Homework 5
Fundamental Algorithms, Fall 2002, Professor Yap

Due: Thursday Nov 28, in class

1. **Linear Bin Packing Problem (5+15 Points)**
   In the Linear Bin Packing Problem discussed in class, you are given

   $$M, w = (w_1, w_2, \ldots, w_n)$$

   such that $0 \leq w_i \leq M$, and the goal is to find a **solution** $[n_1, n_2, \ldots, n_k]$ that minimizes k. This is a solution if $1 \leq n_1 < n_2 < \cdots < n_k = n$ and $\sum_{j=n_{i-1}+1}^{n_i} w_j \leq M$ holds for all $i = 1, \ldots, k$. Assume $n_0 = 0$.
   (a) Suppose we remove the constraint that w_i be non-negative. Show the greedy method fails. NOTE: We had discussed this in class. Here we want you to give a counter example where n is as small as possible. Partial credits when n is larger than necessary.
   (b) Give a dynamic programming solution in case w_i can be negative.

2. **Shift Key in Huffman Code (15 Points)**
   We want to encode small as well as capital letters in our alphabet. Thus 'a' and 'A' are to be distinguished. There are two ways to achieve this: (I) View the small and capital letters as distinct symbols. (II) Introduce a special "shift" symbol, and each letter is assumed to be small unless it is preceded by a shift symbol, in which case it is considered a capital. Use the text of this question as your input string. Punction marks and spaces are part of this string. But new lines (CRLF) do not contribute any symbols to the string.
   (a) Compute the Huffman code tree for coding the above string using method (I). Note that the string begins with the words "We want to en..." and ends with "...bute any symbols to the string.". Be sure to compute the number of bits in the Huffman code for this string.
   (b) Same as part (a) but using method (II).
   (c) Discuss the pros and cons.

3. **Amortization (20 Points)**
   Do Exercise 1.1 in the handout on amortization.

4. **Splay Trees (15 Points)**
   Do Exercise 2.1 in the handout on amortization.

5. **Prim's Algorithm (20 Points)**
   Hand simulate Prim's algorithm on the following graph (figure 1) beginning with v_1.

   It amounts to filling in the following table, row by row. The vertex set V is partitioned into the two sets, $S \subseteq V$ and $U = V \setminus S$. The set U is in a priority queue. Each row of the table represents the array $lc[u]$ $(u \in U)$ where $lc[u]$ is the least cost of an edge that connects u to any node in S.

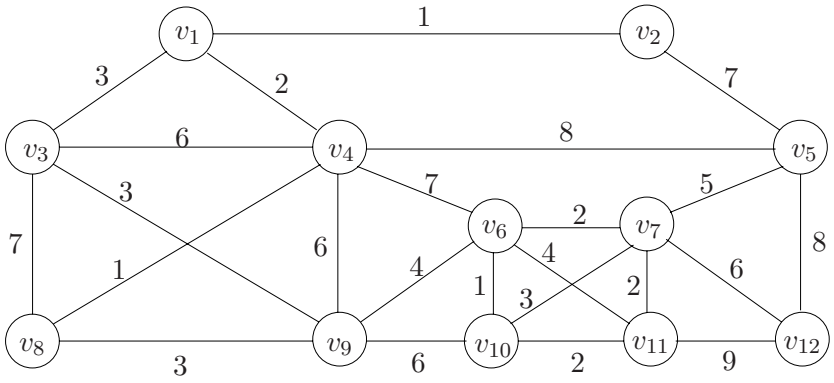

Figure 1: Graph of a House

We also maintain a set T of edges. The set T forms a minimum spanning tree for S. The table also has an entry $mst[T]$ for the sum of the weights of edges in T. We have filled in the first two rows already.

i	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	$mst[T]$	New Edge in T
1	$\underline{1}$	3	2	∞	∞	∞	∞	∞	∞	∞	∞	1	(v_1, v_2)
2	*	"	$\underline{2}$	8	"	"	"	"	"	"	"	3	(v_1, v_4)

Note that the minimum cost in each row is underscored, indicating the item to be removed from the priority queue (or the set U). A 33 entry just means it is unchanged form before. An * entry means that the node is no longer in U.

6. **Kruskal's Algorithm and Union Find (10+20 Points)**
We want to simulate Kruskal's algorithm on the same graph as the previous question.

(a) First show the list of edges, sorted by non-decreasing weight. View vertex v_1, v_2, v_3, etc as the integers $1, 2, 3$, etc. We want you to break ties as follows: assume each edge has the form (i, j) where $i < j$. When the weights of (i, j) and (i', j') are equal, then we want (i, j) to appear before (i', j') iff (i, j) is less than (i', j') in the lexicographic order, i.e., either $i < i'$ or $(i = i'$ and $j < j')$.

(b) We want you to maintain the union-find data structure needed to answer the basic question in Kruskal's algorithm (namely, does adding this edge creates a cycle?). The algorithms for the Union Find problems MUST use the 2 heuristics we discussed: rank heuristic and path compression. At each stage of Kruskal's algorithm, when we consider an edge (i, j), we want you to preform the corresponding $Find(i), Find(j)$ and, if necessary, $Union(Find(i), Find(j))$. You must show the result of each of these operations on the union-fine data structure.