Princeton Univ Spring '26 COS 423: Theory of Algorithms

Homework 3

Out: *Mar 6*

Problems:

§1 Let $G = (V, E)$ be a graph with edge capacities.

(a) Give an example of G with n vertices such that Push-Relabel on G increases the height of *some* vertex to $2n - 2$.

(b) Recall that Relabel-to-Front moves a vertex that is Relabled during Discharge to the front of the list. Consider a variant where we keep the list in sorted order by *height* from the highest to the lowest. That is, after discharging a vertex u, we keep moving u forward in the list until u is in the right place, then continue discharing the vertices after u in the new order of the list. Does this variant also have $O(n^3)$ running time? Does this variant also always output a max flow?

(c) Suppose that you have run Push-Relabel to find the max flow in G. Give a fast algorithm to find a s-t min cut in G.

§2 (a) Give an example where a perfectly balanced 2-d tree spends $\Theta(\sqrt{n})$ time on a range counting query.

(b) Let $\epsilon > 0$ be a parameter. Suppose each node of the 2-d tree is ϵ-*balanced*, i.e., for a subtree of size s, its left and right subtrees both have size at least $\lfloor (1/2-\epsilon)(s-1) \rfloor$. What is the worst-case query time for a range counting query?

(c) Suppose we rebuild a subtree when it is no longer ϵ-balanced, what is the total running time for inserting n points?

§3 Given an array $A[0..n-1]$ for $n = 2^k$, its *bit-reversal permutation* swaps elements whose indices have binary representation that the reverse of each other. That is, for an index a, we represent it as a k-bit binary string $\langle a_{k-1}, a_{k-2}, \ldots, a_0 \rangle$, reversing its bits gives the binary string $\langle a_0, a_1, \ldots, a_{k-1} \rangle$. Suppose $a = \sum_{i=0}^{k-1} a_i \cdot 2^i$, define $\mathrm{rev}_k(a) = \sum_{i=0}^{k-1} a_{k-1-i} \cdot 2^i$. The bit-reversal permutation is to swap $A[a]$ and $A[\mathrm{rev}_k(a)]$ for all a.

(a) Given a function rev_k that runs in $O(k)$ time, design an algorithm to compute the bit-reversal permutation in $O(nk)$ time.

(b) Consider how to "bit-reverse-increment" in amortized $O(1)$ time. That is, design an algorithm such that given a, it outputs $\mathrm{rev}_k(\mathrm{rev}_k(a)+1)$, and iteratively calling the algorithm n times should take $O(n)$ time in total. Show how to use it to compute the bit-reversal permutation in $O(n)$ time.