B02303042 經濟四 鄭雅文

第一題

(a) 前三行 if 的判斷式為初值的設定，後面的 else 是計算非初值的 C(n,m)的值，若 com[n-1][m-1]的值尚未被設定，因 C(n,m) = C(n-1,m) + C(n-1,m-1)，所以便到 combiRec(n - 1, m, com) 和 combiRec(n - 1, m - 1, com)遞迴尋找，但是若 com[n-1][m-1]的值被設定了，直接 return com[n-1][m-1]，複雜度為 O(1)。雖然有遞迴搜尋的步驟，但是表格只會被填一次，且表格的值不會再被更改，所以可解決重複計算的問題。

(b)

```
int** com = new int*[n];
for(int i = 0; i  <  n; i++)
{
    com[i] = new int[m]; //allocate memory to each row.
    for(int j = 0 ; j < m ; j++ ){
        com[i][j] = -1; //set each element to "-1".
    }
}
// declare an n by m dynamic array
// and initialize all elements in it to -1
```

(c)

```
int combiRec(int n, int m, int** com)
{
        for(int i = 0 ; i < n ; i++ ){
                if(i < m)
                        com[i][i] = 1;
        }
//      if (n==m) return 1

        for(int i = 0; i < n ; i++){
                        com[i][0] = i+1;
        }
//      if (m==1) return n

        for(int i = 1;i <= n ; i++)
                for(int j=1;j<=m;j++){
                        if( i > j ){
                                if(com[i-1][j-1] == -1){
                                com[i-1][j-1] = com[i-2][j-1] + com[i-2][j-2];
                                }
                        }
                }
//      filling the table
        return com[n-1][m-1];;
}
```

(d) 將(a)小題展開之後，會得到跟(c)小題相同的 n*m 的 matrix，所以以時間複雜度來說是相同等級的複雜度，(a)小題的好處是只用到需要的 com[i][j]，但壞處是每次遞迴呼叫需要用到 stack 保存變數；(c)小題的好處是不用用到遞迴呼叫可以減少一些執行時間，但是可能建到不需要用到的 com[i][j]

第二題

(a)

```
建一距離表格 distance[m][n]，其中 distance[i][j]表零售店 i 到物流中心 j 的距離
while(j*存在)
        找的到 j*的零售店的 Count = 0
        for 零售店 i = 1 to m
                minDistance = 400
                minCenter = -1
                maxRemain = 0
                if( 零售店還有需求)
                        for 物流中心 j = 1 to n
                                if(物流中心還有剩餘容量 && 銷貨毛利為正)
                                        找的到 j*的物流中心的 Count++
                                        if(distance[i][j] < minDistance)
                                                minDistance = distance[i][j]
                                                minCenter = j
                                                maxRemain = j 的剩餘容量
                                        else if(distance[i][j] == minDistance
                                                && j 的剩餘容量 > maxRemain)
                                                minCenter = j
                                                maxRemain = j 的剩餘容量
                                        end of if
                                end of if
                        end of for
                        j_i* 為 minCenter
                end of if
        end of for
        if(找的到 j*的零售店的 Count == 0)
                設 i*存在為 false
        end of if

        minDistance = 400
        minStore = -1
        maxNeed = 0
        for 零售店 i = 1 to m
                if( j_i* != -1)
                        if(distance[i][ j_i*] < minDistance)
                                minDistance = distance[i][ j_i*]
```

```
                                    minStore = i
                                    maxNeed = i 的需求數量
                            else if(distance[i][ j_i^* ] == minDistance && i 的需求數量 > maxNeed)
                                    minStore = i
                                    maxNeed = i 的需求數量
                            end of if
                    end of if
            end of for
            i*即為 minStore
            if(零售店 i*的未滿足需求 > 物流中心 j*的剩餘容量)
                補貨數量 = 物流中心 j*的剩餘容量
            else
                補貨數量 = 零售店 i*的未滿足需求
            end of if

            利潤 += (價格-成本*距離) * 補貨數量
            未滿足需求 -= 補貨數量
            零售店 i*之需求數量 -= 補貨數量
            物流中心 j 之剩餘容量 -=補貨數量
end of while
```

(b)

```
int main()
{
        int r = 3;
        int** array = new int*[r];
        for(int i = 0; i < r; i++)
        {
                array[i] = new int[i + 1];
                for(int j = 0; j <= i; j++)
                        array[i][j] = j + 1;
        }
        print(array , r); // later
        for(int i = 0; i < r; i++){
                delete [] array[i];
        }
        delete [] array;// some delete statements
return 0;
}
```