**CSL356** Sept 11,15,16

# TUTORIAL SHEET 6 solutions

1. Build a table $T[]$ where $T[i]$ stores the minimize the sum of placement and access cost assuming we have servers $S_i, \ldots, S_n$ and we place a copy at $S_i$. So, $T[n]$ is $c_n$. Now to compute $T[i]$, we need to place a copy at $S_i$. Suppose the optimal solution for the problem considered by $T[i]$ places the next copy at $T[k]$ $(k > i)$ – then we need to pay for the access cost of $k - i$. Therefore,

$$T[i] = c_i + \min_{k=i+1,\ldots,n}(k - i + T[k]).$$

2. We modify Bellman Ford algorithm. We build a table $S[i, u]$, which stores the length of the shortest path from $u$ to $w$ which uses at most $i$ edges. Further, we have a table $T[i, u]$ which stores the number of paths using $i$ edges from $u$ to $w$ whose cost is $S[i, u]$ (i.e., the shortest path using $i$ edges). Now, suppose the out-neighbours of a vertex $u$ are $v_1, \ldots, _k$. Then,

$$S[i+1, u] = \min_{r=1}^{k}(l_{(u,v_r)} + S[i, v_r]).$$

Now, let $v_{s_1}, \ldots, v_{s_l}$ be the neighbours of $u$ which achieve the minimum above, i.e., for which $S[i+1, u] = l_{(u,v_r)} + S[i, v_r]$. Then, we update

$$T[i+1, u] = T[i, v_{s_1}] + \ldots + T[i, v_{s_l}].$$

See how to initialize the tables.

3. Build a table $T[i, s]$ which tells the optimal solution for month $i$ till $n$ given that you have $s$ trucks at the beginning of month $i$ (before you place any order for this month). How many trucks can you order at the beginning of month $i$ if you already have $s$ trucks ? The maximum would be $S - s + d_i$. If you order $o_i$ trucks, then $o_i + s$ must be at least $d_i$. Thus, the number of trucks you can order lies in the range $[\max(0, d_i - s), S - s + d_i]$. If $d_i < s$, you may not order any trucks. So, if $d_i < s$, then

$$T[i, s] = \max(T[i+1, s - d_i] + C(s - d_i), \max_{l=0}^{S-s+d_i}(K + C(s + l - d_i)).$$

The argument for the case $d_i > s$ is similar except that we will not have the first term, and the range of $l$ in the second term will be from $d_i - s$ to $S - s + d_i$.

4. Have a table $T[]$, where $T[i]$ tells you where the part of the string $s[i...n]$ can be reconstituted (so, the table entry is true or false).

5. Note that there are only 7 ways in which you can tile a particular column – call these ways $W_1, \ldots, W_7$ (2 ways in which you can put two pebbles, 4 for 1 pebble, and 1 for 0 pebble). Call two arrangements $W_i, W_j$ compatible if placing pebbles like $W_i$ and $W_j$ in two adjacent columns (with $W_i$ being on the left) does not violate any rules. Now have a table $T[i, c]$ which tells you the optimal placement for columns $c$ till $n$ provided the configuration in the column $c$ is $W_i$. Now,

$$T[i, c] = \text{value}(W_i) + \max_j T[j, c+1],$$

where the maximum is taken over those configurations $W_j$ which are compatible with $W_i$.

6. Have table $T[]$, where $T[i]$ tells you the day after $i$ on which the stock price is maximum (this is the day on which you should sell in case you buy on day $i$ provided the stock price is higher than the price on day $i$, otherwise you should not sell at all if you buy on day $i$). Clearly, $T[n-1]$ is $n$. If $i < n-1$, then $T[i]$ is either $i+1$ or $T[i+1]$ depending on which day has higher stock price.