ASSIGNMENT 3

DUE: Saturday July 9, 11:59 PM. DO NOT COPY. ACKNOWLEDGE YOUR SOURCES.
Please read http://www.student.cs.uwaterloo.ca/~cs341 for general instructions and policies.

Note: “Giving” an algorithm means doing the four parts (i)–(iv) as described on the course web page. This goes without saying for all future assignments and exams.


You are in charge of a photo contest, and must display all the photos sent in to the contest. You have photos $p_1, p_2, \ldots, p_n$, where photo $p_i$ has width $w_i$ and height $h_i$ (these need not be standard dimensions). The photos must remain in order. You need to place them in rows of maximum width $W$. Photos $p_i, \ldots, p_j$ can go in a row $r$ only if $\sum_{k=i}^{j} w_k \leq W$; the height of this row $r$ is defined to be $\max\{h_k : i \leq k \leq j\}$. The goal is to pack the photos into rows to minimize the sum of the heights of the rows.

Give a dynamic programming algorithm to find the minimum sum of row heights. As usual, give a correctness argument and a big-O expression for your runtime.

As part of your answer, explain in words what subproblem an entry of your DP table represents (i.e., not how it’s computed, but what the value it contains is supposed to represent).

Hint. Use $n$ subproblems. A runtime of $O(n^2)$ is possible, but you can get almost full marks for a clear solution with an $O(n^3)$ runtime.
2. [10 marks] **Dynamic programming. The power balancing problem.**

You are in charge of two power stations $S_A, S_B$, which can handle $M_A, M_B$ maximum watts of electrical load, respectively, before shutting down and causing total power blackout. Unfortunately, the power demand is currently higher than you can support, so you’re going to have to cut off service to some clients. Certain clients are very important, such as hospitals and telecommunication infrastructure, and you would like to somehow maximize the importance of clients that continue to receive power. There are a total of $n$ clients attempting to purchase power from your power stations, each of which has a positive integer importance $I_1, I_2, \ldots, I_n$, and a positive integer demand $D_1, D_2, \ldots, D_n$ in watts.

Each client can be powered by either $S_A$ or $S_B$. (You are free to choose which power station is used for each client.)

Your goal is to select two sets of clients $A_1, A_2, \ldots, A_k$ and $B_1, B_2, \ldots, B_\ell$ to be powered by stations $S_A$ and $S_B$, respectively, such that maximum electrical loads of both stations are respected, and the total importance of the powered clients is maximized.

In other words, we want to maximize the total importance $\sum_{j=1}^k I_{A_j} + \sum_{j=1}^\ell I_{B_j}$ subject to the constraints $\sum_{i=1}^k D_{A_i} \leq M_A$ and $\sum_{j=1}^\ell D_{B_j} \leq M_B$.

(a) Give an efficient dynamic programming algorithm to output the **maximum total importance** that you can obtain. Give a correctness argument, and a big-O expression for your runtime.

As part of your answer, explain in words what subproblem an entry of your DP table represents (i.e., not how it’s computed, but what the value it contains is supposed to represent).

*Hint: a runtime of $O(n \cdot M_A \cdot M_B)$ is sufficient to receive full marks.*

(b) Give an algorithm that takes the DP table produced by your algorithm above as input, and prints an optimal assignment of clients to $S_1$ and $S_2$. 


3. [10 marks] **Graphs.** The non-circular edge problem.

The input is a directed graph \( G = (V, E) \) containing \( m = |E| \) edges and \( n = |V| \) nodes. Give an \( O(n + m) \) time algorithm to output all edges that are not part of a cycle.

As usual, give pseudocode\(^1\), a correctness argument, and analyze the runtime of your algorithm (i.e., explain why it has the desired complexity).

4. [10 marks] **Graphs.** The almost-unweighted graph problem.

The input is an undirected weighted graph \( G = (V, E) \) and a source node \( s \). The edge weights in this graph are restricted: Every edge has weight 1 or 3. Let \( n = |V|, m = |E| \), and \( V = (v_1, v_2, ..., v_n) \).

The desired output is an array \( d[1..n] \), where \( d[i] \) contains the length of the shortest path from \( s \) to \( v_i \). (In other words, we want to solve the single-source shortest-path problem.)

Provide an \( O(n + m) \) algorithm (including pseudocode\(^1\)) to solve this problem.

As usual, justify correctness and analyze the runtime of your algorithm (i.e., explain why it has the desired complexity).

---

\(^1\)You are allowed to reference graph algorithms that you learned about in class, such as BFS, DFS, SCC, topsort, etc. If you want to modify such an algorithm, you can simply explain the changes you would make. You might wonder how to include precise pseudocode without completely replicating all of the pseudocode for an algorithm you want to modify. As an example of how to do this well, you could explain in words how you would modify BFS() to obtain a new procedure called ModifiedBFS(), and then invoke ModifiedBFS() in your pseudocode. If in doubt, make your solution as clear and understandable as possible.