CSE 3500 Algorithms and Complexity. Fall 2016
Exam II Solutions

1. We first merge $X_1, X_2, \ldots, X_m$ as follows. Recursively merge $X_1, X_2, \ldots, X_{m/2}$ to get Y_1. We also recursively merge $X_{(m/2)+1}, \ldots, X_m$ to get Y_2. We then merge Y_1 and Y_2 to get Y. Note that Y is a sorted sequence containing all the elements from $X_1, X_2, \ldots, X_m$. Let $T(m)$ be the time needed to merge two $m/2$ sequences. Then $T(m) = T(m/2) + \frac{n}{2}$. This solves to: $T(m) = O(n \log m) = O\left(n(\log n)^{1/3}\right)$.

   We then sort X_{m+1} using the radix sort algorithm to get Z. We know that we can sort N integers in the range $[1, N^{f(N)}]$ in $O(N\ f(N))$ time. This means that we can sort X_{m+1} in $O\left(n(\log n)^{1/3}\right)$ time.

   Finally, we can merge Y and Z in $O(n)$ time. The total run time is $O\left(n(\log n)^{1/3}\right)$.

2. (a) Here is an algorithm:

   Find(X, R)
   1) Find the median M of X;
   2) Partition X into X_1 and X_2 such that $X_1 = \{q \in X : q < M\}$ and $X_2 = \{q \in X : q > M\}$;
   3) Let $\sum_{q \in X_1} q = S_1$ and $\sum_{q \in X_2} q = S_2$;
   4) **if** $S_1 + M < R$ **then** Find($X_2, R - S_1 - M$);
   5) **if** $S_1 < R$ and $S_1 + M \geq R$ **then** output M and quit;
   6) **if** $S_1 > R$ **then** Find(X_1, R);

   Analysis: Let $T(n)$ be the run time of Find(X, R) where X has n elements. Steps $1, 2,$ and 3 take a total of $O(n)$ time. Also note that only one of the three cases in steps $4, 5,$ and 6 will hapen and these three cases are exhaustive. In the worst case either step 4 happens or step 6 happens. Thus $T(n)$ satisfies: $T(n) \leq T(n/2) + O(n)$. Using the Master theorem, we can see that $T(n) = O(n)$.

   (b) The problem is much simpler when X is in sorted order. Let $X = k_1, k_2, \ldots, k_n$ in sorted order. Here is an algorithm:

   $sum = 0.0$;
   for $i = 1$ **to** n **do**
   if $sum < R$ and $sum + k_i \geq R$ **then** output k_i and quit;
   $sum = sum + k_i$;

   Clearly, the above algorithm takes $O(n)$ time.

3. Kruskal's algorithm starts with a forest with 7 nodes and no edges. The edges are sorted in non-decreasing order of the edge weights: $(1,2), (3,5), (1,3), (2,3), (3,6), (5,6), (4,5), (2,7), (6,7), (2,4), (5,7)$. We insert the edge $(1,2)$ into the forest; Followed by this we insert the edge $(3,5)$. The next edge $(1,3)$ will cause a cycle if inserted into the forest and hence is thrown out. Proceeding in this manner, we get the following minimum spanning tree with a total weight of 14: $(1,2), (1,3), (2,7), (3,5), (3,6), (4,5)$.

4. To begin with we have: $S = \{s\}$, $dist(1) = 10, dist(2) = 2, dist(3) = \infty, dist(4) = \infty$, and $dist(5) = 12$.

Phase 1: Node 2 has the minimum $dist$ value and hence it enters S next: $S = \{s, 2\}$. $dist(1) = \min\{dist(1), dist(2) + W(2,1)\} = \min\{10, 2+3\} = 5$. $dist(3) = \min\{dist(3), dist(2) + W(2,3)\} = \min\{\infty, 2+2\} = 4$. $dist(4) = \min\{dist(4), dist(2)+W(2,4)\} = \min\{\infty, 2+\infty\} = \infty$. $dist(5) = \min\{dist(5), dist(2) + W(2,5)\} = \min\{12, 2+\infty\} = 12$.

Phase 2: Node 3 has the least $dist$ value and hence it enters S next: $S = \{s, 2, 3\}$. $dist(1) = \min\{dist(1), dist(3) + W(3,1)\} = 5$. $dist(4) = \min\{dist(4), dist(3) + W(3,4)\} = \min\{\infty, 4+1\} = 5$. $dist(5) = \min\{dist(5), dist(3) + W(3,5)\} = \min\{12, 4+2\} = 6$.

Phase 3: Nodes 1 and 4 have the same minimum $dist$ value. We could insert any one of these into S next. Let $S = \{s, 2, 3, 4\}$. $dist(1) = \min\{dist(1), dist(4)+W(4,1)\} = \min\{5, 5+\infty\} = 5$. $dist(5) = \min\{dist(5), dist(4) + W(4,5)\} = \min\{6, 5+3\} = 6$.

Phase 4: Node 1 enters S next: $S = \{s, 1, 2, 3, 4\}$. $dist(5) = \min\{dist(5), dist(1)+W(1,5)\} = \min\{6, 5+\infty\} = 6$.

Phase 5: Node 5 enters S next.

Thus the shortest path weights to the nodes $1, 2, 3, 4$, and 5 are $5, 2, 4, 5$, and 6, respectively.

5. The recurrence relation we derived for the 0/1 knapsack problem is: $f_i(y) = \max\{f_{i-1}(y), f_{i-1}(y - w_i) + p_i\}$. We are interetsed in computing $f_n(m)$. We compute a $(n+1) \times (m+1)$ matrix M whose first row and the first column are all zeros. We can compute M in a row major order as follows:

   $f_1(1) = \max\{f_0(1), f_0(1-2) + 7\} = \max\{0, -\infty\} = 0.$

   $f_1(2) = \max\{f_0(2), f_0(2-2) + 7\} = \max\{0, 7\} = 7.$

   $f_1(3) = \max\{f_0(3), f_0(3-2) + 7\} = \max\{0, 7\} = 7.$

   ...

   $f_2(3) = \max\{f_1(3), f_1(3-3) + 3\} = \max\{7, 3\} = 7.$

   ...

   $f_2(5) = \max\{f_1(5), f_1(5-3) + 3\} = \max\{7, 7+3\} = 10.$

   ...

   $f_3(5) = \max\{f_2(5), f_2(5-4) + 3\} = \max\{10, 0+2.5\} = 10.$

   ...

   $f_3(8) = \max\{f_2(8), f_2(8-4) + 2.5\} = \max\{10, 9.5\} = 10.$

   The final answer is 10.

6. We utilize the fact that we can compute the edit distance between two strings X and Y in $O(mn)$ time where $|X| = n$ and $|Y| = m$. We can compute the edit distance between every pair of strings and output the pair whose distance is minimum. Let $|S_i| = \ell_i$, for $1 \le i \le n$.

   The total time needed is $O\left(\sum_{j=1}^{n} \sum_{i=1}^{n} \ell_i \ell_j\right) = O\left(\sum_{j=1}^{n} \ell_j(\ell_1 + \ell_2 + \cdots + \ell_n)\right) = O\left(\sum_{j=1}^{n} \ell_j N\right) = O\left(N \sum_{j=1}^{n} \ell_j\right) = O(N^2)$.