复习：
基础排序、贪心

基础-提高衔接计划 B

Maxmilite

2025-07-14

课前提示

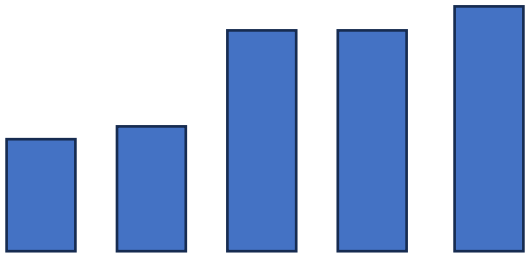
- 上课的时候专心听讲解，**不要跟着老师抄代码**，下课后独立完成。
- 不使用 AI 做题，AI 会做不等于自己会。
- 不抄袭题解（含对照题解抄一遍），抄对不等于会做。
- 看完题解后，关闭题解独立练习。
- 练习中途遇到问题，应当分析题目及自己的思路，而非回忆题解或再次参考题解。
- 做过的题在课后需要重新独立完成，不参考老师的课件、代码，不参考自己以前的代码。

目录

为什么要排序？

- 有序性是一个很好的性质。
- 可以协助快速处理很多事情
- 试卷分数从高到低排序
- 同学按照身高从矮到高排序
- 价格从低到高排序

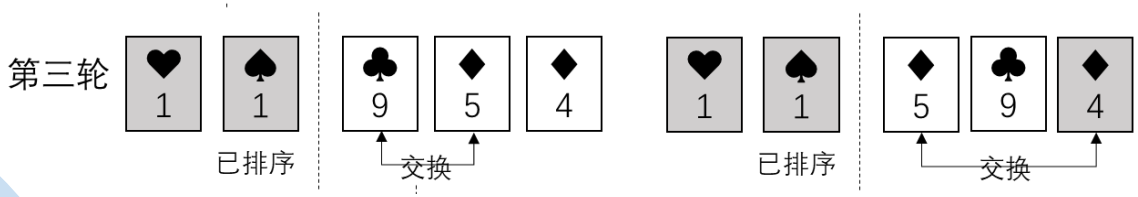
- 排序是将数据按照一定顺序进行排列的过程。
- 从小到大
- 字典序
- ……



选择排序

选择排序

- 思路:
- 进行 n 次大循环。第 i 次大循环中:
- 用小循环寻找数列中第 i 小的元素。
- 具体方式：如果后面发现更小的数字，则交换和当前第 i 个位置的数交换。
- 由于前 i-1 个数已经排序完毕，求第 i 小的元素相当于求第 i 至第 n 项中的最小值。



选择排序

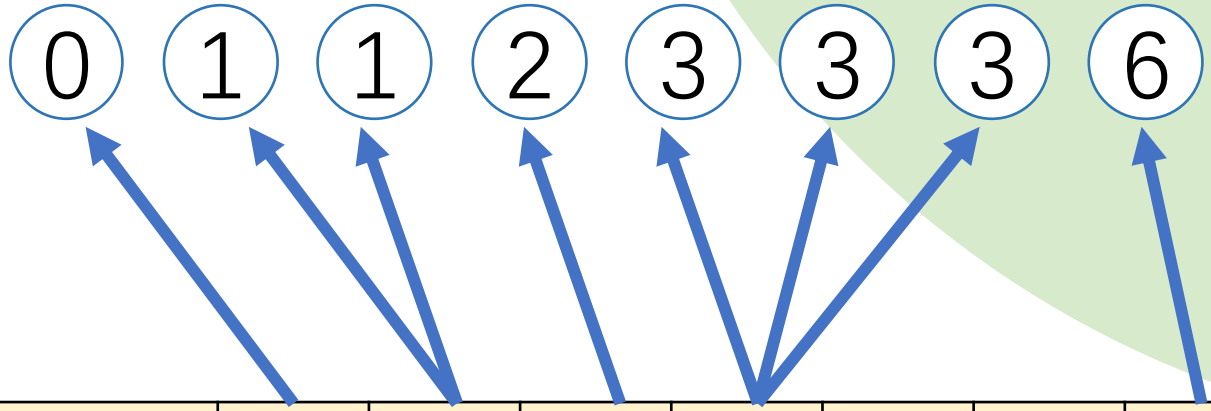
- 代码实现
- n 次大循环，第 i 次大循环中，用小循环寻找数列中第 i 小的元素。如果后面发现更小的数字，则交换至第 i 个位置。

```
int n, a[105], tmp;
cin >> n;
for(int i = 1; i <= n; i++)
    cin >> a[i];
for(int i = 1; i <= n - 1; i++) //大循环：i 从 1 到 n-1
    for(int j = i + 1; j <= n; j++) //小循环：j 从 i 的下一个到 n
        if(a[i] > a[j]) { //如果 a[i] 比 a[j] 大，交换他们
            tmp = a[i];
            a[i] = a[j];
            a[j] = tmp;
        }
for(int i = 1; i <= n; i++)
    cout << a[i] << " ";
```

计数排序

计数排序

- 记录每个数字有多少个
- 根据数字的数量，依次输出每个数字。



数量	1	2	1	3	0	0	1
数字	0	1	2	3	4	5	6

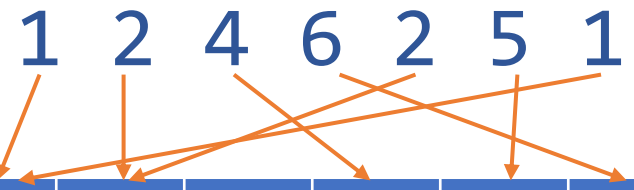
计数排序

- https://www.luogu.com.cn/problem/T634797
- 第一行输入 $n(n \le 1000000)$。第二行输入 n 个 1 到 10000 的整数。将这些数字从小到大排序。

```
7
1 2 4 6 2 5 1
```

```
1 1 2 2 4 5 6
```

- 设立数组 a[10010]，分别放置数字 1 到 10000 的数量。
- 比如 a[3] 就是 "3" 这个数字的数量。
- 然后读入数字统计。最后，i 从 1 到 10000，输出 a[i] 个 i。



x	0	1	2	3	4	5	6	7	8	9
a[x]	0	2	2	0	1	1	1	0	0	0

计数排序

- 代码实现

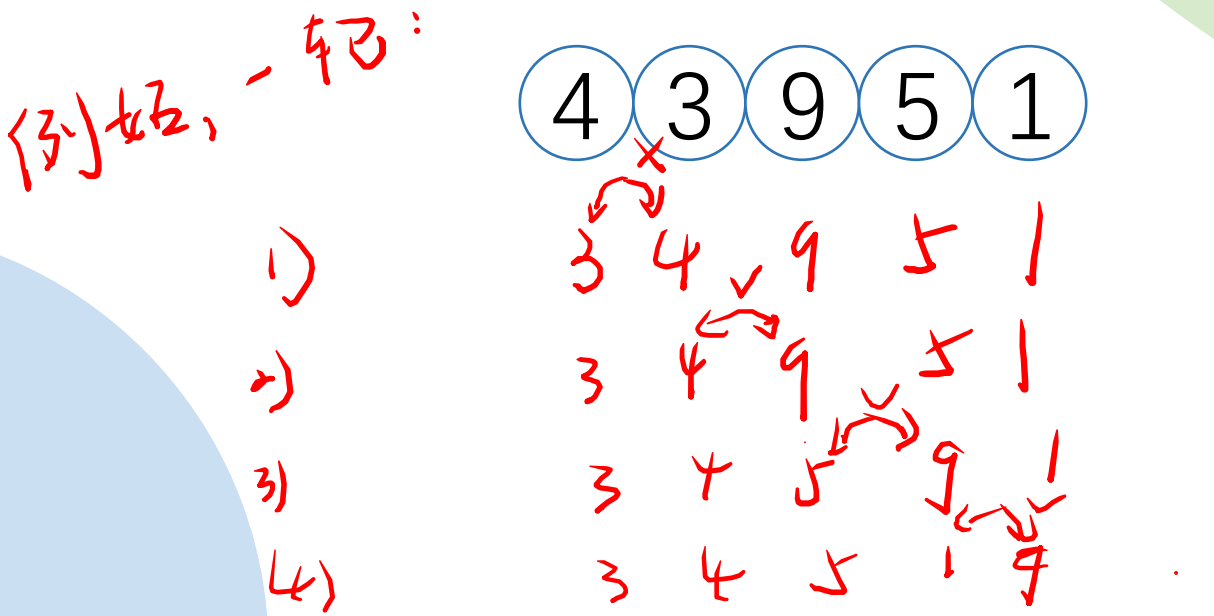
```
cin >> n;
for (int i = 1; i <= n; i++) {
    int x;
    cin >> x;
    a[x]++;
}
for (int i = 1; i <= 10000; i++) {
    for (int j = 1; j <= a[i]; j++)
        cout << i << " ";
}
```

- 思考：
- 如果需要从大到小排序，该如何做？
- 如果数字的范围为-10000至10000，该如何做？

冒泡排序

冒泡排序

- 输入 $n(n \leq 100)$ 和 n 个 0 到 100000000 的整数，然后将这些数字从小到大排序。
- 重复地比较相邻的两个元素，如果顺序错误就交换它们。



冒泡排序

- 核心代码

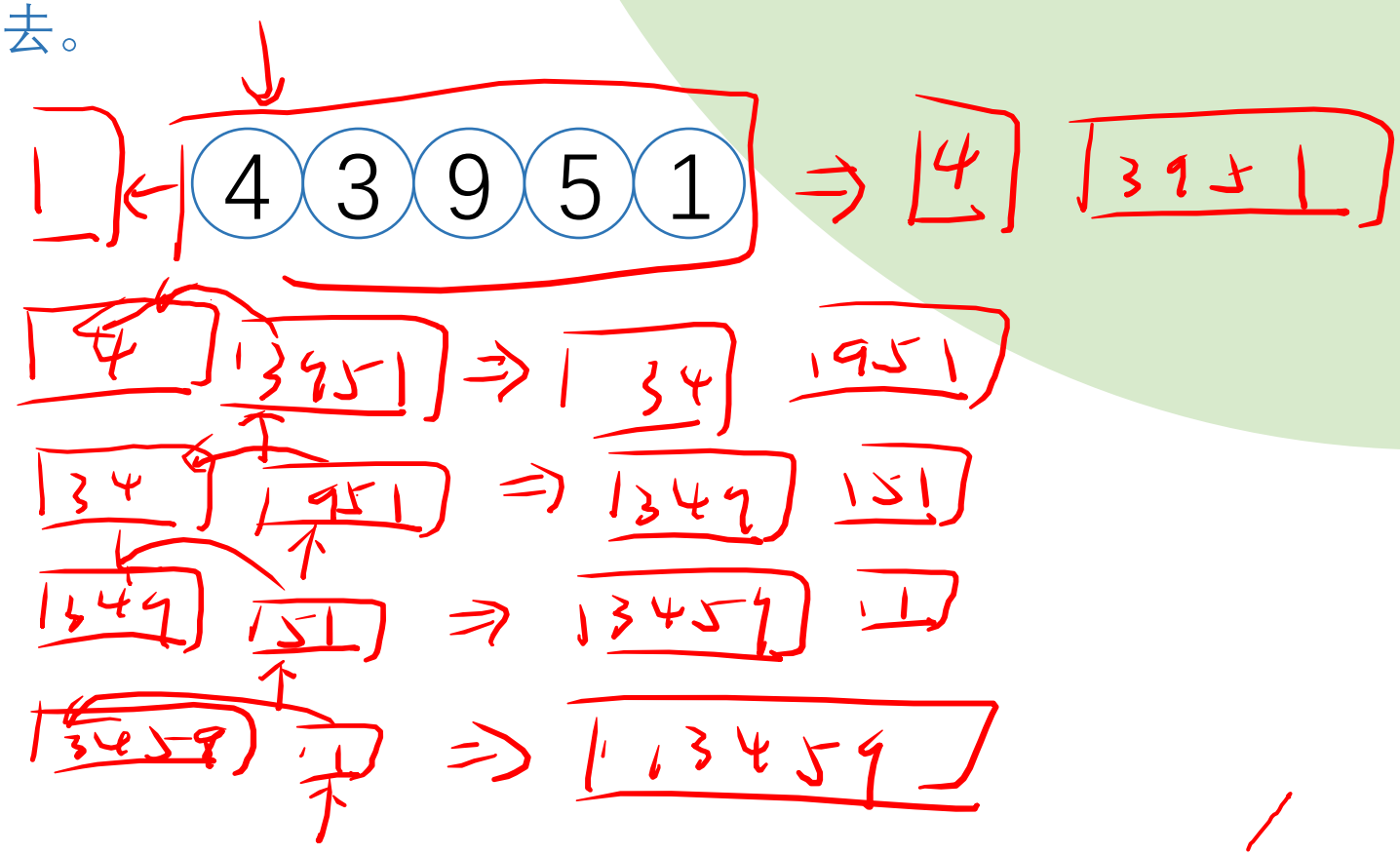
```
for (int i = 1; i < n ; i++)
    for (int j = 1; j < n - i; j++)
        if (a[j] > a[j + 1])
            swap(a[j], a[j + 1]);
```

- 特点：
- 每次循环都能保证至少一个（最大）元素的位置被确定。
- 后 i 个元素有序，且为最大的 i 个元素。
- 时间复杂度
- 两重循环，每个循环次数最多到 n，所以是 $O(n^2)$
- 空间复杂度
- $O(n)$

插入排序

插入排序

- 这边有 5 个数字，需要依次插入已有的排序好的数组。
- 拿到一个数字，从右往左边看，找到第一个比他小的位置，然后插入进去。



插入排序

- 代码实现
- n 次大循环。
- 第 i 次大循环中，记录要插入的数字 a[i]。
- 从第 i 个元素开始，如果前面一个数字比待插数字大，则往后面挪动一格。
- 直到前面一张牌不大于待插牌，或者到最左边。
- 将待插牌插入到空格里面去。

```
scanf("%d", &n);
for (int i = 1; i <= n; i++)
    scanf("%d", &a[i]);
for (int i = 1; i <= n; i++) {
    int now = a[i], j;
    // 记录一下待插牌，等下还要放回去
    for (j = i - 1; j >= 1; j--)
        if (a[j] > now)
            a[j + 1] = a[j];
        else break;
    a[j + 1] = now;
}
for (int i = 1; i <= n; i++) {
    printf("%d ", a[i]);
}
```

快速排序（sort 的使用）

STL中的排序算法

- 实际上，只要一句话就可以帮你快速排序!
- algorithm（算法）库包含很多常用的算法，包括排序。
- 包含头文件：

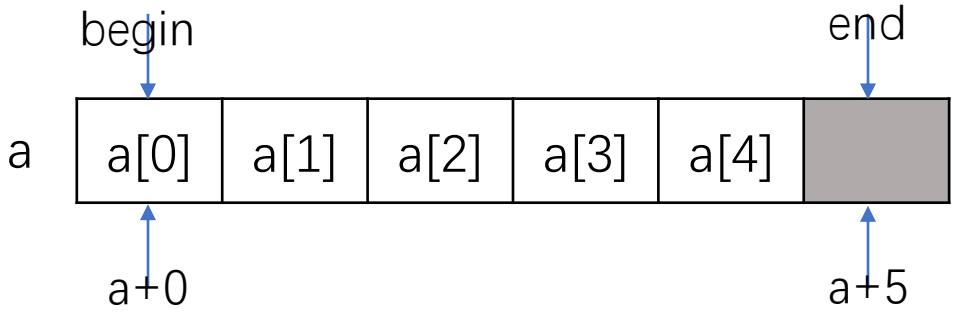
```
#include <algorithm>
```

- 对于数组下标：
- 对一个长度为 n 的数组 a 排序（a[0] 到 a[n-1] 进行排序）

```
sort(a, a+n);
// 对 a[0] 到 a[n-1] 进行排序
```

- 对一个长度为 n 的数组 a 排序（a[1] 到 a[n] 进行排序）

```
sort(a+1, a+n+1);
// 对 a[1] 到 a[n] 进行排序
```



sort 排序

- C++ 中 sort 实现采用快速排序的算法。
- 算法难度稍微有点大，本次课先不介绍原理。

- 时间复杂度：$O(n \log n)$
- 空间复杂度：$O(n \log n)$

- 和数据类型（值域无关），整数、浮点数等都可以使用。
- 可以轻松处理 10^6 规模的测试数据。

自定义排序

如果我想排序，但不是从小到大排序，怎么办？
或者，能对结构体元素进行排序吗？

- sort 函数可以增加第三个参数 cmp，也就是自定义排序的基准。
- cmp 函数需要两个待排序的元素 x、y 作为参数。
- 返回一个 bool 值，表示 x 是否严格小于 y。
- 例如：实现整数从大到小排序。

```
sort(a + 1, a + n + 1, cmp);
```

```
bool cmp(const int& x, const int& y){
    return x > y;
}
```

bool 是一种只有 true(1) 和 false(0) 的类型。

- cmp 函数名称可以任取，保持上下一致即可。

结构体排序

- 举例：将结构体 a 数组先按 x 从小到大排序。x 相同的，再按 y 从大到小排序。

```
struct node {
    int x, y;
} a[maxn];

bool cmp(const node &a, const node &b) {
    if (a.x != b.x) return a.x < b.x;
    return a.y > b.y;
}

sort(a + 1, a + n + 1, cmp);
```

排序

- 总结
- 稳定性是指相等的元素经过排序之后相对顺序是否发生了改变。

算法名称	平均时间复杂度	平均空间复杂度	是否稳定
冒泡排序	$O(n^2)$	$O(n)$	是
选择排序	$O(n^2)$	$O(n)$	否
插入排序	$O(n^2)$	$O(n)$	是
计数排序	$O(n+w)$，w 为值域	$O(n+w)$	是
快速排序	$O(n\log n)$	$O(n)$	否

例题

- 题目来源 CSP-J 2022 初赛 | 第 12 题
- 以下排序算法的常见实现中，哪个选项说法是错误的（ B ）。

A．冒泡排序算法是稳定的

B．简单选择排序是稳定的

C．简单插入排序是稳定的

~~D．归并排序算法是稳定的~~

贪心题选讲

P1376 机器工厂

- https://www.luogu.com.cn/problem/P1376
- 小 T 开办了一家机器工厂，第 i 周生产一台机器需要花费 Ci 元。若没把机器卖出去，每保养一台机器，每周需要花费 S 元（S 固定）。
- 机器工厂接到订单，在第 i 周需要交付 Yi 台机器给委托人，第 i 周刚生产的机器，或者之前的存货，都可以进行交付。
- 请你计算出这 n 周时间内完成订单的最小代价。

P1376 机器工厂

解法

对于第 i 周:

- 可以当场生产机器;
- 也可以选择以前的某一周生产的机器。

从以上的策略中找到代价（成本）最低的方案。

对于每一周，从第 1 周到第 i 周枚举，找到成本最低方案。

优化

对于过去的每一周的不同方案，它们成本差值是不变的。

成本最低的方案会一直最优，除非被新的成本更低的方案替代。

可以记录到目前为止的最优方案，和本周的方案比较。

$$cost = \min(cost + S, C_i)$$

P1376 机器工厂

```cpp
int n, s;
struct node {
    int c, y;
} a[10005];
int main() {
    cin >> n >> s;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i].c >> a[i].y;
    }
    long long ans = 0; int min_cost = 1000000000;
    for (int i = 1; i <= n; ++i) {
        min_cost = min(min_cost + s, a[i].c);
        ans += min_cost * a[i].y;
    }
    cout << ans << endl;
    return 0;
}
```

P3817 小A的糖果

- https://www.luogu.com.cn/problem/P3817
- 小 A 有 n 个糖果盒，第 i 个盒中有 a_i 颗糖果。
- 小 A 每次可以从其中一盒糖果中吃掉一颗，他想知道，要让任意两个相邻的盒子中糖的个数之和都不大于 x，至少得吃掉几颗糖。

P3817 小A的糖果

解法

首先，需要保证前面两个盒子的糖果总和不超过 x。

优先吃掉前面的糖果，还是后面的呢？

应该优先吃掉后面的。

因为，如果吃前面的，那么对后面的对糖果数减小，没有任何帮助。吃后面的可能会让再后面的一对少吃一些。

实现

从前到后枚举每一相邻对。

如果这一对糖果数超过了 x，则吃掉后面的盒子的糖果，直到 x。

P3817 小A的糖果

```cpp
int n, x, a[100005];
int main() {
    cin >> n >> x;
    for (int i = 1; i <= n; ++i)
        cin >> a[i];
    long long ans = 0;
    for (int i = 1; i <= n; ++i) {
        if (a[i] > x) {
            ans += a[i] - x; a[i] = x;
        }
        if (a[i] + a[i + 1] > x) {
            ans += a[i] + a[i + 1] - x;
            a[i + 1] = x - a[i];
        }
    }
    cout << ans << endl;
    return 0;
}
```

B4051 小杨的武器

https://www.luogu.com.cn/problem/B4051

- 小杨有 n 种武器，每种武器有初始熟练度 c_i。
- 参加 m 场战斗，每场战斗选择一种武器，第 j 场战斗中，武器熟练度会增加 a_j。

  注意 a_j 也有可能小于等于 0。
- 他想通过合理选择每场战斗使用的武器，使最终所有武器熟练度的最大值尽可能大。
- 数据范围：$n, m \leq 10^5$，c_i, a_j 在 $[-10^{4,10^4}]$。

B4051 小杨的武器

解法

当 $a_j > 0$ 时，武器可以增加熟练值（对最终答案有用）。

当 $a_j \leq 0$ 时，武器不会增加熟练值（对最终答案没有用）。

所以找出原来 c_i 最大的那一个，将所有 $a_j > 0$ 加给它即可。

当 $a_j \leq 0$ 时可以加给其他随便哪一个，不用关心。

注意

n 可能等于 1。这个情况下不得不将所有的 a_j 加给唯一的武器。

B4051 小杨的武器

```cpp
int n, m;
int c[100005], a[100005];
int main() {
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> c[i];
    for (int i = 1; i <= m; ++i) cin >> a[i];
    if (n == 1) {
        int ans = c[1];
        for (int i = 1; i <= m; ++i) ans += a[i];
        cout << ans << endl;
    } else {
        int ans = -20000;
        for (int i = 1; i <= n; ++i) ans = max(ans, c[i]);
        for (int i = 1; i <= m; ++i)
            if (a[i] > 0) ans += a[i];
        cout << ans << endl;
    }
}
```

P11960 平均分配

https://www.luogu.com.cn/problem/P11960

- 小 A 有 2n 件物品，小 B 和小 C 各刚好买 n 件；
- 每件物品两人出价不同（b_i 为小 B 出价，c_i 为小 C 出价）；
- 需计算小 A 卖出所有物品的最大收入；
- 输入为 n 及两人对 2n 件物品的出价，输出最大收入。

$1 \le n \le 10^5$，$0 \le b_i \le 10^9$，$0 \le c_i \le 10^9$

P11960 平均分配

解法

假设全部卖给小 B，那么可以获得一定的收入。

那么将哪一个卖给小 C，可以增加尽可能多的收入呢？

当然是，小 C 相比于小 B 出价更高，且差价最多的那个。

这个过程重复 n 次即可。

实现与优化

不需要每次挑选一个商品。

计算小 C 减去小 B 得到的差额。将差额从大到小排序。

其中最大的 n 个商品卖给小 C，剩下卖给小 B。

也有可能出现小 C 出价都低于小 B，没办法，亏得最少就是盈利。

P11960 平均分配

```cpp
int n;
int b[200005], c[200005];
int d[200005];
bool cmp(int x, int y) { return x > y; }
int main() {
    cin >> n;
    for (int i = 1; i <= 2 * n; ++i) cin >> b[i];
    for (int i = 1; i <= 2 * n; ++i) cin >> c[i];
    for (int i = 1; i <= 2 * n; ++i) d[i] = c[i] - b[i];
    sort(d + 1, d + 2 * n + 1, cmp);
    long long ans = 0;
    for (int i = 1; i <= n; ++i) ans += d[i];
    for (int i = 1; i <= 2 * n; ++i) ans += b[i];
    cout << ans << endl;
    return 0;
}
```

B3872 巧夺大奖

https://www.luogu.com.cn/problem/B3872

小明有 n 个时间段，每个时间段必须选择一个小游戏完成，共有n个小游戏可选。

每个小游戏 i 有一个期限 T_i（必须在第 T_i 个时间段结束前完成）和奖励 R_i，若超期则无奖励。

请安排游戏的完成顺序，使在期限内完成的游戏奖励总和最大。

$1 \le n \le 500$，$1 \le T_i \le n$，$1 \le R_i \le 1000$。

B3872 巧夺大奖

解法

贪心策略如下：

1. 按奖金排序，每次选择奖金最多的任务完成（因为钱多）。
2. 尽可能拖延完成时间（把前面的时间留给其他游戏）。

只要在指定时间前可以进行，就进行。

有没有可能放弃奖金更大游戏，使全局收益更大？

不可能。因为这个参加这个游戏，最多只会挤占掉另外最多一个游戏的机会。另外一个游戏的奖金不如这个。

B3872 巧夺大奖

```cpp
int n, v[505];
struct node {
    int t, r;
} a[505];
bool cmp(node x, node y) { return x.r > y.r; }
int main() {
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> a[i].t;
    for (int i = 1; i <= n; ++i) cin >> a[i].r;
    sort(a + 1, a + n + 1, cmp);
    for (int i = 1; i <= n; ++i)
        for (int j = a[i].t; j; --j)
            if (!v[j]) { v[j] = a[i].r; break; }
    int ans = 0;
    for (int i = 1; i <= n; ++i) ans += v[i];
    cout << ans << endl;
}
```