算法

第八章 贪心算法

圣诞老人

题解

圣诞老人的礼物-Santa Clau’s Gifts(百练4110)

圣诞节来临了，圣诞老人准备分发糖果，现在有多箱不同的糖果，每箱糖果有自己的价值和重量，每箱糖果都可以拆分成任意散装组合带走。圣诞老人的驯鹿雪橇最多只能装下重量W的糖果，请问圣诞老人最多能带走多大价值的糖果。

输入

第一行由两个部分组成，分别为糖果箱数正整数n(1 <= n <= 100)，驯鹿能承受的最大重量正整数w（0 < w < 10000），两个数用空格隔开。其余n行每行对应一箱糖果，由两部分组成，分别为一箱糖果的价值正整数v和重量正整数w，中间用空格隔开。

输出

输出圣诞老人能带走的糖果的最大总价值，保留1位小数。输出为一行，以换行符结束。

> 解法：
>
> 按礼物的价值/重量比从大到小依次选取礼物，对选取的礼物尽可能多地装，直到达到总重量w
>
> 复杂度： O(nlogn)

code c++

```
#include<iostream>
#include<algorithm>
#include<iomanip>
#include<cstdio>
#include<vector>
using namespace std;

const double eps = 1e-6;
class Candy
{
public:
    int v, w;
    bool operator<(const Candy &c)const{
        return double(v) / w - double(c.v) / c.w>eps;
    }
};
int main(){
    vector<Candy>candies(110);
    int n, w;
    cin >> n >> w;
    for (int i = 0; i < n; ++i){
        cin >> candies[i].v >> candies[i].w;
    }
    sort(candies.begin(), candies.begin()+n);
    int totalW = 0;
    double totalV = 0;
    for (int i = 0; i < n; ++i){
        if (totalW + candies[i].w <= w){
            totalW += candies[i].w;
            totalV += candies[i].v;
        }
        else{
            totalV += candies[i].v*double(w - totalW) / candies[i].w;
            break;

        }
    }
    printf("%.1f",totalV);
    system("pause");
    return 0;
```

```
}
```

测试用例

样例输入

4 15

100 4

412 8

266 7

591 2

样例输出 1193.0

看电影

题解

例题：电影节 (百练4151)

大学生电影节在北大举办! 这天，在北大各地放了多部电影，给定每部电影的放映时间区间，区间重叠的电影不可能同时看（端点可以重合），问李雷最多可以看多少部电影。

例题：电影节(百练4151)

输入

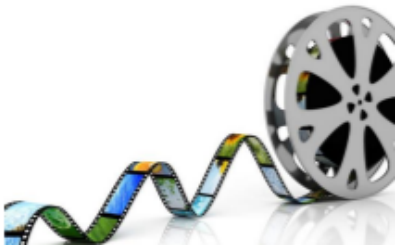
多组数据。每组数据开头是n(n<=100)，表示共n场电影。
接下来n行，每行两个整数(均小于1000)，表示一场电影的放映区间
n=0则数据结束

输出

对每组数据输出最多能看几部电影

贪心解法

将所有电影按结束时间从小到大排序，第一步选结束时间最早的那部电影。 然后，每步都选和上一部选中的电影不冲突且结束时间最早的电影。



复杂度： O(nlogn)

code c++

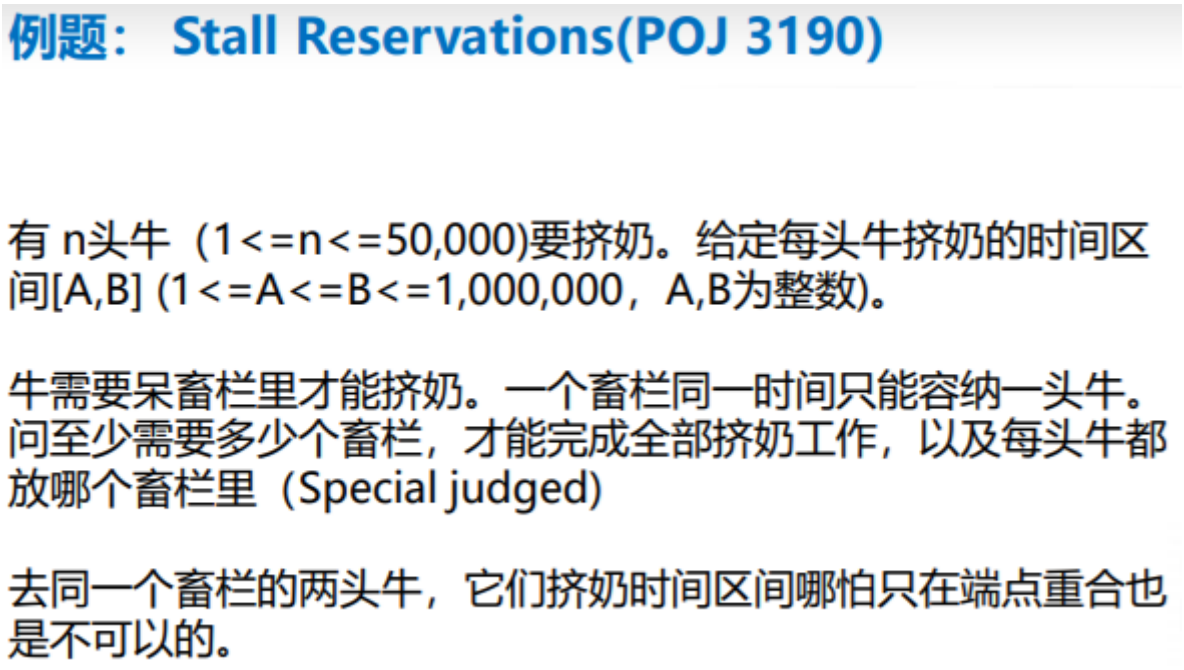
```
#include<iostream>
#include<algorithm>
#include<iomanip>
#include<cstdio>
#include<vector>
using namespace std;
const double eps = 1e-6;
struct Move{
    int start;
    int end;
    bool operator<(const Move &c)const{
        return  end - c.end < eps;
    }
};
void main(){
    vector<Move>move(110);
    int n;
    cin >> n;
    for (int i = 0; i < n; ++i){
        cin >> move[i].start >> move[i].end;
    }
    int res = 1;
    sort(move.begin(), move.begin() + n);
    int end =0 ;
    for (int i = 0; i < n; ++i){
        end = move[i].end;
        while (i < n){
            if (end <= move[i + 1].start){
                ++res;
                break;
            }
            else ++i;
        }
    }
    cout << res;
    system("pause");
}
```

样例输入

```
12
1 3
3 4
0 7
3 8
15 19
15 20
10 15
8 18
6 12
5 10
4 14
2 9
out is 5请按任意键继续. . .
```

喂奶牛

题解



贪心解法：

所有奶牛都必须挤奶。到了一个奶牛的挤奶开始时间，就必须为这个奶牛找畜栏。因此按照奶牛的开始时间逐个处理它们，是必然的。

S(x)表示奶牛x的开始时间。E(x)表示x的结束时间。对E(x), x可以是奶牛，也可以是畜栏。畜栏的结束时间，就是正在其里面挤奶的奶牛的结束时间。同一个畜栏的结束时间是不断在变的。

例题： Stall Reservations(POJ 3190)

1) 把所有奶牛按开始时间从小到大排序。
2) 为第一头奶牛分配一个畜栏。
3) 依次处理后面每头奶牛i。处理 i 时，考虑已分配畜栏中，结束时间最早的畜栏x。

若 E(x) < S(i), 则不用分配新畜栏，i可进入x,并修改E(x)为E(i)
若 E(x) >= S(i)，则分配新畜栏y,记 E(y) = E(i)

直到所有奶牛处理结束

需要用优先队列存放已经分配的畜栏，并使得结束时间最早的畜栏始终位于队列头部。

code c++

```
#include<iostream>
#include<algorithm>
#include<iomanip>
#include<cstdio>
#include<vector>
#include<queue>
using namespace std;
const double eps = 1e-6;
struct Cow
{
    int a, b;//挤奶牛区间
    int No;//编号
    bool operator<(const Cow & c)const{
        return a < c.a;
    }
};
int pos[50100];
struct Stall
{
    int end;
    int No;
    bool operator<(const Stall & s)const{
        return end>s.end;
```

```
    }
    Stall(int e, int n) :end(e), No(n){}

};
void main(){
    int n;
    cin >> n;
    vector<Cow>cows(50100);
    for (int i = 0; i < n; ++i){
        cin >> cows[i].a >> cows[i].b;
        cows[i].No = i;
    }
    sort(cows.begin(), cows.begin() + n);
    int total = 0;
    priority_queue<Stall>pq;
    for (int i = 0; i < n; ++i){
        if (pq.empty()){
            ++total;
            pq.push(Stall(cows[i].b, i));
            pos[cows[i].No] = total;
        }
        else{
            Stall st = pq.top();
            if (st.end < cows[i].a){
                //当前吃奶结束时间小于下头牛开始时间，不需要添加牛棚
                pq.pop();
                pos[cows[i].No] = st.No;
                pq.push(Stall(cows[i].b, st.No));

            }
            else{
                ++total;//需要加牛棚
                pq.push(Stall(cows[i].b, total));
                //将吃奶结束时间和牛棚号写进去
                pos[cows[i].No] = total;
                //牛在第几个牛棚排队吃奶
            }
        }
    }
    cout <<"out is"<<endl<< total << endl;
    for (int i = 0; i < n; ++i){
        cout << pos[i] << endl;
        //第i个牛在第几号牛棚
    }
    system("pause");
}
```

样例输入

5
1 10
2 4
3 6
5 8
4 7
out is
4

```
1
2
3
2
4
请按任意键继续. . .
```