CS 170 HW 11

Due **2020-04-13, at 9:59 pm**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".

2 Opting for releasing your solutions

We are considering releasing a subset of homework submissions written by students for students to see what a full score submission looks like. If your homework solutions are well written, we may consider releasing your solution. If you wish that your solutions not be released, please respond to this question with a "No, do not release any submission to any problems". Otherwise, say "Yes, you may release any of my submissions to any problems".

3 Project Pregame

Please refer to the project spec to answer these questions.

(a) Respond to the following logistics questions.

(i) How many members from each group should fill out the Phase 0 group sign-up form?

(ii) Under what circumstances are you allowed to make changes to your group after April 12th?

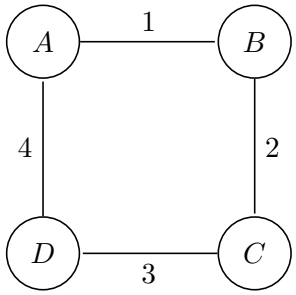
(iii) When are the Phase 1 and Phase 2 due dates?

(iv) What are the deliverables in Phase 1?

(v) What are the deliverables in Phase 2?

(b) If the input graph is **complete**, what is the cost (average pairwise distance between all pairs of vertices) of the optimal cell tower network?

(c) What is the cost of the optimal network for the following graph?



(d) Give an example of a graph for which the **optimal** network T has vertex $v \in T$ whose removal keeps the resulting sub-graph a **valid** network.

(e) Let T be a line graph on three vertices with both edge weights equal to 1. What is the cost of T?

(f) Let T be a line graph on $n \geq 2$ vertices with all edge weights equal to 1. Prove that the cost of T is $\frac{n+1}{3}$.
(Hint: induction!)

4 Online Matching

Consider the maximum bipartite matching problem on a unweighted, undirected bipartite graph $G = (L, R, E)$, where L and R denote the left and right side vertices.

In class, we saw that the problem can be solved via max flow when the graph is given to you. However, consider the following harder, *online* setting. In the online version, L is known, but the vertices in R arrive one at a time. When a vertex $u \in R$ arrives, its incident edges are also revealed, and at this moment, we must make a immediate and irrevocable decision to match u to one of its available neighbors in L or not to match it at all. The decision is irrevocable in the sense that once a vertex is matched or left unmatched, it cannot be later rematched (to another vertex). The goal is to maximize the matching size.

(a) Consider the deterministic greedy algorithm, where $u \in R$ is matched to an arbitrary available neighbor. Let OPT denote the maximum possible matching size in the offline setting (*i.e.*, when the graph is fully known). Show that the greedy online algorithm obtains a matching of size at least half of OPT.
Hint: Revisit Problem 4b of Discussion 5.

(b) Show that no deterministic online algorithm can achieve strictly better than half of OPT on all possible inputs.
Hint: Construct two inputs, where if the algorithm does better than half of OPT in one of them, it must achieve at most half of OPT in another. Use the fact that the algorithm cannot see future. Keep the construction simple—four vertices are enough!

(c) *Insanely hard challenge yet worth no point*: Show that the following randomized algorithm achieves $1 - 1/e$ fraction of OPT on any input. Give a random ordering of the vertices in L. When $u \in R$ arrives, if u has an unmatched neighbor in L, then we choose the unmatched neighbor $v \in L$ that comes earliest in the random ordering. (Otherwise, leave it unmatched.)

5 Global Mincut to Min s-t Cut

In class, we showed how to use maximum *s-t* flow to solve the minimum *s-t* cut. Now we ask you to consider the *global mincut* problem. Given a weighted, undirected graph G, the problem asks you to find a minimum weight set of edges whose removal disconnects the graph. (Note that there is no notion of s and t).

Show how to use maxflow or min s-t cut algorithms to solve this problem efficiently. Prove that your reduction would lead to the optimal solution. How many times do you need to invoke the maxflow or min s-t cut subroutine?

6 Dijsktra's Sort

Show how to use Dijsktra's algorithm to sort n real numbers (not necessarily non-negative or integral) in ascending order in $O(n \log n)$ time. You may use the intermediate outputs of Dijsktra's to do the sorting (as opposed to using it as a blackbox).

Argue that this means that improving upon the $O(m + n \log n)$ run-time of Dijsktra's algorithm (with Fibonacci heap) would lead to a faster sorting algorithm than merge and quick sort.

Hint: Given the numbers, construct a graph such that the order of vertices being extracted from priority queue by Dijsktra's algorithm corresponds to the right sorting order.

7 Path TSP and Cycle TSP

In the Traveling Salesman Problem (TSP), we are given an undirected graph with non-negative weights and asked to find a minimum weight cycle that contains each vertex exactly once.

In the s-t Traveling Salesman Problem (s-t TSP), we are given an undirected graph with non-negative weights, two vertices s and t, and are asked to find a minimum weight path that starts at s, visits all other vertices exactly once, and ends at t.

Show that if there is an algorithm to solve TSP, then you can use it to solve s-t TSP.