CSE 548 Assignment 4

Divyansh Upreti (SBU ID: 112026646)

1 Prob 1- 3

1.1

The given algorithm will fail in the scenario presented in Fig 1. The presented algorithm will calculate the longest path from v_1 to v_7 as 3 (consisting of path $(v_1, v_2), (v_2, v_6), (v_6, v_7)$). It ignores the actual longest path (length 4) consisting of $(v_1, v_3), (v_3, v_4), (v_4, v_5), (v_5, v_7)$.

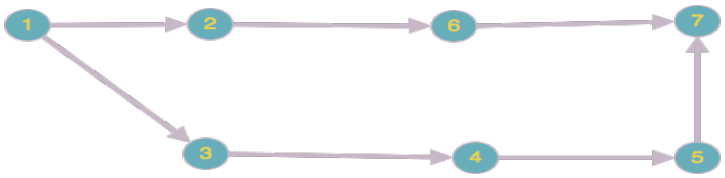

Figure 1: Example graph

1.2

The correct algorithm to calculate longest path using dynamic programming is below:

Algorithm 1 longestPath(n)

```
1: begin
2: DP(1,..,n)
3: DP(1)=0 (Distance from source to itself)
4: for i=(2,..n) do
5:     DP(i)=-INFINITY (Unreachable)
6:     for edges from j to i do
7:         DP(i) = max(DP(i),DP(j)+1)
8:     end for
9: end for
10: return DP[n]
```

Time complexity = $\mathcal{O}(n^2)$, n = number of vertices (as each node can have every other node connected to it).

This algorithm is correct and can be proved by contradiction. If we choose a middle vertex and suppose that optimal path would be passing through this vertex to reach longest path, then this makes this an optimal path to reach the destination but our algorithm should have picked up this vertex at a prior stage if this was true so by contradiction the longest path is correct.

2 Prob 1- 5

This algorithm has optimal substructure as the optimal segmentation of prefix of string y is also optimal. So following algorithm works:

Algorithm 2 bestSegmentation(n)

```
1: begin
2: DP(1,..,n)
3: DP(1)= Quality(first letter)
4: for i=(1,..n) do
5:     for j=(1,..i do
6:         DP(i) = max(DP(i),DP(j-1)+Quality(j to n)
7:     end for
8: end for
9: return DP[n]
```

Time complexity = $\mathcal{O}(n^2)$, n = number of letters

The algorithm is correct and can be proved such because to find out optimal segmentation for a string of i characters from a prefix of j characters, we need to simply add the quality of letters from j+1 to i (the last word to be appended) and as given above we are exhaustively checking this for each possibility in this characters range so we are guaranteed to get optimal solution.

3 Prob 1- 13

As we need to analyze the product of sequence of characters, to detect the cycle, we will make use of following logarithmic property:

Since:
$\log(a * b) = \log(a) + \log(b)$:
A trading cycle is an opportunity cycle when:

$\prod_{i=1,j=1}^{n} r_{ij} > 1$,where i ranges till n.
Hence: $\sum_{i=1j=1}^{n} \log(r_{ij}) > 0$
And so: $-\sum_{i=1j=1}^{n} \log(r_{ij}) < 0$

Thus, all we need to do is detect negative cycle which can be done easily by Bellman Ford algorithm.

Here as number of edges in total will be n. Time complexity of the problem will be equal to Bellman Ford's running time complexity of $\mathcal{O}(vertices * edges)$ i.e. $\mathcal{O}(n^2)$.

4 Prob 1-19

Let the string be s(1...n)
The signal string x' will be x'(1...n)
Ths signal string y' will be y'(1...n)

s is an interleaving of x' and y' iff the last character of s comes from either x' or y'. Removing the last character makes the problem of s(1...n) reduced to s(1...n-1) with prefixes of x' and y' as search space. So we need to make s(1...n-1) optimal to find the larger optimal soultion. Thus, this problem can be solved by dynamic programming.

Algorithm:

Algorithm 3 checkInterleaving(s,x',y')

```
1:  begin
2:  DP(1...n)(1...n)
3:  DP(0,0)= True(Trivial base case)
4:  for k=(1...n) do
5:      for i=(1...n) and j=(1...n) do
6:          if (i+j==k) and DP(i-1,j)=True and s[i+j]=x'[i] then
7:              DP(i,j)=True
8:          else if (i+j==k) and DP(i,j-1)=True and s[i+j]=y'[j] then
9:              DP(i,j)=True
10:         else
11:             DP(i,j)=False
12:
13:
14:             for i=(1...n) do
15:                 for j=(1...n) do
16:                     if (i+j)==n and DP(i,j)==True then
17:                         return True
18:
19:
```

Time complexity = $\mathcal{O}(n^2)$,

The correctness of algorithm can be proved inductively. To find optimal solution for s(1...n), we are using optimal solution of smaller problem s(1...n-1) and appending it to a character from x' and y'

5 Prob 1-24

Let A votes in precinct belong to A party
Let B votes in precinct belong to B party

This problem has a optimal substructure because to solve for $(i+1)^{th}$ precinct (problem of the form DP(i+1,V,a1,a2), where first parameter is precinct, second parameter is precincts already allocated to A, and third and fourth denote the votes of A in district 1 and district 2, inductively:
We need to solve for DP(i,V-1,a1-v,y)
and to solve for $(i+1)^{th}$ precinct in district 2:
We need to solve for DP(i,V,a1,a2-v)

We build up the table and if there is any entry which served our purpose of dividing voters into two divisions (solution for problem (DP(n,n/2,a1,a2)), where a1 and a2 greater than m*n/4 as true then we have found out the solution.

Time complexity will be $\mathcal{O}(n^2m^2)$ as total number of iterations in finding out the solution will be n^2m^2, where n is the number of precincts and m is the total number of voters in each precinct.