been done.

   Some projects might have a positive value, i.e., you gain a net profit from them. While some other projects may have a negative value, i.e., you have to invest more into them than what you gain from them. The goal is to select a subset of projects which maximizes the total value, under the constraint that if a project is selected then all its pre-requisites must also be selected.

Reduce this problem to s-t-minimum-cut problem. That is design an algorithm which can use s-t-minimum-cut subroutine. Note that minimum cut is also a subset (of vertices) selection problem.

9. Tiger counting: The tiger population in the country is counted once in every four years. To count their number, cameras are installed in the areas visited by them. Tigers are uniquely identified by their stripes. Interestingly, the stripes on the two sides of a tiger are not correlated. Hence, they always install cameras in pairs, where the two cameras face each other. This way they can get pictures from both the sides of the tigers. Sometimes there are obstacles or some malfunctioning, due to which they get only one side of the tiger. Suppose they get (pairs of) images where they have both sides of k distinct tigers. And additionally, they have some images of the left stripes of m distinct tigers and some images of the right stripes of n distinct tigers. When there is no way to figure out which of the left images and right images belong to the same tiger, they take a conservative estimate, which is $k + \max\{m, n\}$.

   At some point, allegedly to show a higher count, they changed the protocol and counted the above scenario as $k + m + n$ distinct tigers.

   Let's say there are some experts who can identify which pairs of left and right images are definitely from distinct tigers. For example, for $m = 3$, $n = 3$, suppose they say that the pairs (ℓ_1, r_2), (ℓ_1, r_3), (ℓ_2, r_2), (ℓ_2, r_3) are definitely from distinct tigers. With this additional information, the conservative estimate will be 4 distinct tigers. How will you find this conservative estimate for any given list of pairs from experts.