CSCE 629 - 602 Analysis of Algorithms

September 12, 2016

Homework I
Prof. Anxiao (Andrew) Jiang

Rishabh Yadav
UIN: 425009824

1A. Solution

Let us take a rod of length 4. Let the length be l_i, price be p_i and density be d_i and value in the table as follows.

l_i	1	2	3	4
p_i	1	8	21	24
d_i	1	4	7	6

For greedy approach the highest d_i is of rod of length 3 as part of our solution, so then we will take one rod of length 3. After which the only remaining rod of length 1 is remaining so that is taken into our solution. Profit for this solution is,
$p_{total} = p_3 + p_1$
$p_{total} = 21 + 1$
$p_{total} = 22$

If we look carefully we see that the optimal solution for the above example is rod of length 4, for which the profit is 24. Clearly this is more than the profit from the solution of the Greedy approach. Hence greedy algorithms here do not give optimal solutions.

2A. Solution 15-7

15-7 a

Idea

The idea is to maintain an Table (T) of size kXn where k is the total number of sounds in $s =< \sigma_1, \sigma_2.....\sigma_k >$ and n is the total number of vertices $v \epsilon$ V in the given graph. We will fill out the Table (T) with initial values as 0. For the first sound σ_1 we will pre-populate the first column of the Table (T) with a 0 if there is no edge with label σ_1 (the first sound in the sequence $s =< \sigma_1, \sigma_2.....\sigma_k >$). And store 1 for each edge $E(v_0, v_j)$ with sound label $\sigma(v_0, v_j) = \sigma_1$. While doing this we also store the path till now to reach this current vertex in the table.

The idea is to iterate through each column, i (with sound label σ_i), we look for vertices with cell value 1 in the previous column (i-1) and see if the there is an outward going edge with sound label σ_i for those vertex ($E(v_j, v_l)$ with sound label $\sigma(v_j, v_l) = \sigma_i$). Then we update vetrex v_l row and update the entry T[i][l] = 1. Also store the path till this point $P_{tillNow}$. After completion, we need look at the last column, all the entries with value 1 can will give us the solution(s). Also as we are storing the path till now, we can also find the path by examining the $P_{tillNow}$ of all the entries with values 1 in the last column

Pseudo Code

1. initialize table T[k][n] = {0}, where $1 \leq j \leq$k is the number of sounds in the sequence and V is the number of vertices.
2. pre-populate the first column cells, T[σ_1][v_j], with 1 if there is an outward edge (v_0,v_j) ϵ E, the edges in the graph. Store the Path till now for $P_{tillNow} =< v_0 >$.
3. for every σ_i, where $i = 2$ to k in $s =< \sigma_1, \sigma_2.....\sigma_k >$
   (a) for every vertex $v_j \epsilon$ V, $j = 1$ to n
      i. Check all the out edges of v_j with edge label σ_i
      ii. if edge found then update T[σ_i][v_j] = 1 and also update the $P_{tillNow}$
4. check each vertex v_j in the last column, σ_k
   (a) if cell has entry 1 then return $P_{tillNow} + vertex\ v_j$
5. return $NO - SUCH - PATH$

Proof of correctness

Let the desired path with sound sequence $s =< \sigma_1, \sigma_2.....\sigma_k >$ be P_{path}.

Let sound sequence till current point be $s_{k-1} =< \sigma_1, \sigma_2.....\sigma_{m-1} >$. In next iteration, for every vertex (v_i) we consider all the outward going valid entry (i.e. 1) in the

column $m-1$ of the table. After examining all the edges we update cells in the column m corresponding to v_j such that $(v_i, v_j)\ \epsilon$ E and $\sigma(v_i, v_j) = \sigma_m$. This way we are actually considering all the valid path till point v_j and leaving out all the unnecessary edges, which we know will not lead to a path path P with sound sequence s. In the end we are examining the last column of the constructed table, which will contain the constructed path for all the valid entries(non-zero). Any valid entry will give us a desired path P_{path}, and if all the columns have zero then no such path exists.

Time Complexity

The time complexity of the above mentioned algorithm is as follows,

For each cell in the table we need to examine all the out-ward going edges. If we look one column level at a time, then for every column we need to examine all the edges and vertices in the given graph. The total time taken to compute value of one column is $O(e+n)$, where n is the number of vertices in the graph and e is the number of edges. We totally have k columns, Hence the total time Complexity is $O(e+n)*k$.

$Time\ Complexity = O(e+n)*k.$

15-7 b

Idea

For solving part two instead of storing Boolean values in the table, we store the product of probabilities of the edges in the path till that point. We just need to examine the last column, σ_k. Fin the cell with maximum probability value and return the path $P_{tillNow} + vertex\ v_j$

Pseudo Code

1. initialize table T[k][n] = {0}, where $1 \leq j \leq$k is the number of sounds in the sequence and V is the number of vertices.
2. pre-populate the first column cells, T$[\sigma_1][v_j]$, with probability pr_e where p_e is the probability of the edge $(v_0, v_j)\ \epsilon$ E, the edges in the graph. Store the Path till now for $P_{tillNow} =< v_0 >$. Also store the product of probability till now, $prob_{tillNow} = prob_{tillNow} * prob_e$.
3. for every σ_i, where $i = 2$ to k in $s =< \sigma_1, \sigma_2.....\sigma_k >$
   (a) for every vertex $v_j\ \epsilon$ V, $j = 1$ to n
      i. Check all the out edges of v_j with edge label σ_i
      ii. if edge found then update T$[\sigma_i][v_j] = prob_e$ also update the $P_{tillNow}$
4. check each vertex v_j in the last column, σ_k
   (a) find max $prob_{tillNow}$ and return $P_{tillNow} + vertex\ v_j$

5. return $NO - SUCH - PATH$

Proof of correctness

Let the desired path with sound sequence $s =< \sigma_1, \sigma_2.....\sigma_k >$ be P_{path}.

Let sound sequence till current point be $s_{k-1} =< \sigma_1, \sigma_2.....\sigma_{m-1} >$. In next iteration, for every vertex (v_i) we consider all the outward going valid entry (i.e. 1) in the column $m-1$ of the table. After examining all the edges we update cells in the column m corresponding to v_j such that (v_i, v_j) ϵ E and $\sigma(v_i, v_j) = \sigma_m$. This way we are actually considering all the valid path till point v_j and leaving out all the unnecessary edges, which we know will not lead to a path path P with sound sequence s. In the end we are examining the last column of the constructed table, which will contain the constructed path for all the valid entries(non-zero). If all the columns have zero then no such path exists. And if the last column has valid entries (non-zero), then we find the entry with max product of $prob_j$ since they are the only paths of which one is our desired path P_{path}.

Time Complexity

The time complexity of the above mentioned algorithm is as follows,

For each cell in the table we need to examine all the out-ward going edges. If we look one column level at a time, then for every column we need to examine all the edges and vertices in the given graph. The total time taken to compute value of one column is $O(e+n)$, where n is the number of vertices in the graph and e is the number of edges. We totally have k columns, Hence the total time Complexity is $O(e+n) * k$.
However there is one difference between this and part a. Here in the last step we also iterate through the last column to find the maximum probable path which we store as the product of $prob_{tillNow}$ Since in worst case we may have n vertices so finding maximum value will take $O(n)$ time. However the complexity is,

$Time\ Complexity = O(e+n) * k.$

References

Chapter 22. Introduction to Algorithms by T. Cormen, C. Leiserson, R. Rivest, C. Stein