CS 515
Tenghuan Li933638707; liten@oregonstate.edu
Midterm

Problem1:

For this problem, we can know that n is spaces in a long row, E is meaning empty, and C is meaning customer.

Assume that S(0) =1.

So for this question, the recursion:

```
S(n){
    If n ==0;
        return 1;
    If n ==1;
        return 2;
    return S(n-1)+S(n-2); (n>1)
}
```

For this question, we can know is same as Fibonacci question.

T(n) = T(n-1)+T(n-2)+O(1);

Proof:

For this question, we can know that we assume if n=0, S(0)=1;

When n=1; we can get 2 result is {E, C}; if we want to know n spaces can get how many result, the result is equal S(n) = {S(n-1)+E} + {S(n-2)+EC}.

For example, when n=3, S(3)= S(2)+S(1)=5, the possibilities is when S(1) = {E,C} at the end + EC, so we can get 2 possibilities that {EEC, CEC}. S(2) = {EE,EC,CE}the possible is that S(1) at the end + E, {EEE, ECE, CEE}. When it makes together is that {EEE, EEC, ECE, CEC, CEE}.

Problem2:

For the question, we can know that two rules: 1: can only wear Ci over Cj, if Ci> Cl; 2: some pairs of coats have incompatible materials, so they cannot be in direct contact.

Assume that coat is C = {C1, C2…., Cn}. So, we need to start with Cn-1.

For Cn-1, we need to compare its material with Cn, if incomp[i,j] = incomp[j,i]=0, we can know that they can wear together. So, we create a new list call R. We save the all possible with Cn-1 answer into Rn-1.

Then, we go to the Cn-2, we need to compare its material with Cn and Cn-1, if incomp[i,j] = incomp[j,i]=0. Same as the Cn-1, they can wear together, so put all Cn-2 possible answer in to Rn-2.

To sum up, when we run start at Cn-1 to C1, we also get all the possible answers and save in Rn-1 to R1. When we want to solve the problem, we only need to find the maximum value from R to be the answer to the problem. That is the maximum coats I can wear.

Problem 3:

For this problem, we can know that T is a binary rooted tree with positive, negative or zero weights on its edges.

So, we assume that root is 0, the edges call V. In each node, we save the maximum value of the edge of its left subtree or right subtree is recorded.

```
T(tree t){
    root ←0;
    If this node has no child nodes;
        return 0;
    If this node has left subtree;
        int left = T(t-> left)+ V(t-> left);
    If this node has right subtree;
        int right = T(t-> right)+ V(t-> right);
    if(left >= right && left>0)
        return left;
    else if(right >= left && right >0)
            return right;
    else
        return 0;
}
```

The time complex is O(n), because this is going through all the nodes of the binary tree.

Proof:

If the all weights are negative, so we just return the roof =0, is the max. When we go through the all nodes, we get the max weight in this tree, and return it back to the recursive. Because every time we return the maximum weight, when we go back to the root node we get the value, it must be the weight of the longest path.

Problem 4:

For this question, we need to find the longest subset of L in which no pair of segments intersects. And, we know that each pair of line's endpoints on y=0 and y=1 is distinct. So, assume that each line segment l in the set L= {a, b, c,….}, set the end points at y=0 is S0 = {Xa,Xb,Xc…}, set the end pints at y=1 call S1 = {Xa, Xb, Xc….}.

We want to find the longest subset of L which no pair of segments intersect. So, we just need to find the largest common sequence of S0 and S1. Call LCS{S0, S1} is the longest subset of L in which no pair of segments intersects. We know S0 and S1 length is same =n.

Pseudo algorithm:

```
LCS(S0, S1){
    int c[n][n];
    for i = 1 to n
        for j=1 to n{
            if (S0.[i] ==S1.[j])
                c[i][j]= c[i-1][j-1]+1;
            else
```

```
                c[i][j] = max(c[i-1][j], c[i][j-1]);
            }

        return c[n][n];
    }
```

Since this pseudo algorithm, we can know it two for loop. Because each c[i][j] is calculated in constant time, this array is n*n elements.

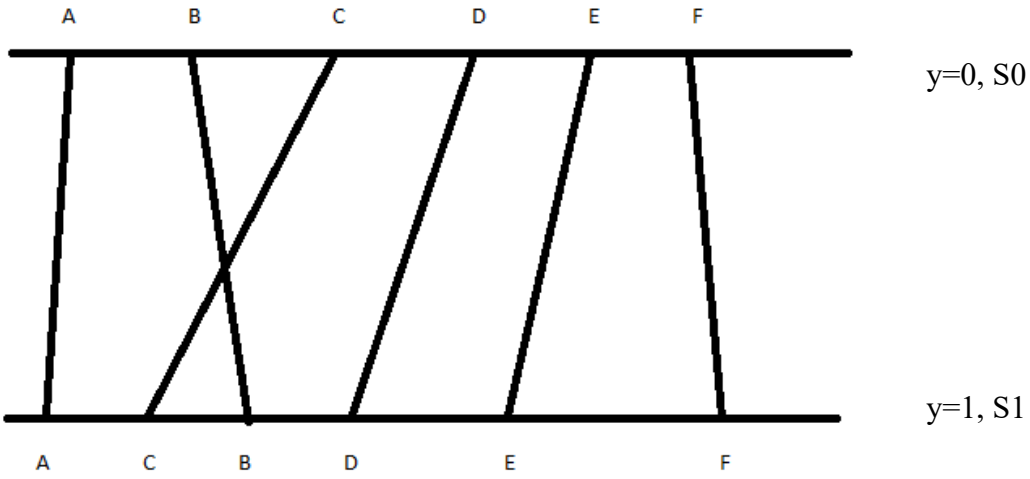
T(n)= O(n^2).

Proof:

When c[i][j] = c[i-1][j-1] + 1; (s0[i] == s1[j]), assume c[i][j] > c[i-1][j-1] + 1, then we can get the paradoxical result that c[i-1][j-1] is not the longest result.

When s0[i] != s1[j], assume c[i][j] > max(c[i-1][j], c[i][j-1]); then we can get c[i][j] > c[i-1][j], so the s0[i] it must in the c[i][j] sequence.

Because s0[i] != s1[j], so s1[j] it must not in the sequence of c[i][j]. So, this assume can get the c[i][j] = c[i][j-1]. This result contradicts the hypothesis.



As the picture show, S0 = {A,B,C,D,E,F} and S1= {A,C,B,D,E,F}, So the longest of no pair of segments intersects is L=5 ({A,B,D,E,F})