02-713 Homework #5
Due: Mar. 3 by 9:30am

You may discuss these problems with your current classmates, but you must write up your solutions independently, without using common notes or worksheets. You must indicate at the top of your homework who you worked with. Your write up should be clear and concise. You are trying to convince a skeptical reader that your answers are correct. Your homework should be submitted via Autolab (`https://autolab.cs.cmu.edu/02713-s14/`) as a typeset PDF. A LaTeX tutorial and template are available on the class website if you choose to use that system to typeset. For problems asking for an algorithm: describe the algorithm, an argument why it is correct, and an estimation of how fast it will run. Use O notation for running times.

1. Often we want to compute shortest paths between all pairs of vertices in a graph $G = (V, E)$. There are several ways to do this. One way to do this is to run Dijkstra's algorithm using each vertex as the start node. This takes $O(|V||E| \log |V|)$ time, but doesn't work if there are negative weights. Here we explore one fix to that suggested by Edmonds and Karp.

   (a) Give an example where Dijkstra's algorithm fails if there are edges of negative weight even if there are no negative cycles.

   (b) Let $G = (V, E)$ be a directed graph with possibly-negative weights $d(u, v)$ on the edges (but no negative cycles). Define a new graph G' that is created from G by adding a new vertex s and edges of 0-weight from s to every node in V. Let $dist_s(v)$ be the shortest path distance from s to v in G' computed via Bellman-Ford. Define new weights on G as $d'(u, v) = d(u, v) + dist_s(u) - dist_s(v)$. Argue that $d'(u, v)$ defined in this way is always positive.

   (c) Argue that any path is a shortest path in G using weights $d'(u, v)$ if and only if it is a shortest path in G using weights $d(u, v)$. (In other words, changing the weights to $d'(u, v)$ preserves which paths are shortest paths.)

   (d) Describe how to use (b) and (c) to compute all pairwise shortest paths in $O(|V||E| \log |V|)$-time.

2. Let $T = (V, E)$ be a tree with nodes V and edges E that is rooted at node r, and let $w(u)$ be the weight of node u. Recall that an independent set is a subset U of V such that no edge exists in between any two nodes in U. Give a polynomial time algorithm to compute the weight of largest-weight Independent Set in T.

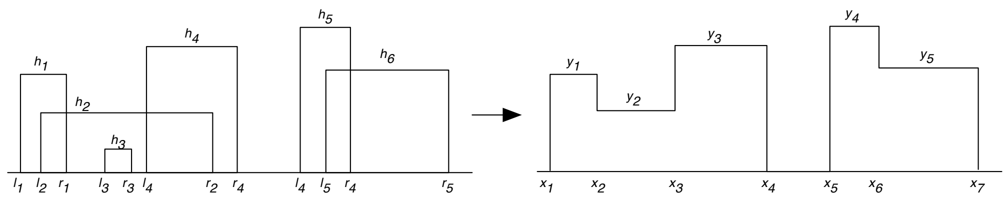
3. Let $x_1, \ldots, x_n$ be a list of integers. Give an $O(n)$ divide-and-conquer algorithm to find the largest possible sum of a subsequence of consecutive items in the list.

   Example: 10 -20 3 4 5 -1 -1 12 -3 1
   $\rightarrow$ 3 + 4 + 5 + -1 + -1 + 12 = 22

4. You have have found n id cards with magnetic strips that each encodes some id number. Due to security precautions, you do not have any way to read the id number off of a card directly. However, you do have a tester machine that will read two cards and tell you whether they have the same id number encoded on them.

   Give an $O(n \log n)$-time algorithm to test whether some subset of more than $n/2$ cards have the same identifier encoded on them.

5. **02-713 only!** You want to draw a two-dimensional skyline like the following:



You are given a list of the building x-coordinates and their heights:

$$(l_1, h_1, r_1), (l_2, h_2, r_2), \ldots, (l_n, h_n, r_n)$$

This list will be sorted in increasing order of left-most x-coordinate. **Sketch** an $O(n \log n)$ algorithm to produce the skyline line to draw in the format:

$$x_1, y_1, x_2, y_2, x_3 \ldots$$

meaning that at x_1 we draw a building at height y_1 until x_2 at which point we draw a line up or down to high y_2 and then continue horizontally until x_3 and so on.

Note that for this problem, you need only sketch the main ideas, not provide as detailed a discussion as typically.