CS 670 - Fall 2010 - Solutions to Midterm

October 29, 2010

Problem 1

Sort the array B. For every $i \in \{1, 2, \ldots, n\}$ search (using binary search) in the array B if there exists a j such that $A[i] = B[j]$. The running time is $\mathcal{O}(n \log n)$.

Problem 2

Put the array A in a min-heap (in $\mathcal{O}(n)$ time). Start extracting the(a) minimum element from the heap one by one (in $\mathcal{O}(\log n)$ time). Clearly, the i^{th} extraction yields b_i. After each extraction, check if $b_1 + b_2 + \ldots + b_i > b_{i+1}$. If true, then $d_A = i$. Hence after $d_A + 1$ extractions, we obtain $b_1, b_2, \ldots, b_{d_A}$.

The total running time is $\mathcal{O}(n + d_A \log n)$

Problem 3

(1) Assume that the max weight subsequence does not end with q. Since both A and B end with q, appending q to the end of the max weight subsequence will result in a subsequence of higher weight(a contradiction). Hence every max weight subsequence should end with q.

(2) Let $Opt_{i,j}$ denote the weight of the maximum weight subsequence of the sequences $A[1, \ldots, i]$ and $B[1, \ldots, j]$.

If $A[i] = B[j]$, then $Opt_{i,j} = A[i] + Opt_{i-1,j-1}$ (append $A[i]$ to the subsequence corresponding to $Opt_{i-1,j-1}$). This follows from the definition of Opt_{ij} and part (1).

Otherwise, $Opt_{i,j} = \max(Opt_{i,j-1}, Opt_{i-1,j})$ (keep track of subsequence corresponding to the choice made).

The initial condition is $Opt_{0,0} = Opt_{1,0} = Opt_{0,1} = 0$. We need to be careful about the order of computation of the subproblems. One orderering that

works is $Opt_{1,1}, Opt_{1,2}, \ldots, Opt_{1,m}, Opt_{2,1}, Opt_{2,2}, \ldots, Opt_{2,m}, \ldots, Opt_{n,m}$.

The number of subproblems is $\mathcal{O}(nm)$ and computing each one takes constant time. The running time is thus $\mathcal{O}(nm)$.

Problem 4

(1) Let $n = 100q + r$ where $0 \leq r < 100$. Assume that an optimum solution has q' coins of denomination 100, where $q' \neq q$. Clearly $q' < q$ (otherwise the total would exceed n). Now, $n - q'100 > 100$. The optimal solution had to have change for exactly 100 cents using ten cents and one cent coins (This is because it either had atleast 10 ten cent coins or $t < 10$ ten cent coins and atleast $100 - 10t$ one cent coins). By replacing this change for 100 cents with one 100 cent coin, we reduce the number of coins (by at least 9). Hence the solution was not optimal and we arrive at a contradiction.

(2) From part (1), we know the optimal solution has to have exactly q hundred cent coins. Now consider the change for the remaining r cents. We claim that the number of ten cent coins an optimal solution has is exactly t, where $t = \lfloor \frac{r}{10} \rfloor$. Otherwise the optimal solution contains at least 10 one cent coins which can be replaced by 1 ten cent coins thereby contradicting its optimality.

Hence the only optimal solution is to have exactly q hundred cent coins, t ten cent coins and rest in one cent coins.

Problem 5

After each split the number of teams increases by 1. Likewise after each merge the number of teams decreases by 1. Assume that the number of merges is strictly greater than $N + n_0$. At the end, the number of teams remaining is $n_0 + N - \#merges < 0$. This is a contradiction as the number of teams cannot be negative. Hence the number of merges is at most $n_0 + N$ and the total cost is at most $NA + (n_0 + N)B$.