贪心算法

贪心算法的介绍

贪心算法思想：

顾名思义，贪心算法总是作出在当前看来最好的选择。也就是说贪心算法并不从整体最优考虑，它所作出的选择只是在某种意义上的局部最优选择。当然，希望贪心算法得到的最终结果也是整体最优的。虽然贪心算法不能对所有问题都得到整体最优解，但对许多问题它能产生整体最优解。如单源最短路经问题，最小生成树问题等。在一些情况下，即使贪心算法不能得到整体最优解，其最终结果却是最优解的很好近似。

贪心算法的基本要素：

1.贪心选择性质。所谓贪心选择性质是指**所求问题的整体最优解可以通过一系列局部最优的选择**，即贪心选择来达到。这是贪心算法可行的第一个基本要素，也是贪心算法与动态规划算法的主要区别。

动态规划算法通常以自底向上的方式解各子问题，而贪心算法则通常以自顶向下的方式进行，以迭代的方式作出相继的贪心选择，每作一次贪心选择就将所求问题简化为规模更小的子问题。

对于一个具体问题，要确定它是否具有贪心选择性质，必须证明每一步所作的贪心选择最终导致问题的整体最优解。

2.当一个问题的最优解包含其子问题的最优解时，称此问题具有最优子结构性质。问题的最优子结构性质是该问题可用动态规划算法或贪心算法求解的关键特征。

贪心算法的基本思路：

从问题的某一个初始解出发逐步逼近给定的目标，以尽可能快的地求得更好的解。当达到算法中的某一步不能再继续前进时，算法停止。

该算法存在问题：

1. 不能保证求得的最后解是最佳的；

2. 不能用来求最大或最小解问题；
3. 只能求满足某些约束条件的可行解的范围。

例题

【1】p1007 独木桥

P1007 独木桥

提交答案 加入题单

题目背景

复制Markdown 展开

战争已经进入到紧要时间。你是运输小队长，正在率领运输部队向前线运送物资。运输任务像做题一样的无聊。你希望找些刺激，于是命令你的士兵们到前方的一座独木桥上欣赏风景，而你留在桥下欣赏士兵们。士兵们十分愤怒，因为这座独木桥十分狭窄，只能容纳 1 个人通过。假如有 2 个人相向而行在桥上相遇，那么他们 2 个人将无法绕过对方，只能有 1 个人回头下桥，让另一个人先通过。但是，可以有多个人同时呆在同一个位置。

题目描述

突然，你收到从指挥部发来的信息，敌军的轰炸机正朝着你所在的独木桥飞来！为了安全，你的部队必须撤下独木桥。独木桥的长度为 L，士兵们只能呆在坐标为整数的地方。所有士兵的速度都为 1，但一个士兵某一时刻来到了坐标为 0 或 $L+1$ 的位置，他就离开了独木桥。

每个士兵都有一个初始面对的方向，他们会以匀速朝着这个方向行走，中途不会自己改变方向。但是，如果两个士兵面对面相遇，他们无法彼此通过对方，于是就分别转身，继续行走。转身不需要任何的时间。

由于先前的愤怒，你已不能控制你的士兵。甚至，你连每个士兵初始面对的方向都不知道。因此，你想要知道你的部队最少需要多少时间就可能全部撤离独木桥。另外，总部也在安排阻拦敌人的进攻，因此你还需要知道你的部队最多需要多少时间才能全部撤离独木桥。

输入格式

第一行共一个整数 L，表示独木桥的长度。桥上的坐标为 $1, 2, \cdots, L$。

第二行共一个整数 N，表示初始时留在桥上的士兵数目。

第三行共有 N 个整数，分别表示每个士兵的初始坐标。

输出格式

共一行，输出 2 个整数，分别表示部队撤离独木桥的最小时间和最大时间。2 个整数由一个空格符分开。

输入输出样例

输入 #1 复制

```
4
2
1 3
```

输出 #1 复制

```
2 4
```

说明/提示

对于 100% 的数据，满足初始时，没有两个士兵同在一个坐标，$1 \le L \le 5 \times 10^3$，$0 \le N \le 5 \times 10^3$，且数据保证 $N \le L$。

```cpp
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int Max=0,Min=0;
    int l,n;
    cin>>l>>n;
    for(int i=0;i<n;i++)
    {
        int m;
        cin>>m;
        Max=max(Max,max(l-m+1,m));
        Min=max(Min,min(l-m+1,m));//精髓
    }
    cout<<Min<<' '<<Max;
}
```

【2】P1090 [NOIP2004 提高组] 合并果子

P1090 [NOIP2004 提高组] 合并果子 / [USACO06NOV] Fence R

提交答案 加入题单

题目描述

复制Markdown 展开

在一个果园里，多多已经将所有的果子打了下来，而且按果子的不同种类分成了不同的堆。多多决定把所有的果子合成一堆。

每一次合并，多多可以把两堆果子合并到一起，消耗的体力等于两堆果子的重量之和。可以看出，所有的果子经过 $n-1$ 次合并之后， 就只剩下一堆了。多多在合并果子时总共消耗的体力等于每次合并所耗体力之和。

因为还要花大力气把这些果子搬回家，所以多多在合并果子时要尽可能地节省体力。假定每个果子重量都为 1 ，并且已知果子的种类 数和每种果子的数目，你的任务是设计出合并的次序方案，使多多耗费的体力最少，并输出这个最小的体力耗费值。

例如有 3 种果子，数目依次为 1 ， 2 ， 9 。可以先将 1 、 2 堆合并，新堆数目为 3 ，耗费体力为 3 。接着，将新堆与原先的第三堆合并，又得到新的堆，数目为 12 ，耗费体力为 12 。所以多多总共耗费体力 $=3+12=15$ 。可以证明 15 为最小的体力耗费值。

输入格式

共两行。
第一行是一个整数 $n(1 \leq n \leq 10000)$ ，表示果子的种类数。

第二行包含 n 个整数，用空格分隔，第 i 个整数 $a_i(1 \leq a_i \leq 20000)$ 是第 i 种果子的数目。

输出格式

一个整数，也就是最小的体力耗费值。输入数据保证这个值小于 2^{31} 。

输入输出样例

输入 #1 复制

```
3
1 2 9
```

输出 #1 复制

```
15
```

说明/提示

对于 30% 的数据，保证有 $n \leq 1000$:

对于 50% 的数据，保证有 $n \leq 5000$;

对于全部的数据，保证有 $n \leq 10000$。

```cpp
#include<bits/stdc++.h>
using namespace std;
priority_queue<int,vector<int>,greater<int> >q;//优先队列声明
int n,weight,ans;
int main()
{
    cin>>n;
    for(int i=0;i<n;i++)
    {
        cin>>weight;
        q.push(weight);
    }
    while(q.size()>=2)
    {
        int a=q.top();
        q.pop();
        int b=q.top();
        q.pop();
        ans+=(a+b);
        q.push(a+b);
    }
    cout<<ans;
}
```

【3】P1199 [NOIP2010 普及组] 三国游戏

题目描述

复制Markdown 展开

小涵很喜欢电脑游戏，这些天他正在玩一个叫做《三国》的游戏。

在游戏中，小涵和计算机各执一方，组建各自的军队进行对战。游戏中共有 N 位武将（N为偶数且不小于4），任意两个武将之间有一个"默契值"，表示若此两位武将作为一对组合作战时，该组合的威力有多大。游戏开始前，所有武将都是自由的（称为自由武将，一旦某个自由武将被选中作为某方军队的一员，那么他就不再是自由武将了），换句话说，所谓的自由武将不属于任何一方。

游戏开始，小涵和计算机要从自由武将中挑选武将组成自己的军队，规则如下：小涵先从自由武将中选出一个加入自己的军队，然后计算机也从自由武将中选出一个加入计算机方的军队。接下来一直按照"小涵→计算机→小涵→……"的顺序选择武将，直到所有的武将被双方均分完。然后，程序自动从双方军队中各挑出一对默契值最高的武将组合代表自己的军队进行二对二比武，拥有更高默契值的一对武将组合获胜，表示两军交战，拥有获胜武将组合的一方获胜。

已知计算机一方选择武将的原则是尽量破坏对手下一步将形成的最强组合，它采取的具体策略如下：任何时刻，轮到计算机挑选时，它会尝试将对手军队中的每个武将与当前每个自由武将进行一一配对，找出所有配对中默契值最高的那对武将组合，并将该组合中的自由武将选入自己的军队。下面举例说明计算机的选将策略，例如，游戏中一共有6个武将，他们相互之间的默契值如下表所示：

武将编号	1	2	3	4	5	6
1		5	28	16	29	27
2	5		23	3	20	1
3	28	23		8	32	26
4	16	3	8		33	11
5	29	20	32	33		12
6	27	1	26	11	12	

双方选将过程如下所示：

	小涵	轮到计算机时可选的自由武将	计算机	计算机选将说明
第一轮	5	1 2 3 4 6	4	小涵手中5号武将与4号的默契值最高，所以选择4号
第二轮	5 3	1 2 6	4 1	小涵手中的5号和3号武将与自由武将中配对可产生的最大默契值为29，是由5号与1号配对产生的，因此计算机选择1号
第三轮	5 3 6	2	4 1 2	

小涵想知道，如果计算机在一局游戏中始终坚持上面这个策略，那么自己有没有可能必胜？如果有，在所有可能的胜利结局中，自己那对用于比武的武将组合的默契值最大是多少？

假设整个游戏过程中，对战双方任何时候均能看到自由武将队中的武将和对方军队的武将。为了简化问题，保证对于不同的武将组合，其默契值均不相同。

输入格式

共 N 行。

第一行为一个偶数 N，表示武将的个数。

第 2行到第 N行里，第$i+1$行有N_i个非负整数，每两个数之间用一个空格隔开，表示i号武将和$i+1, i+2, \ldots, N$号武将之间的默契值（$0 \leq$默契值$\leq 1,000,000,000$）。

输出格式

共 1 或 2行。

若对于给定的游戏输入，存在可以让小涵获胜的选将顺序，则输出1，并另起一行输出所有获胜的情况中，小涵最终选出的武将组合的最大默契值。如果不存在可以让小涵获胜的选将顺序，则输出 0。

输入输出样例

输入 #1 复制

```
6
5 28 16 29 27
23 3 20 1
8 32 26
33 11
12
```

输出 #1 复制

```
1
32
```

输入 #2 复制

```
8
42 24 10 29 27 12 58
31 8 16 26 80 6
25 3 36 11 5
33 20 17 13
15 77 9
4 50
19
```

输出 #2 复制

```
1
77
```

说明/提示

【数据范围】

对于40%的数据有 $N < 10$。

对于70%的数据有$N \leq 18$.

对于100%的数据有$N \leq 500$.

```
#include<bits/stdc++.h>
using namespace std;
int a[505][505];
int main()
{
    int n;
    cin>>n;
    for(int i=1;i<n;i++)
    {
        for(int j=i+1;j<=n;j++)
        {
            cin>>a[i][j];
            a[j][i]=a[i][j];
        }
    }
    int ans=0;
    for(int i=1;i<=n;i++)
    {
        sort(a[i]+1,a[i]+1+n);
        if(a[i][n-1]>ans) ans=a[i][n-1];
    }
    cout<<'1'<<endl;
    cout<<ans;
}
/*小涵和电脑都拿不到每个武将的最大默契值，因为小涵每拿起最大组合的其中一个，
另一个就被电脑拿走了，但是电脑也无法占有该组合的最大值，因为另一个在小涵手里，这个组合的最大值其实就作废了。
但是小涵在第二次选将时可以拿到第一次选中的武将的次大默契值组合。
在双方都拿不到最大默契值的情况下，显然谁拿到次大默契组合的最大值，谁就赢了。*/
```

【4】P1208 [USACO1.3]混合牛奶 Mixing Milk

P1208 [USACO1.3]混合牛奶 Mixing Milk

提交答案 加入题单

题目描述

复制Markdown 展开

由于乳制品产业利润很低，所以降低原材料（牛奶）价格就变得十分重要。帮助 Marry 乳业找到最优的牛奶采购方案。

Marry 乳业从一些奶农手中采购牛奶，并且每一位奶农为乳制品加工企业提供的价格是不同的。此外，就像每头奶牛每天只能挤出固定数量的奶，每位奶农每天能提供的牛奶数量是一定的。每天 Marry 乳业可以从奶农手中采购到小于或者等于奶农最大产量的整数数量的牛奶。

给出 Marry 乳业每天对牛奶的需求量，还有每位奶农提供的牛奶单价和产量。计算采购足够数量的牛奶所需的最小花费。

注：每天所有奶农的总产量大于 Marry 乳业的需求量。

输入格式

第一行二个整数 n, m，表示需要牛奶的总量，和提供牛奶的农民个数。

接下来 m 行，每行两个整数 p_i, a_i，表示第 i 个农民牛奶的单价，和农民 i 一天最多能卖出的牛奶量。

输出格式

单独的一行包含单独的一个整数，表示 Marry 的牛奶制造公司拿到所需的牛奶所要的最小费用。

输入输出样例

输入 #1 复制

```
100 5
5 20
9 40
3 10
8 80
6 30
```

输出 #1 复制

```
630
```

说明/提示

【数据范围】
对于 100% 的数据：
$0 \le n, a_i \le 2 \times 10^6$，$0 \le m \le 5000$，$0 \le p_i \le 1000$

题目翻译来自 NOCOW。

USACO Training Section 1.3

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N=5005;
int n,m,ans;
struct milk
{
    int volumn;
    int price;
}arr[N];
bool cmp(milk a,milk b)
{
    return a.price<b.price;
}
int main()
{
    cin>>n>>m;
    for(int i=0;i<m;i++)
        cin>>arr[i].price>>arr[i].volumn;
    sort(arr,arr+m,cmp);
    for(int i=0;i<m;i++)
    {
        if(n>=arr[i].volumn)
        {
            ans+=arr[i].volumn*arr[i].price;
            n-=arr[i].volumn;
        }
        else
        {
            ans+=n*arr[i].price;
            break;//注意要break
        }
    }
    cout<<ans;
}
```

【5】P1223 排队接水

P1223 排队接水

提交答案 加入题单

题目描述

复制Markdown 展开

有 n 个人在一个水龙头前排队接水，假如每个人接水的时间为 T_i，请编程找出这 n 个人排队的一种顺序，使得 n 个人的平均等待时间最小。

输入格式

第一行为一个整数 n。

第二行 n 个整数，第 i 个整数 T_i 表示第 i 个人的等待时间 T_i。

输出格式

输出文件有两行，第一行为一种平均时间最短的排队顺序；第二行为这种排列方案下的平均等待时间（输出结果精确到小数点后两位）。

输入输出样例

输入 #1 复制

```
10
56 12 1 99 1000 234 33 55 99 812
```

输出 #1 复制

```
3 2 7 8 1 4 9 6 10 5
291.90
```

说明/提示

$n \leq 1000, t_i \leq 10^6$，不保证 t_i 不重复。

当 t_i 重复时，按照输入顺序即可（sort 是可以的）

```cpp
#include<bits/stdc++.h>
using namespace std;
int arr[1005],num[1005];
int n;
double sum=0;
int main()
{
    cin>>n;
    for(int i=0;i<n;i++)
    {
        cin>>arr[i];
        num[i]=i+1;
    }
    for(int i=0;i<n;i++)
    {
        for(int j=i;j<n;j++)//按从小到大对每个人序号进行排序
        {
```

```
            if(arr[i]>arr[j])
            {
                int k=arr[i];
                arr[i]=arr[j];
                arr[j]=k;
                int x=num[i];
                num[i]=num[j];
                num[j]=x;
            }
        }
    }
    for(int i=0;i<n;i++)
        cout<<num[i]<<' ';
    cout<<endl;
    for(int i=0;i<n-1;i++)
    {
        sum+=arr[i]*(n-i-1);//求总等待时间（当前的人用时*后面剩下的人数量）
    }
    cout<<setprecision(2)<<setiosflags(ios::fixed)<<sum/n;
}
```

【6】P1803 凌乱的yyy

P1803 凌乱的yyy / 线段覆盖

提交答案 加入题单

题目背景

复制Markdown 展开

快 noip 了，yyy 很紧张!

题目描述

现在各大 oj 上有 n 个比赛，每个比赛的开始、结束的时间点是知道的。

yyy 认为，参加越多的比赛，noip 就能考的越好（假的）。

所以，他想知道他最多能参加几个比赛。

由于 yyy 是蒟蒻，如果要参加一个比赛必须善始善终，而且不能同时参加 2 个及以上的比赛。

输入格式

第一行是一个整数 n，接下来 n 行每行是 2 个整数 a_i, b_i ($a_i < b_i$)，表示比赛开始、结束的时间。

输出格式

一个整数最多参加的比赛数目。

输入输出样例

输入 #1 复制

```
3
0 2
2 4
1 3
```

输出 #1 复制

```
2
```

说明/提示

对于 20% 的数据，$n \leq 10$。

对于 50% 的数据，$n \leq 10^3$。

对于 70% 的数据，$n \leq 10^5$。

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+5;
int n,ans=1;//注意ans初始值要赋为1
struct t
{
```

```
    int st;
    int end;
}arr[N];
bool cmp(t a,t b)//判断结束时间先后
{
    return a.end<b.end;
}
int main()
{
    cin>>n;
    for(int i=0;i<n;i++)
    {
        cin>>arr[i].st>>arr[i].end;
    }
    sort(arr,arr+n,cmp);//按结束时间先后排序
    int cur=arr[0].end;
    for(int i=1;i<n;i++)//按结束时间依次遍历
    {
        if(cur<=arr[i].st)//优先选取结束时间靠前的考试参加
        {
            cur=arr[i].end;
            ans++;
        }
    }
    cout<<ans;
}
```

【7】7-3 h0145. 会议安排

7-3 h0145. 会议安排 分数 20

全屏浏览题目 切换布局 作者 黄正鹏 单位 贵州工程应用技术学

学校的礼堂每天都会有许多活动，有时间这些活动的计划时间会发生冲突，需要选择出一些活动进行举办。小刘的工作就是安排学校礼堂的活动，每个时间最多安排一个活动。现在小刘有一些活动计划的时间表，他想尽可能的安排更多的活动，请问他该如何安排。

输入格式:

第一行是一个整型数m(m<100)表示共有m组测试数据。
每组测试数据的第一行是一个整数n(1<n<10000)表示该测试数据共有n个活动。
随后的n行，每行有两个正整数Bi,Ei(0<=Bi,Ei<10000),分别表示第i个活动的起始与结束时间（Bi<=Ei)

输出格式:

对于每一组输入，输出最多能够安排的活动数量。
每组的输出占一行

输入样例:

在这里给出一组输入。例如:

```
2
2
1 10
10 11
3
1 10
9 11
11 20
```

输出样例:

在这里给出相应的输出。例如:

```
2
2
```

代码长度限制	16 KB
时间限制	400 ms
内存限制	64 MB

```cpp
#include<bits/stdc++.h>
using namespace std;
int m,n,ans=1;
struct T
{
    int st;
    int end;
};
T t[10005];
bool cmp(T t1,T t2)
{
   return t1.end<t2.end;
}
int main()
{
    cin>>m;
    for(int i=0;i<m;i++)
    {
```

```
        cin>>n;
        for(int j=0;j<n;j++)
            cin>>t[j].st>>t[j].end;
        sort(t,t+n,cmp);
        int cur=t[0].end;
        for(int i=1;i<n;i++)
        {
            if(cur<=t[i].st)
            {
                ans++;
                cur=t[i].end;
            }
        }
        if(i==m-1)cout<<ans;
        else cout<<ans<<endl;
        ans=1;
    }
}
```