計算方法設計hw1

105021120 朱世耘

程式註解:

```
void next_comb(int choose[],int n,int m){      //找出Cn取m中的下一個組合
    for(int i=m-1;i>=0;--i){
        if(choose[i]<n+i-m+1){              //尋找choose中還能再加的index（從後往前）
            ++choose[i];                    //把該項+1
            for(int j=i+1;j<=m;++j){
                choose[j]=choose[i]+j-i;      //加完後把該項後的每一項都依序為該項+1
            }
            break;
        }
    }
}
int main() {
    int T;
    int N,K,S;
    cin>>T;                                 //輸入T
    for (int t=0; t<T; t++) {
        int sum=0;
        cin>>N>>K>>S;                       //輸入N,K,S
        int A[N][S];                        //用A來存放輸入資料
        int max[S];                         //用max存放每種element個別的最大值
        for (int i=0; i<S; i++) {
            max[i]=0;
        }
        for (int i=0; i<N; i++) {
            for(int j=0;j<S;j++){
                cin>>A[i][j];
            }
        }
        if (K>=S){                                  //當K>=S,只要把每種element在全部
            for (int i=0; i<N; i++) {               //object中的最大值挑出來相加即可
                for (int s=0; s<S; s++) {       //也就是說不用考慮到挑選出的是哪K個element
                    if (A[i][s]>max[s]){
                        max[s]=A[i][s];
                    }
                }
            }
            for (int s=0; s<S; s++) {
                sum = sum + max[s];             //把每個element的最大值相加到sum
            }
        }
        else{
            // Do K<S situation
            int sum1=0;
            int big[S];
            int choose[K];
            for (int i=0; i<K; i++) {
                choose[i]=i;                    //初始化choose的內容為0到K-1
            }                                   //為第一個組合
            while (choose[0]<=(N-K)) {          //當choose[0]>N-K時代表都跑過了
                for (int i=0; i<K; i++) {
                    for (int s=0; s<S; s++) {
                        if (A[choose[i]][s]>max[s]){
                            max[s]=A[choose[i]][s];    //整理每一種組合下的max
```

```
                        }
                    }
                }
                for (int s=0; s<S; s++) {
                    sum1 = sum1 + max[s];     //各項element的max相加到sum1
                }
                if (sum1>sum) {
                    sum=sum1;                 //若此種組合比之前組合的max大，則覆蓋
                }
                next_comb(choose, N, K);      //跳到下一個組合
                sum1=0;
                for (int i=0; i<S; i++) {
                    max[i]=0;                 //歸零
                }
            }
        }
        cout<<sum<<endl;                      //輸出結果
    }
}
```

Time complexity:

next_comb(int choose[m],int n,int m):

O(m)*O(m)=O(m^2)　　//兩個for迴圈

int main():

O(T)　*(　　O(S) + O(N*S)　+兩種case)
執行T次｜初始化max ｜輸入｜

case K>=S:

O(N*S)　　+ O(S)　　=O(N*S)
雙重for迴圈｜for迴圈

case K<S

O(K)　　+O(C N取K) *(O(K*S)+O(S)+O(K^2)　　+O(S))
for迴圈｜while迴圈｜ for迴圈　　｜next_comb｜for迴圈
＝O(K)+O(N*K)*(O(K*S)+O(K^2))
=O(K)+O(N*K*K)*(O(S)+O(K))
=O(N*K*K)*(O(S)+O(K))　　//because S>K
=O(N*S^3)

total:
O(T)*(O(S)+O(N*S)+O(N*S)+O(N*S^3))=O(T*N*S^3)