題目：
請分別使用 DP 與 Greedy 作法分別解決此問題，並比較 n 大於多少時，兩方法 的執行時間有明顯差異

Dp解法：
在table[i][j]=1~i的活動中得到profit j 的最少總活動時間

遞迴式：

```
table[i][j]=   ∞                                         if i<j
               min{table[i-1][j],table[i-1][j-1]+t[i]}           if table[i-1][j-1]+t[i]<d[i]
               table[i-1][j]                             else
```

程式碼：

```
#include <iostream>
#include <vector>
#include <limits.h>
using namespace std;

typedef struct activity{
    int time;
    int deadline;
    int order;
}activity;

bool compare(activity a ,activity b){
    return a.deadline<b.deadline;

}

void Scheduling(activity act[],vector<vector<int>> table,int i,int j){
    if (i==0 || j==0){
        return;
    }
    if(table[i][j]==table[i-1][j-1]+act[i-1].time){
        Scheduling(act,table,i-1,j-1);
        if(act[i-1].time<act[i-1].deadline){
        printf("%d ",act[i-1].order);
        }
    }else if(table[i][j]==INT_MAX){
        Scheduling(act,table,i,j-1);
    }
    else{
        Scheduling(act,table,i-1,j);
    }


}

void working(activity act[],int n){

    vector<int> answer;
    sort(act, act+ n,compare);
    vector<vector<int> > table(n+1,vector<int>(n+1,0));


    for(int i=0;i<=n;i++){
        for(int j=0;j<=n;j++){
            if(i<j){
        table[i][j]= INT_MAX;
            }
        }
    }


    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
           if(table[i-1][j-1]+act[i-1].time<=act[i-1].deadline) {
               if(table[i-1][j-1]!=INT_MAX){
                table[i][j]=min(table[i-1][j],table[i-1][j-1]+act[i-1].time);
               }else{
                   table[i][j]=INT_MAX;
               }
            }else{
                table[i][j]=table[i-1][j];
            }

        }
    }
    Scheduling(act,table,n,n);

}


int main(int argc, const char * argv[]) {
    int n,p,d;
    scanf("%d",&n);
    activity *act=(activity*)calloc(n,sizeof(activity));
    for (int i=0; i<n; i++) {
        scanf("%d",&p);
        act[i].time=p;
        scanf("%d",&d);
        act[i].deadline=d;
        act[i].order=i+1;
    }
    working(act, n);


    return 0;
}
```

greedy解法：

先把活動按照deadline排序，然後從第一個開始加入answer這個vector，判斷如果pi超過di就不考慮，而如果加入第k個活動時，總運行時間超過k的deadline，就在answer裡找出最大的pi，然後把它刪掉。

程式碼：
```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

typedef struct activity{
    int time;
    int deadline;
    int order;
}activity;


bool compare(activity a ,activity b){
    return a.deadline<b.deadline;

}

void Scheduling(activity a[], int n)
{
    vector<activity> answer;
    sort(a, a + n,compare);
    int totaltime=0;

    for(int i =0; i<n;i++){
        if(a[i].time>a[i].deadline){
            continue;
        }
        answer.push_back(a[i]);
        totaltime += a[i].time;

        if (totaltime >a[i].deadline) {

            int max=answer[0].time,index = 0;

            for(int j=1;j<answer.size();j++){
                if(answer[j].time>max){
                    max=answer[j].time;
                    index=j;
                }
            }

            totaltime-=max;
            answer.erase(answer.begin()+index);
        }

}
    for (int i=0; i<answer.size(); i++) {
        printf("%d ",answer[i].order);
    }

}

int main(int argc, const char * argv[]) {
    int n,p,d;
    scanf("%d",&n);
    activity *act=(activity*)calloc(n,sizeof(activity));
    for (int i=0; i<n; i++) {
        scanf("%d",&p);
        act[i].time=p;
        scanf("%d",&d);
        act[i].deadline=d;
        act[i].order=i+1;
    }
    Scheduling(act,n);

    return 0;
}
```

測試運行時間：

n	DP(per_second)	GREEDY(per_second)
1	0.000032	0.000077
5	0.000087	0.000048
10	0.000169	0.000072
15	0.000379	0.000096
25	0.000764	0.000052
30	0.001025	0.000040
40	0.003814	0.000050
50	0.003898	0.000061
70	0.014282	0.000099
80	0.011497	0.000051
100	0.025366	0.000065

經由上述實驗，我觀察到是n>10以後，Greedy方法的速度都一直是快於DP的解法，Greedy都是維持差不多的速率，而兩者的差距一直擴大。