Homework 6

Fundamental Algorithms, Fall 2022, Professor Yap; Section Leaders Daniel Feldan and Zeming Lin

Due: Fri Nov 11, in GradeScope by 11pm.

INSTRUCTIONS:

- Please download the latest version of Chapter 5 (`l5-class.pdf`) from the Schedule Page in ClassWiki. The page references below refers to this version.
- The reading list for Chapter 5 may be narrowed down to §1, 3, 4, 6.
- For general instructions about homework, please look in hw1 or hw2.

(Q1) (20 Points) Improving Brute Force Bin Packing

Improve the bin packing upper bound in Lemma 2 (¶V.6, p.6) to $O((n/e)^{n-(1/2)})$.
HINT: Repeat the Karp-Held trick that saved us a factor of n. Fix two weights w_1, w_2 and consider two cases: either w_1, w_2 belong to the same bin or they do not.

(Q2) (6+20 Points) Linear Bin Packing with Negative Weights

(a) Consider the linear bin packing problem when the weights in $w = (w_1, \ldots, w_n)$ can be negative. A solution with k bins is determined by its the sequence of breakpoints, $0 = t_0 < t_1 < t_2 < \cdots < t_k = n$ where the ith bin holds the weights

$$w(t_{i-1}..t_i] := \sum_{j=t_{i-1}+1}^{t_i} w_j.$$

Here is a greedy way to define these indices: for $i \geqslant 1$, assuming t_{i-1} is defined, let t_i to be the largest index (but at most n) such that $w(t_{i-1}..t_i] \leqslant M$. Either prove that this solution is optimal (for linear bin packing), or give a counter example.
NOTE: this algorithm is no longer "online".

(b) Give an $O(n^2)$ algorithm for linear bin packing when there are negative weights.
HINT: When solving the problem for (M, w), assume that you already know the solution for each (M, w') where w' is a suffix of w.

(Q3) (6+20 Points) Activities Selection to maximize length

Consider the activities selection problem. See ¶V.16, p.21. Let $S = \{I_1, \ldots, I_n\}$ be a set of activities where each activity $I_i = [s_i, f_i)$ is a half-open interval. We want to find a compatible set $A \subseteq S$ which maximizes the **length** $|A|$ where

$$|A| := \sum_{I \in A} |I|$$

and $|I_i| := f_i - s_i$. Denote by $Opt(S)$ the maximum length of $A \subseteq S$.

Let $S_{i,j} = \{I_i, I_{i+1}, \ldots, I_j\}$ for $i \leqslant j$ and $Opt_{i,j} := Opt(S_{i,j})$.

(a) Show by a counter example that the following algorithm does not work:

$$Opt_{i,j} = \max\{Opt_{i,k} + Opt_{k+1,j} : i \leqslant k \leqslant j-1\} \tag{1}$$

HINT: May assume $|S| \leqslant 4$ in the counter example.

(b) Give an $O(n \log n)$ algorithm for computing $Opt_{1,n}$.
HINT: order the activities in the set S according to their finish times.

(Q4) (10+20 Points) Huffmann coding

(a) Let s be the following string: "`hello!␣this␣is␣my␣little␣world!`". Show the Huffman code C for s, and what is $|C(s)|$?

(b) Suppose you are given the frequencies f_i in sorted order. Show that you can construct the Huffman tree in linear time.

(Q5) (10+10 Points) MST

We consider minimum spanning trees (MST's) of the bigraph $G = (V, E)$ where each vertex $v \in V$ is given a numerical value $C(v) \geqslant 0$. The **cost** $C(u, v)$ of an edge $(u-v) \in E$ is defined to be $C(u)+C(v)$. For each simulation below, please also state the cost of your MST.
Please organize your computation so that we can verify the computation results. See ¶V.37 (p.60) for how to do this for Prim's algorithm. For Kruskal's algorithm, it is best to list the edges in increasing sorted order, and successively, accept or reject each edge.

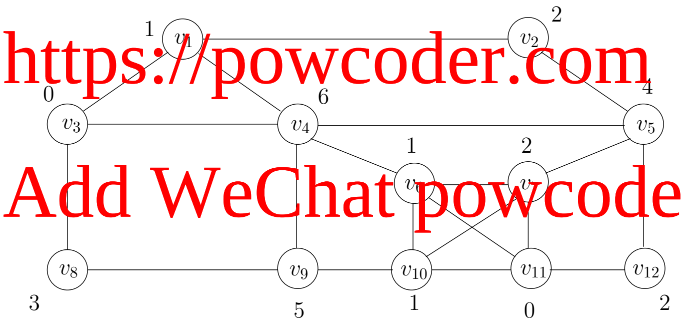

Figure 1: The house graph: The cost of edge $v_i - v_j$ is defined as $C(v_i) + C(v_j)$, where $C(v)$ is the value of vertex v. E.g. $C(v_1 - v_4) = 1 + 6 = 7$.

(a) Simulate Prim's algorithm on the bigraph G of Figure 1.

(b) Simulate Kruskals's algorithm on G.