cs473: Algorithms Assigned: *Tue., Sep. 17, 2019*

Problem Set #3

Prof. Michael A. Forbes
Prof. Chandra Chekuri
Due: *Wed., Sep. 25, 2019 (10:00am)*

Some reminders about logistics.

- **Submission Policy:** See the course webpage for how to submit your pset via gradescope.
- **Collaboration Policy:** For this problem set you are allowed to work in groups of up to three. Only one copy should be submitted per group on gradescope. See the course webpage for more details.
- **Late Policy:** Late psets are not accepted. Instead, we will drop several of your lowest pset problem scores; see the course webpage for more details.

All (non-optional) problems are of equal value.

1. Consider the subset sum problem: given n positive integers $a_1, \ldots, a_n > 0$ and a target integer T, decide whether there exists a subset $S \subseteq \{a_1, \ldots, a_n\}$ that sums to exactly T. It is easy to adapt the dynamic program presented in lecture for the knapsack problem to solve this problem in $O(nT)$-time and $O(nT)$ space, even if we want to output the desired subset (or otherwise declare that no subset exists).

   In lecture we saw a space-saving trick (which is due to Hirschberg) for computing not only the optimal edit-distance between two strings in smaller space, but *also* to output an optimal alignment within the same space bound. Apply this trick to the knapsack problem, to obtain a $O(nT)$-time, $O(T)$-space algorithm that outputs a subset summing to exactly T (or otherwise declares that no subsets exists).

2. You are given a directed graph $G = (V, E)$ where each edge e has a length/cost c_e and you want to find the lengths of the shortest paths from a given node s to all the nodes in V. Suppose there are only k edges $f_1 = (u_1, v_1), f_2 = (u_2, v_2), \ldots, f_k = (u_k, v_k)$ that have negative length, and the rest have non-negative lengths. The Bellman-Ford algorithm for shortest paths with negative length edges takes $O(nm)$ time where $n = |V|$ and $m = |E|$, even when k is small such as $k \leq O(1)$.

   Show that you can take advantage of the fact that there are only k negative length edges to find the lengths of the shortest paths from s in $O(kn \log n + km)$ time — effectively this is the running time for running Dijkstra's algorithm k times. Your algorithm should output the following: *either* that the graph has a negative length cycle reachable from s, *or* the lengths of the shortest paths from s to each of the nodes $v \in V$.

   Hint: First solve the case when there is only a single negative length edge. You will get half the credit if you only solve this case. Then solve the case of two negative length edges and see if you can generalize that approach. Note that this is only one approach, there may be others.

3. In the selection problem we are given an array A of n numbers (not necessarily sorted) and an integer k, and the goal is to output the rank k element of A. Consider a randomized algorithm where we pick a number x uniformly at random from A and use it as a pivot as in quick sort

to partition A into numbers less than equal to x and numbers greater than x. The algorithm recurses on one of these arrays depending on k and the size of the two arrays. It can be shown that this algorithm runs in $O(n)$ expected time and has the advantage of being quite simple when compared to deterministic algorithms with the same runtime (e.g., those that compute median of medians).

(a) Write down a description of randomized quick selection in pseudocode. Show that the expected depth of the recursion of randomized quick selection is $O(\log n)$. (You can also prove that the expected running time is $O(n)$. You do not have to submit this as part of this problem, but you should prove this for yourself as it will help with the next part.)

Hint: Write a recurrence for the depth of the recursion.

(b) Let $A_1, A_2, \ldots, A_h$ be h *sorted* arrays where A_i has n_i elements, and let $n = \sum_{i=1}^{h} n_i$. Assume that the arrays have distinct elements. Describe a randomized algorithm that given integer k finds the k-th smallest element in the combined set of arrays in $O(h \log^2 n)$ expected time.

Hint: Adapt the randomized quick selection algorithm and analysis from the first part.

Note: It is possible to deterministically achieve a time bound of $O(h \log n)$, but you are not asked for this.

4. (**optional**, *not* for submission) Let $G = (V, E)$ be an undirected graph. Recall that $S \subseteq V$ is a *dominating set* if the following property holds: for every $u \in V$ we have $u \in S$ or some neighbor of u is in S. In the *domatic partition problem* we are given a graph G and the goal is to find a maximum number of mutually disjoint dominating sets in G. Let δ be the degree of a minimum degree node u in G. It is easy to see that the domatic number is at most $\delta + 1$ since each dominating set has to contain either u or some neighbor of u. In this problem we will see that the domatic number of a graph on n nodes and minimum degree δ is at least as large as $\lceil \frac{\delta+1}{c \ln n} \rceil$ for some sufficient large universal constant c. Note that this guarantees only 1 dominating set if $\delta + 1 \leq c \ln n$ (the entire vertex set can be chosen as the dominating set).

Let $k = \lceil \frac{\delta+1}{c \ln n} \rceil$. Consider the following randomized algorithm. For each node u independently assign a color $g(u)$ that is chosen uniformly at random from the colors $\{1, 2, \ldots, k\}$.

(a) For a fixed node v and a fixed color i show that with probability at least $1 - \frac{1}{n^2}$ there is a node with color i that is either v or a neighbor of v. Choose c sufficiently large to ensure this.

(b) Using the above show that for a fixed color i the set of nodes that are colored i form a dominating set for G with probability at least $1 - \frac{1}{n}$.

(c) Using the above two parts argue that the domatic number of G is at least k.

Hint: The simple union bound is useful for this problem.