ADA HW6

Kushagra Arora 2015049 Akarsha Sehwag 2015010

22 March 2017

1 Question 1

We can use **dynamic programming** to solve this problem.
Let $dp[i][j]$ denote the maximum numbers of apples that can be collected while reaching $a_{i,j}$ from $a_{0,0}$ i.e. the top left corner. A is the matrix with number of apples in each cell.

Setting up initial conditions:
For the first row, we can just move right.

$$dp[0][j] = \sum_{k=0}^{j} a_{0,k}$$

Similarly, for the first column.

$$dp[i][0] = \sum_{k=0}^{i} a_{k,0}$$

Now for all the other nodes, we need to take maximum of going right or going down.

$$dp[i][j] = max\{dp[i-1][j], dp[i][j-1]\} + a_{i,j} \forall i > 1, j > 1$$

Now, the answer to the given question is $dp[n-1][m-1]$.

2 Question 2

Let $T(i,j)$ represent the number of ways to get True for a subexpression between i and j.
Let $F(i,j)$ represent the number of ways to get False for a subexpression between i and j
Let $Total(i,j) = T(i,j) + F(i,j)$

$$T(i,j) = \sum_{k=i}^{j-1} \begin{cases} T(i,k) * T(k+1,j) & \text{if } operator k^{th} position is and \\ Total(i,k) * Total(k+1,j) - F(i,k) * F(k+1,j) & \text{if } operator k^{th} position is or \\ T(i,k) * F(k+1,j) + F(i,k) * T(k+1,j) & \text{if } operator k^{th} position is xor \end{cases}$$

$$F(i,j) = \sum_{k=i}^{j-1} \begin{cases} Total(i,k) * Total(k+1,j) - T(i,k) * T(k+1,j) & \text{if } operator k^{th} posiitonis and \\ F(i,k) * F(k+1,j) & \text{if } operator k^{th} potitionis or \\ T(i,k) * T(k+1,j) + F(i,k) * F(k+1,j) & \text{if } operator k^{th} position is xor \end{cases}$$

3 Question 3

Let $Cost(A, i)$ give the minimum cost of the path that travels each vertex in S exactly once starting from 1. We will have vertex 1 in every A.

$$Cost(A,i) = \begin{cases} dist(i,1) & \text{if } size(A) = 2 \\ min(Cost(A - \{i\}, j) + dist(j,i) \forall j! = i, j! = 1 & \text{if } size(A) > 2 \end{cases}$$

4 Question 4

Proof by contradiction. Let us consider that there exists a connected graph such that it does not have a spanning tree.

That means there exists no subgraph of the graph that has no cycles and is connected.

Let us consider the fact that there exists no subgraph that has no cycles. Now, an edge can be removed from each cycle to make it acyclic without making it disconnected(Since a cut-edge is a part of no cycles.).

Now, if our graph is connected, we can apply above conjecture to say that we can remove all cycles to make it a tree.

Thus, every connected graph has a spanning tree.

5 Question 5

Let us consider, the i^{th} node of the graph, v_i. $Nv_i = \{v_{i-2}, v_{i-1}, v_{i+1}, v_{i+1}\}$

We can see that greedily choosing the edge for v_i, we can choose the edge with minimum weight i.e. v_{i-2}.

Such that the weight added to the total weight is $i + i - 2 = 2i - 2$

Also, for v_2, we need to add the edge $e_{1,2} = 3$.

Thus our total weight becomes:

$$(\sum_{i=3}^{i=n} 2i - 2) + 3 = (n+1)(n-2) + 1 = n^2 - n + 1.$$

6 Question 6

Let T_1 and T_2 be two distinct minimum spanning trees.

Now, let $e = PQ \in T_1, e \notin T_2$ such that e is of minimum weight. $\Rightarrow T_2 \cup e$ has a cycle. This means, that T_2 contains a path from P to Q which does not include e. If T_1 contains all the edges on the path from P to Q, it will form a cycle. Thus, there exists atleast one edge f such that $f \notin T_1$. Since, all edges have different weights, we can say that $w(e) < w(f)$. So, in T_2, we can replace f by e to obtain a spanning tree of lesser weight. This is a contradiction.

7 Question 7

Proof of correctness of reverse delete algorithm:

First we prove that the subgraph produced by the algorithm is a spanning Tree. We know that, the algorithm deletes an edge if it does not disconnect the graph. That means, at the end of the algorithm, we are left with a subgraph which is connected, and has no cycles. Since, if it had any cycle. The algorithm would remove the max-edge of the cycle and it would still remain connected.

Secondly, we need to prove that we get the minimum spanning tree. Let T be the MST of graph G. Let us say that the algorithm produces $T^{'}$ as the solution. Let e be the edge deleted at step i. We have 2 cases :
Case 1 : $e \notin T \Rightarrow T^{'} = T$ is an MST of G. Case 2 : $e \in T$. But e does not produce a diconnectedness in G. However, e disconnects T into t_1, t_2. However, since subgraph is connected there is an edge f that connects t_1, t_2. Using the proof given in the next answer, we say that $T^{'} = T - e + f$ is a spanning tree of G.

1. $w(f) < w(e)$. $\Rightarrow T'$ has a weight lesser that T, which is not possible.
2. $w(f) > w(e)$. This is also not possible, since in the algorithm we would have found f before e and since it is a part of cylce, it is not a cut-edge. We would have removed it.
3. $w(f) = w(e)$. Weight of T' = Weight of T. Thus, T' is the MST.

8 Question 8

Let T be a tree of N nodes. $\Rightarrow$ No. of edges in $T = N - 1$. No. of edges in $T^{'} = T \cup (u, v) - (x, y) = N - 1 + 1 - 1 = N - 1$. Hence, to prove that $T^{'}$ is a tree. We, need to prove that it is connected. Let us prove this by contradiction. Let us suppose that $T^{'}$ is disconnected. We know that $T^{'}$ contains exactly 1 cycle(say C). Now, $(x, y) \in C$. Since, $T^{'}$ is disconnected, (x, y) is a cut-edge. Since, no cut edge is a part of any cycle. $\Rightarrow (x, y) \notin C$. This is a contradiction.