CSCE 629-601 Homework 1

Name: Adil Hamid Malla **UIN:** 425008306

1. Exercise 15.1-2 on page 370

Solution:
The solution to this problem is very trivial. Now lets see what our input is:

INPUT: We have a set of the prices p_i for some M lengths of the rod. Moreover we are given formula for calculating the density of each rod length ($d_i = \frac{p_i}{i}$).

OUTPUT: Distinction that solution using the DP is better than that from the Greedy Algorithm.
Now lets make a table of the length of rod and their corresponding prices and densities.

i	1	2	3	4
p_i	1	40	75	48
$d_i = \frac{p_i}{i}$	1	20	25	12

Going forward lets take this example to show that the greedy algorithm not necessarily provides the optimal solution to this problem. If we see according to the DP algorithm, we will be doing the maximization of the revenue (value in terms of price) by using the formula:

$$r_n = \max_{k=1,\cdots,n} \{p_k + r_{n-k}\}$$

Thus the solution we get is :

$$r_0 = 0;$$
$$r_1 = 1;$$
$$r_2 = 40;$$
$$r_3 = 75 = \max\{r_1 + r_2, r_3\};$$
$$r_4 = 80 = \max\{r_1 + r_3, r_2 + r_2, r_4\};$$

So the maximum revenue we got using the DP on prices of rod length 4 is 80.

Now lets see what our greedy algorithm will do with the same set of the input. The formula for greedy algorithm will be based on the density of the length of rod.

$$r_n = p_k \text{ (where } 0 \leq k \leq n \text{ is index at which } d_k \text{ has max value)} + r_{n-k}$$

Here we are making a greedy choice on every step, starting with first value, we will cut the rod length which has the highest density and then cut the remaining length again by making a greedy choice. So following this algorithm we get the below result:

$$r_0 = 0;$$
$$r_1 = 1;$$
$$r_2 = 40;$$
$$r_3 = 75 = r_3 \rightarrow d_3;$$

using the maximum density d_3

$$r_4 = 76 = r_3 + r_1$$

first selecting the maximum density d_3 and then remaining length rod d_1

So the maximum revenue we get by using the Greedy algorithm on densities of rod length 4 is 76.

So we proved by contradiction that the maximum revenue obtained from the greedy algorithm is not the optimal solution.

2. Exercise 15.7 on page 408

Solution:
INPUT: We are given a directed graph G=(V,E) for speech recognition. Each edge (u,v) $\in$ E is labeled with sound $\sigma(u, v)$ from a finite set $\sum$ of sounds. Each path starts from a distinguished vertex $v_0 \in V$.

OUTPUT: (**a**) Given an edge-labeled graph G with distinguished vertex v_0 and a sequence s = (σ_1 , σ_2, $\cdots$,σ_k) of sounds from $\sum$, we have to return a path in G that begins with distinguished vertex v_0 and has s as its label. If no such path exists then return NO-SUCH-PATH :NSP. Also analyze the time complexity of the algorithm.
Lets store the graph G in an adjacency matrix. So we will need V rows and V columns to store the same. Since we have the graph G labeled with weights equaling to the sounds produced, so we will store the directed graph with each entry representing sound $\sigma(u, v)$ from vertex u to v.
Now given a vertex v_0 we have to find a path following the set of sounds which we are already given in s. Since the solution can be obtained in various ways, brute-force is one such way where all the path can be traversed and stored and then compared to the given sound label to compare the matching one, if no such path is found then we can say no such path exists, but we have to provide an efficient algorithm which reduces the number of the path we traverse hence to make the algorithm fast.
Lets solve the question using the method we learned in class i.e. breaking a problem into smaller parts of it.

Algorithm 1 Path Tracing Algorithm

procedure RETURN-PATH(σ, V) ▷ Returns path if exists
  let $p[k, V]$ be a new array
  let $sigmatable[V, V]$ be adjacency matrix representation for our graph
  for $v = 0 \rightarrow V$ **do** ▷ initialization for starting with v_0
    if $v = v_0$ **then** $p[0, v] = 1$
    else $p[0, v] = 0$
    end if
  end for
  for $i = 0 \rightarrow k$ **do** ▷ Solving for path
    for $v = 1 \rightarrow V$ **do**
      for $u = 1 \rightarrow V$ **do**
        if $p[i-1, u] = 1$ and $sigmatable[u, v] = \sigma_i$ **then** $p[i, v] = 1$
This condition will be true when $p[i-1, u] = 1$ and $(u, v) \in E$ and also if $\sigma_i = sigmatable[u, v]$
        else $p[i, v] = 0$
        end if
      end for
    end for
  end for
  Path[j+1] be an array to store the path
  for $i = 0 \rightarrow k$ **do**
    for $v = 0 \rightarrow V$ **do**
      if $p[k, v] = 1$ **then** ▷ If last row has 1 in it means we have reached till last of the label and path exists
        if $p[i, v] = 1$ **then** Path[k-i] = v; ▷ traverse back to get the path- any path that is why we are continuing continue;
        end if
      else
        return NSP ▷ No such path exists
      end if
    end for
  end for
  $Path[0] = v_0$
  return $Path[]$ ▷ Path Exists
end procedure

So if i have to get a path from v_0 and using the labels (σ_1 , σ_2,σ_3), we can solve the problem by finding the solution by traversing the first edge from v_0 to v_1 considering a label σ_1 exists from v_0 to v_1 and $v_1 \in E$ and then, we solve the smaller problem of finding the path from v_1 using the remaining labels i.e. σ_2,σ_3. Mathematically we can solve it by using the following formula:

$$p(i,v) = \begin{cases} 1 \text{ , if } p(i-1,u) = 1, \forall u \in V, where\,(u,v) \in E \text{ and edge label is } \sigma_i \\ 0 \text{ , otherwise} \end{cases} \tag{1}$$

where $0 \leq i \leq k$, $1 \leq u \leq V$ and $1 \leq v \leq V$ (number of vertices in G).

Correctness of this solution is validated by the fact that if the path exists in the graph, then the bigger problem will have a solution by making the first step to next node and then solving the path finding problem for rest of the labels. Thus we are using the previous solution (*subproblem*) in finding the bigger solution. Solution to this problem is mentioned in Algorithm 1.
Now lets calculate the time complexity of our Algorithm. Since there are two things we are storing one is the graph and other is calculating the path matrix to find the path till end which is denoted by $p[i,v]$.
Since we populate the path matrix using the label given $0 \leq i \leq k$ and traverse all the vertices $1 \leq v \leq V$, moreover we also search for all previous vertex u for which the value is 1, then $1 \leq u \leq V$. So if previous row has all the ones (worst case scenario) then we have to traverse V^2 times. so considering we are calculating only the path matrix then time complexity is:

$$O(n) = O\left(kV^2\right)$$

Since populating the adjacency matrix is $O\left(V^2\right)$.

OUTPUT: (**b**) Now if every Edge has an associated non negative probability $pr(u,v)$ of traversing the edge (u,v) from the vertex u and thus producing the corresponding sound. Now using the algorithm above we have to find the path which we returned is most probable path starting at v_0 and having a label s. Also analyze the running time of the algorithm.
Now here only one thing is added to previous problem, since it was possible to get many paths in previous example now we have to find the most probable path, meaning that we are maximizing the probability of the path.
Here we can reuse the same algorithm, just we have to enhance the logic when we are populating the path matrix. Moreover, we have to use the probability table like that of adjacency list. Let $pr[u,v]$ be a matrix to store the probabilities of all the edges in the graph. Now our function to maximize the probability will be:

$$p(i,v) = \begin{cases} \max\left(p(i-1,u).pr(u,v)\right) \text{ , where } p(i-1,u) = 1, \forall u \in V, where\,(u,v) \in E \\ \text{and edge label is } \sigma_i \text{ and } pr(u,v) \text{ is probability of traversing edge. } (u,v) \end{cases} \tag{2}$$

where $0 \leq i \leq k$, $1 \leq u \leq V$ and $1 \leq v \leq V$ (number of vertices in G).
Here let us assume that we have a adjacency matrix $sigmatable[V,V]$ and also the table for probabilities, i.e. $pr[V,V]$. Now lets write the algorithm for the same:
Now lets calculate the time complexity of our Algorithm. Since there are three things we are storing one is the graph using adjacency matrix, the table for probabilities and last is calculating the path matrix to find the path till end which is denoted by $p[i,v]$.
Since we populate the path matrix using the label given $0 \leq i \leq k$ and traverse all the vertices $1 \leq v \leq V$, moreover we also search for all previous vertex u for which the value is 1, then $1 \leq u \leq V$. So if previous row has all the ones (worst case scenario) then we have to traverse V^2 times. so considering we are calculating only the path matrix then time complexity is:

$$O(n) = O\left(kV^2\right)$$

Algorithm 2 Maximize Probable Path Tracing Algorithm

procedure RETURN-MAXIMUM-PATH(σ, V, pr) ▷ Returns maximum probable path if exists
  let $p[k, V]$ be a new array
  let $sigmatable[V, V]$ be adjacency matrix representation for our graph
  let $pr[V, V]$ be table to store the probability values for each edge in our graph
  for $v = 0 \rightarrow V$ **do** ▷ initialization for starting with v_0
    if $v = v_0$ **then** $p[0, v] = 1$
    else $p[0, v] = 0$
    end if
  end for
  for $i = 0 \rightarrow k$ **do** ▷ Solving for path
    for $v = 1 \rightarrow V$ **do**
      for $u = 1 \rightarrow V$ **do**
        if $p[i-1, u] = 1$ and $sigmatable[u, v] = \sigma_i$ **then** $p[i, v] = \max(p[i-1, u].pr[u, v])$
This condition will be true when $p[i-1, u] = 1$ and $(u, v) \in E$ and also if $\sigma_i = sigmatable[u, v]$
        else $p[i, v] = 0$
        end if
      end for
    end for
  end for
  Path[j+1] be an array to store the path
  for $i = 0 \rightarrow k$ **do**
    for $v = 0 \rightarrow V$ **do**
      if $p[k, v] = 1$ **then** ▷ If last row has 1 in it means we have reached till last of the label and path exists
        if $p[i, v] = 1$ **then** Path[k-i] = v; ▷ traverse back to get the path- any path that is why we are continuing continue;
        end if
      else
        return NSP ▷ No such path exists
      end if
    end for
  end for
  $Path[0] = v_0$
  return $Path[]$ ▷ Path Exists
end procedure