# 贪心： 常见母题

### 一、区间选点问题：

题意：给定N个闭区间[ai, bi],请你在数轴上选择尽量少的点，使得每个区间都至少包含一个选出的点,输出选出的最少点的数量，边界点也算内部。
N <= 1e5
-1e9 <= ai <= bi <= 1e9

### 代码模板：

```
const int N = 1e5 + 10;
int a, b, n;
struct Range {
    int l, r;
    bool operator < (const Range &W) const {
        return r < W.r;
    }
}ran[N];
bool st[N];
int main() {
    cin >> n;
    for(int i = 0; i < n; i ++ ) {
        cin >> a >> b;
        ran[i] = {a, b};
    }
    sort(ran, ran + n);
    int res = 0;
    for(int i = 0; i < n; i ++ ) {
        if(!st[i]) {
            st[i] = true;
            int r_ = ran[i].r;
            for(int j = i + 1; j < n; j ++ ) {
                int t1 = ran[j].l, t2 = ran[j].r;
                if(r_ >= t1 && r_ <= t2) {
                    st[j] = true;
                }
                else break;
            }
            res ++;
        }
    }
    cout << res << endl;
    return 0;
}
```

## 二、最大不相交区间数量

给定 N个闭区间 [ai,bi]，请你在数轴上选择若干区间，使得选中的区间之间互不相交（包括端点）。
输出可选取区间的最大数量。

**代码模板：**

```
const int N = 1e5 + 10;
int n, a, b;
struct Ran {
    int l, r;
    bool operator < (const Ran &W) const
    {
        return r < W.r;
    }
}ran[N];
bool st[N];
int main() {
    cin >> n;
    for(int i = 0; i < n; i ++ ) {
        cin >> a >> b;
        ran[i] = {a, b};
    }
    sort(ran, ran + n );
    int res = 0;
    for(int i = 0; i < n; i ++ ) {
        if(!st[i]) {
            st[i] = true;
            int t = ran[i].r;
            for(int j = i + 1; j < n; j ++ ) {
                int c = ran[j].l;
                if(c <= t) {
                    st[j] = true;
                }
                else break;
            }
            res ++;
        }
    }
    cout << res << endl;
    return 0;
}
```

---

## 三、区间分组：

给定 N 个闭区间 [ai,bi]，请你将这些区间分成若干组，使得每组内部的区间两两之间（包括端点）没有交集，并使得组数尽可能小，输出最小组数。

**代码模板:**

```
const int N = 1e5 + 10;
int n;
struct range {
    int l, r;
    bool operator <(const range &W) const {
        return l < W.l;
    }
}ran[N];
int main() {
    cin >> n;
    for(int i = 0; i < n; i ++ ) {
        int l, r;
        cin >> l >> r;
        ran[i] = {l, r};
    }
    sort(ran, ran + n);
    int res = 0;
    priority_queue<int, vector<int>, greater<int> >  g;
    for(int i = 0; i < n; i ++ ) {
        int st = ran[i].l, ed = ran[i].r;
        if(g.empty() || g.top() >= st) g.push(ed);
        else {
            g.pop();
            g.push(ed);
        }
    }
    cout << g.size() << endl;
    return 0;
}
```

---

**四：区间覆盖问题：**

给定N个闭区间 [ai,bi] 以及一个线段区间 [s,t]，请你选择尽量少的区间，将指定线段区间完全覆盖。输出最少区间数，如果无法完全覆盖则输出 −1。

**代码模板:**

```
const int N = 1e5 + 10;
int n, st, t;
struct Range {
    int l, r;
    bool operator <(const Range& W) const {
        return l < W.l;
    }
}ran[N];

int main()
{
    cin >> st >> t >> n;
    for(int i = 0; i < n; i ++ )
    {
```

```
        int l,r ;
        cin >> l >> r;
        ran[i] = {l, r};
    }
    sort(ran, ran + n);
    int res = 0;
    bool flag = true;
    for(int i = 0; i < n; i ++ ) {
        int j = i, r = -2e9;
        while(j < n && ran[j].l <= st) {
            r = max(r, ran[j].r);
            j ++;
        }
        if(r < st) {
            res = -1;
            break;
        }
        res ++;
        if(r >= t) {
            flag = false;
            break;
        }
        st = r;
        i = j - 1;
    }
    if(flag) res = -1;
    cout << res << endl;
    return 0;
}
```

## 五、(贪心之Huffman树)

## 合并果子

在一个果园里，达达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。

达达决定把所有的果子合成一堆。

每一次合并，达达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。

达达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，所以达达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，使达达耗费的体力最少，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 $12$。

所以达达总共耗费体力= $3+12=15$。

可以证明 $15$ 为最小的体力耗费值。

**输入格式**

输入包括两行，第一行是一个整数 $n$，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 $a_i$ 是第 $i$ 种果子的数目。

时/空限制: 1s / 64MB

总通过数: 28171

总尝试数: 40199

来源:
《算法竞赛进阶指南》, NOIP2004提高组, 模板题

算法标签

**代码模板：**

```
const int N = 1e5 + 10;
int n;
struct Range {
    int l, r;
    bool operator <(const Range &W) const {
        return l < W.l;
    }
}ran[N];
priority_queue<int, vector<int>, greater<int> > heap;
int main() {
    cin >> n;
    for(int i = 0; i < n; i ++ ) {
        int x; cin >> x;
        heap.push(x);
    }
    int res = 0;
    while(heap.size() > 1) {
        int a = heap.top(); heap.pop();
        int b = heap.top(); heap.pop();
        res += (a + b);
        heap.push(a + b);
    }
    cout << res << endl;
    return 0;
}
```

---

**六、排序不等式：**

**母题：排队打水**

有 n个人排队到 1个水龙头处打水，第 i 个人装满水桶所需的时间是 ti，请问如何安排他们的打水顺序才能使所有人的等待时间之和最小？

**代码模板：**

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5 + 10;
int q[N], s[N], n;
int main() {
    cin >> n;
    for(int i = 1; i <= n; i ++ ) cin >> q[i];
    sort(q + 1, q + 1 + n);
    for(int i = 1; i <= n; i ++ ) s[i] = q[i] + s[i - 1];
    ll res = 0, sum = 0;
    for(int i = n; i >= 1; i -- ) {
        sum += q[i];
        res += (s[n] - sum);
    }
    cout << res << endl;
    return 0;
```

```
}
```

---

### 七、绝对值不等式

- 一维货舱选址问题：

在一条数轴上有N家商店，它们的坐标分别为 A1 ~ AN。现在需要在数轴上建立一家货仓，每天清晨，从货仓到每家商店都要运送一车商品。为了提高效率，求把货仓建在何处，可以使得货仓到每家商店的距离之和最小。

```
// 代码模板：
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10;
typedef long long ll;
ll q[N], n, ans;
int main() {
    cin >> n;
    for(int i = 1; i <= n; i ++ ) cin >> q[i];
    sort(q + 1, q + 1 + n);
    for(int i = 1; i <= n; i ++ ) ans += abs(q[n / 2 + 1] - q[i]);
    cout << ans << endl;
    return 0;
}
```

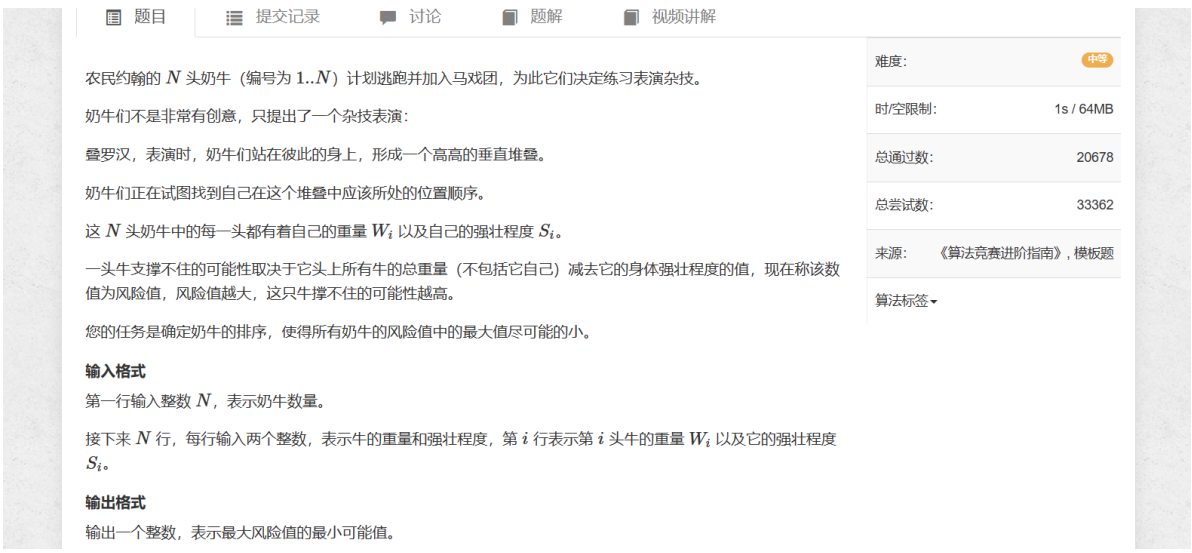
- 二维货舱选址问题：

题意：给定n个二维坐标，求一个点满足到所有坐标距离之和最小的点的有几个，距离是|x - x0| + |y - y0|

拓展：如果是三维或者是n维，需要用模拟退火 + 三分

题解：首先，x和y是割裂的。分别求x数量cnt_x，y的数量cnt_y，最后答案就是cnt_x * cnt_y;

```
// 代码：
void AC() {
    cin >> n;
    vector<int> xa(n + 1, 0), ya(n + 1, 0);
    for(int i = 1; i <= n; i ++ ) {
        int x, y;
        cin >> x >> y;
        xa[i] = x, ya[i] = y;
    }
    if(n % 2) cout << 1 << endl;
    else {
        sort(xa.begin(), xa.end());
        sort(ya.begin(), ya.end());
        int ans = (xa[n / 2 + 1] - xa[n / 2] + 1) * (ya[n / 2 + 1] - ya[n / 2] +
1);
        cout << ans << endl;
    }
}
```



题目 提交记录 讨论 题解 视频讲解

农民约翰的 $N$ 头奶牛（编号为 $1..N$）计划逃跑并加入马戏团，为此它们决定练习表演杂技。

奶牛们不是非常有创意，只提出了一个杂技表演：

叠罗汉，表演时，奶牛们站在彼此的身上，形成一个高高的垂直堆叠。

奶牛们正在试图找到自己在这个堆叠中应该所处的位置顺序。

这 $N$ 头奶牛中的每一头都有着自己的重量 $W_i$ 以及自己的强壮程度 $S_i$。

一头牛支撑不住的可能性取决于它头上所有牛的总重量（不包括它自己）减去它的身体强壮程度的值，现在称该数值为风险值，风险值越大，这只牛撑不住的可能性越高。

您的任务是确定奶牛的排序，使得所有奶牛的风险值中的最大值尽可能的小。

**输入格式**

第一行输入整数 $N$，表示奶牛数量。

接下来 $N$ 行，每行输入两个整数，表示牛的重量和强壮程度，第 $i$ 行表示第 $i$ 头牛的重量 $W_i$ 以及它的强壮程度 $S_i$。

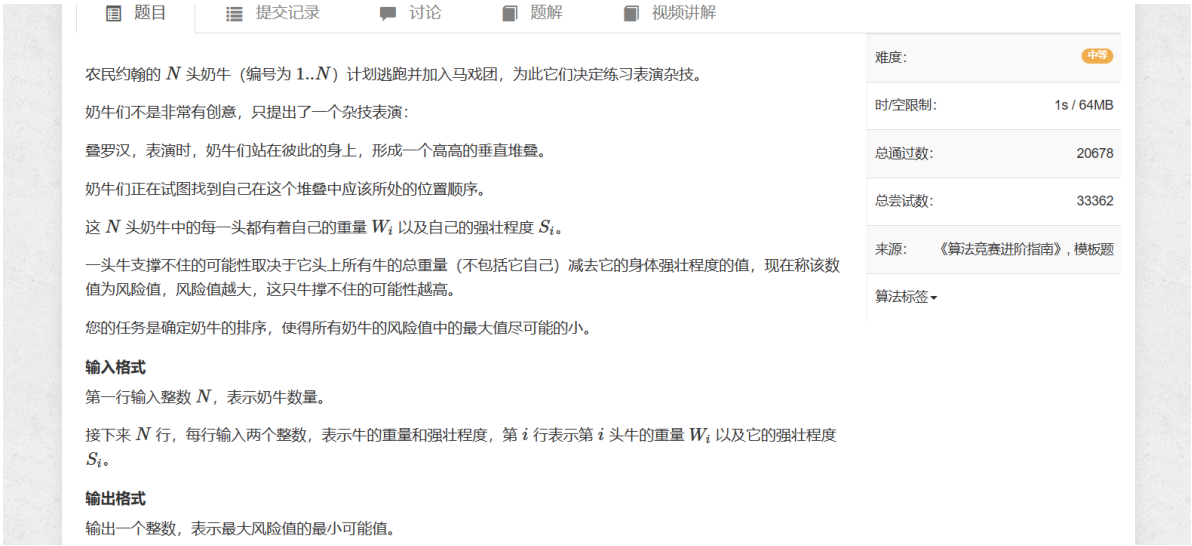
**输出格式**

输出一个整数，表示最大风险值的最小可能值。

难度: 中等
时/空限制: 1s / 64MB
总通过数: 20678
总尝试数: 33362
来源: 《算法竞赛进阶指南》, 模板题
算法标签

推公式的贪心：

母题耍杂技的牛：用w + s来贪心：



题目 提交记录 讨论 题解 视频讲解

农民约翰的 $N$ 头奶牛（编号为 $1..N$）计划逃跑并加入马戏团，为此它们决定练习表演杂技。

奶牛们不是非常有创意，只提出了一个杂技表演：

叠罗汉，表演时，奶牛们站在彼此的身上，形成一个高高的垂直堆叠。

奶牛们正在试图找到自己在这个堆叠中应该所处的位置顺序。

这 $N$ 头奶牛中的每一头都有着自己的重量 $W_i$ 以及自己的强壮程度 $S_i$。

一头牛支撑不住的可能性取决于它头上所有牛的总重量（不包括它自己）减去它的身体强壮程度的值，现在称该数值为风险值，风险值越大，这只牛撑不住的可能性越高。

您的任务是确定奶牛的排序，使得所有奶牛的风险值中的最大值尽可能的小。

**输入格式**

第一行输入整数 $N$，表示奶牛数量。

接下来 $N$ 行，每行输入两个整数，表示牛的重量和强壮程度，第 $i$ 行表示第 $i$ 头牛的重量 $W_i$ 以及它的强壮程度 $S_i$。

**输出格式**

输出一个整数，表示最大风险值的最小可能值。

难度: 中等
时/空限制: 1s / 64MB
总通过数: 20678
总尝试数: 33362
来源: 《算法竞赛进阶指南》, 模板题
算法标签

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 5e4 + 10;
typedef long long ll;
ll ans, n, sum;
struct cow {
    int w, s;
    bool operator <(const cow &W) const {
        return (w + s) < (W.w + W.s);
    }
}cow[N];
int main() {
    cin >> n;
    for(int i = 1; i <= n; i ++ ) {
        int x, y; cin >> x >> y;
        cow[i] = {x, y};
    }
```

```
    sort(cow + 1, cow + 1 + n);
    ans = 0 - cow[1].s;
    sum = cow[1].w;
    for(int i = 2; i <= n; i ++ ) {
        ans = max(ans, sum - cow[i].s);
        sum += cow[i].w;
    }
    cout << ans << endl;
    return 0;
}
```