homework 4

Name: Yifan Zhang

University ID: yz113

Problem 1

We can use DP to solve this problem. $OPT(i)$ means the number of rounds will need when we want to distribute message to all nodes in the sub-tree whose root node is node i. The sub-problem is:

$$OPT(i) = max\{OPT(m)+1, OPT(n)+1\}, \text{(m, n are node i's children)}$$

There are n sub-problems and every sub-problem will take $O(1)$. Overall, the time complexity is $O(n)$.

Problem 2

First we can compute the if this tree is true or not by simply recursively applying the operator to the values of the children. This will be done in $O(n)$. If the root of a tree evaluates to true, then we return 0. If the root of a tree evaluates to false, then we'll need DP to find the cost.

The sub-problem is: For each node we want to compute its cost. If this node is a and node, then we add its children's cost and that will be this and node's cost. Otherwise, for or node, we chose the less cost among its two children and use it as this or node's cost. For leaves, if it is True, then its cost is 0; if it is False, then its cost is 1. This will be done is $O(n)$ too because there will only be n node we need to compute the cost.

Overall time complexity is also $O(n)$.

Problem 3

We can use DP to solve this problem. The sub-problem is: For every node V_i, we want to compute the number of paths from s to V_i, and let's say this number is N_i. Then we have $N_i = \sum_{(V_j,V_i)\in E} N_j$. And for base case, we have $N_S = 1$. Then we can recursively call this algorithm from V_t and we will finally got the N_t.

Time complexity is: We have n sub-problem but every edge will be use only once in the whole algorithm. As a result, the time complexity should be $O(m)$. But if we need to consider the time that we sort every edge based on its pointing node and this will take $O(m+n)$.

Overall, if we have sorted edge, the time complexity is $O(m)$. Otherwise, the time complexity will be $O(m+n)$.

Problem 4

We will need 3 array to solve this problem. The three arrays are:

1. $M_{i,j}$, an alignment which ends with a matching.
2. $X_{i,j}$, an alignment which ends with a gap on X.
3. $Y_{i,j}$, an alignment which ends with a gap on Y.

Then the sub-problem is:

$$\begin{aligned} M_{i,j} &= \alpha_{X_i,Y_j} + min\{M_{i-1,j-1}, X_{i-1,j-1}, Y_{i-1,j-1}\} \\ X_{i,j} &= min\{(\alpha_0+\alpha_1+M_{i-1,j}), (\alpha_1+X_{i-1,j}), (\alpha_0+\alpha_1+Y_{i,j-1})\} \\ Y_{i,j} &= min\{(\alpha_0+\alpha_1+M_{i,j-1}), (\alpha_1+Y_{i-1,j}), (\alpha_0+\alpha_1+X_{i,j-1})\} \end{aligned}$$

And the final answer is $min\{M_{n,m}, X_{n,m}, Y_{n,m}\}$.

Time complexity: we have $m \times n$ sub-problems and every problem finish in $O(1)$. Overall, the time complexity is $O(m \times n)$.

Problem 5

We can use DP to solve this problem. The sub-problem is:

$$\begin{gathered} f(i,j) = min\{f(k,j-1) + (S_i - s_k)^2, k \in [1,i)\} \\ f(0,0) = 0 \end{gathered}$$

$f(n,k)$ is $\sum_{i=1}^{k}[max(S_i) - min(S_i)]^2$. For jth cluster, if the $j+1$'s cluster starts from k, and $f(i,j) == f(i,j-1), i \in (0,k']$. Then the jth cluster contains $(S_{k'}, S_k]$. And the kth cluster ends with S_n, $1st$ cluster starts from S_1.

Time complexity. There are $O(n^2)$ sub-problems and every sub-problem will take $O(k)$, so the overall time complexity is $O(n^2k)$.

Problem 6

We will use DP to solve this problem. The sub-problem can be descript like this($OPT(i,j)$ means the highest reward among $[snail_i, snail_j]$):

$$OPT(i,j) = \begin{cases} 0, & j \le i \\ OPT(i+1,j), & \\ OPT(i,k-1) + M[i,k] + OPT(k+1,j), \quad i \le k \le j & else \end{cases}$$

We can start from $OPT(n,n)$, and when we go down to $OPT(1,n)$, then we have got the answer.

Time complexity. There are $O(n)$ sub-problems and every sub-problem, suppose it will take $T(j)$ time, and we have:

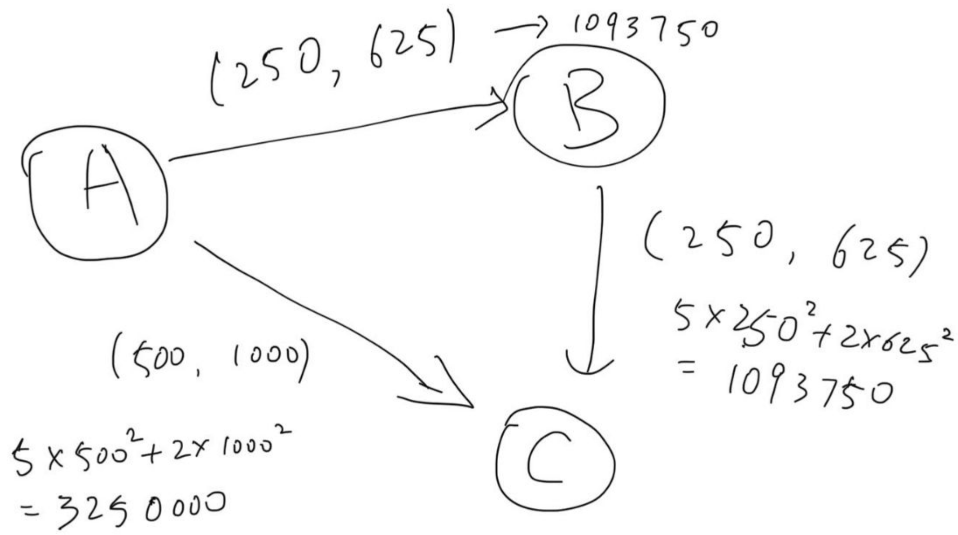
$$T(j) = T(k) + 2, k < j$$

This means that $OPT(i, j)$ make take $O(n^2)$ time.

As a result, overall it will take $O(n^3)$.

Problem 7

1. In some cases, Arthur may not respect the constraints $M < 500$ and $T < 1000$. Here is a counter example:



   Arthur will go from A to B then to C, instead of directly go to C. As a result, he will spend more than 1000 hours to go to his destination.

2. We can use DP to solve this problem. OPT(i, j) is the best choice from node i to node j. The sub-problem is:

$$OPT(i,j) = \begin{cases} 1, & \text{node j is Cambridge} \\ max\{OPT() & u \in V \wedge (u,v) \in E\} \end{cases}$$