CS 270 Combinatorial Algorithms & Data Structures — Spring 2023

Problem Set 1

Due: 11:59pm, Friday, February 3, 2023
Solution maximum page limit: 3 pages

See homework policy at `https://cs270.org/spring23/syllabus/#homework-policies`

Throughout this problem set, whenever a graph is discussed, n is the number of vertices and m is the number of edges. We also assume that every vertex is reachable from some source, implying $m \geq n-1$.

Problem 1: In the BNW algorithm for SSSP from lectures 1 and 2, we discussed a subroutine ElimNeg that provides the following guarantee. If G is a directed graph with possibly negative edge weights but no negative cycles, then given a source vertex s ElimNeg finds shortest path distances from s to the rest of V in time $O(m\eta(G)\log n)$. Here $\eta(G) := \max_{v\in V}\eta_G(v)$, where $\eta_G(v)$ is the number of negative edges on the shortest path from s to v (if there are multiple shortest paths, then take the one with the fewest negative edges). Show how to improve this bound to $O((\sum_{i=1}^{n}\eta_G(v)+1)\log n)$; as discussed in lecture, you can and should assume that all vertex in-degrees and out-degrees are $O(1)$. **Hint:** Recall the memoized functions $h(\cdot), g(\cdot)$ introduced in lecture. Consider implementing the dynamic program bottom-up instead of top-down, and show that not all terms need to be calculated in every iteration.

Problem 2: We saw in class that we can combine Ford-Fulkerson with scaling to get runtime $O(m^2\log U)$ for max flow. Here we show that we can get a similar speed-up for Ford-Fulkerson without scaling, by picking the augmenting path in each step not arbitrarily, but by choosing the one with the largest minimum capacity amongst all its edges.

(a) (2 points) Prove the following: in a directed and capacitated graph any flow f can be decomposed into the sum of at most m flows $f_1,\ldots,f_r$, $r\leq m$, such that each f_i is supported on either a path or a cycle.

(b) (3 points) Given a directed graph with arbitary edge weights (could be positive or negative), and a start vertex s and end vertex t, we would like to find the path from s to t whose minimum weight is as large as possible. Show that this problem can be solved in $O(m+n\log n)$. **Hint:** modify Dijkstra's algorithm, and you may use that Dijkstra's algorithm runs in time $O(m+n\log n)$ when using a Fibonacci heap.

(c) (5 points) Recall the Ford Fulkerson s-t max flow algorithm repeatedly finds augmenting paths in residual graphs. For an augmenting path P define $u(P)$ as the minimum capacity on that path (in the residual graph). Suppose we augment along an augmenting path P which has maximum $u(P)$ value. Use (a) and (b) to show that the resulting max flow algorithm would run in time $O((m + n \log n)m \log(f^*)) = O((m + n \log n)m \log(mU))$, where U is the largest capacity.

Problem 3: This problem shows how to combine scaling with blocking flows.

(a) (3 points) In class we showed that the Dinic's blocking flow algorithm takes time $O(mn^2)$ on capacitated graphs. Show that another valid time bound is $O(mn + nf^*)$, where f^* is the value of the optimal flow. **Hint:** Find a different way of accounting for work done by advances and augments.

(b) (2 points) Show how to achieve an $O(mn \log U)$ time algorithm for max flow by combining blocking flow with scaling.

Problem 4: (1 point) How much time did you spend on this problem set? If you can remember the breakdown, please report this per problem. (sum of time spent solving problem and typing up your solution)