

CSE 361 Complexity of Sequential and Parallel Algorithms

Spring 2008 Exam II – Solutions

1. (a). Let $R = \{R_1, \dots, R_r\}$ be the output of greedy algorithm, and $S = \{S_1, \dots, S_s\}$ be that of an optimal strategy in nondecreasing order of program size. Let i be the least index in which these two differ. Clearly, $R_i < S_i$. Also note that $R_i \notin S$. Thus we can insert R_i into S before S_i and delete S_s , if there is a need. The new solution has at least the same number of programs as before. Proceeding in this fashion, we can make the first r programs of S the same as the corresponding r programs of R . There can not be space left in the tape for any more programs. If there were, the greedy algorithm would have added at least one more program. Thus $s \leq r$.
 - (b). If each $a_i > l$, the ratio is 0.
2. (a). $f_i(y) = \max\{f_{i-1}(y), \max_{k, y \geq kw_i} \{f_{i-1}(y - kw_i) + kp_i\}\}$.
 - (b). If we fix i and y , $f_i(y)$ can be computed in $O(m)$ time. But $i \in [1..n]$, $y \in [1..m]$. Thus the running time is $O(m^2n)$.
 - (c). For $i \in [1..n]$, calculate p_i/w_i . As a result, find the object k with the maximum profit density. Fill the knapsack with this object. Then $x_k = m/w_k$, and the total profit is $\frac{m}{w_k}p_k$.
3. Let $P(i, k)$ be 1 if there exists a subset whose sum is k from among the first i items and zero otherwise. We are interested in computing $P(n, K)$. A recurrence relation for $P(i, k)$ is given by:

$$P(i, k) = 1 \text{ iff either } P(i - 1, k) = 1 \text{ or } P(i - 1, k - k_i) = 1$$

We can use the above recurrence relation to compute $P(n, K)$ in $O(nK)$ time. For example we can compute the following sequence: $P(1, 1), P(1, 2), \dots, P(1, K), P(2, 1), P(2, 2), \dots, P(2, K), \dots, P(n, 1), \dots, P(n, K)$.

4. This is similar to the problem done in class. Let the three sequences under concern be X, Y and Z . Let $cost(i, j, k)$ stand for the minimum cost of making the three sequences $x_1, x_2, \dots, x_i, y_1, y_2, \dots, y_j$, and z_1, z_2, \dots, z_k the same. A recurrence relation for $cost(i, j, k)$ can be written as follows. We can solve for $cost(n, n, n)$ in $O(n^3)$ time.

$$\begin{aligned}
 cost(i, j, k) = \text{Min}\{ & Cost(i - 1, j, k) + 1, \\
 & Cost(i, j - 1, k) + 1, \\
 & Cost(i, j, k - 1) + 1, \\
 & Cost(i, j - 1, k - 1) + 1, \\
 & Cost(i - 1, j - 1, k - 1) + 0(\text{if } x_i, y_j, \text{ and } z_k \text{ are the same}) \text{ and } 1 \text{ (otherwise)}, \\
 & Cost(i - 1, j, k - 1) + 1, \\
 & Cost(i - 1, j - 1, k) + 1
 \end{aligned}$$
5. Use BFT to find the connected components of the given graph. Fill the transitive closure matrix A^* as follows: $A^*(i, j) = 1$ iff either $i = j$ or $i \neq j$ and i and j are in the same connected component. Total complexity = complexity of BFT + complexity of filling the A^* matrix = $O(|V| + |E|) + O(|V|^2) = O(|V|^2)$.
6. A simple algorithm would be to use one of the graph traversal algorithms **BFT** or **DFT**, and color nodes on the way. Say we use BFT. Start with an arbitrary node, color it **Orange**. Then go over to its neighbors and color them **Blue**. Continue in this way until -
 - (a) You attempt coloring an **Orange** node **Blue** or vice versa. That means that there are 2 nodes of the same color that are neighbors. Report that the graphs is *not bipartite*.
 - (b) You end the traversal, with no conflicts. Report that the graph is *bipartite*.