



程序设计与算法(二)

算法基础

郭 炜

微信公众号



微博: <http://weibo.com/guoweiofpku>

学会程序和算法，走遍天下都不怕!

讲义照片均为郭炜拍摄



北京大学
PEKING UNIVERSITY

信息科学技术学院

配套教材：

高等教育出版社

《算法基础与在线实践》

刘家瑛 郭炜 李文新 编著

本讲义中所有例题，根据题目名称在
<http://openjudge.cn>
“百练”组进行搜索即可提交





贪心算法



北京大学
PEKING UNIVERSITY

信息科学技术学院

例题:

Santa Clau' s Gifts



奥地利萨尔茨堡

圣诞老人的礼物-Santa Claus 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 ($1 \leq n \leq 100$)，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。



样例输入

4 15

100 4

412 8

266 7

591 2

样例输出

1193.0



圣诞老人的礼物-Santa Claus Gifts(百练4110)

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： $O(n\log n)$

圣诞老人的礼物-Santa Claus Gifts(百练4110)

```
const double eps = 1e-6;
struct Candy {
    int v; int w;
    bool operator < (const Candy & c) const
    { return double(v)/w - double(c.v)/c.w > eps; }
} candies[110];
int main() {
    int n,w;
    scanf("%d%d",&n,&w);
    for(int i = 0;i < n; ++i)
        scanf("%d%d", &candies[i].v , &candies[i].w);
    sort(candies,candies+n);
    int totalW = 0;
    double totalV = 0;
```

圣诞老人的礼物-Santa Claus Gifts(百练4110)

```
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);
return 0;
```

圣诞老人的礼物-Santa Clau' s Gifts(百练4110)

证明:

替换法。对于用非此法选取的最大价值糖果箱序列，
可以将其按价值/重量比从大到小排序后得到：

序列1: $a_1, a_2 \dots$

用序列1和按上述解法选取的序列2依次进行比较：

序列2: $b_1, b_2 \dots$

价值/重量比相同的若干箱糖果，可以合并成一箱，所以两个序列中元素都不重复

对于发现的第一个 $a_i \neq b_i$, 则必有: $a_i < b_i$

则在序列1中，用 b_i 这种糖果, 替代若干重量的 a_i 这种糖果，则会使得序列1的总价值增加，这和序列1是价值最大的取法矛盾

所以：序列1 = 序列2 （序列2不可能是序列1的一个前缀且比序列1短）

贪心算法

每一步行动总是按某种指标选取最优的操作来进行，该指标只看眼前，并不考虑以后可能造成的影响。

贪心算法需要证明其正确性。

“圣诞老人礼物”题，若糖果只能整箱拿，则贪心法错误。

贪心算法

每一步行动总是按某种指标选取最优的操作来进行，该指标只看眼前，并不考虑以后可能造成的影响。

贪心算法需要证明其正确性。

“圣诞老人礼物”题，若糖果只能整箱拿，则贪心法错误。

考虑下面例子：

3个箱子(8,6) (5,5) (5,5)，雪橇总容量10



北京大学
PEKING UNIVERSITY

信息科学技术学院

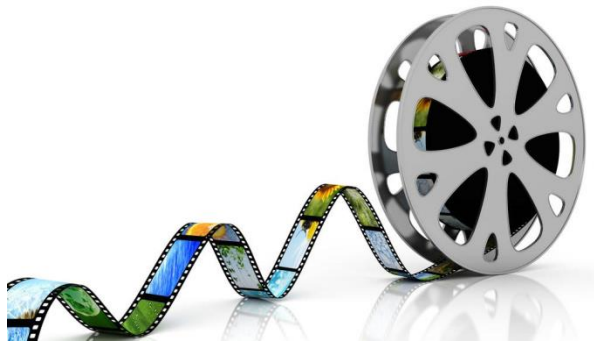
例题:电影节



维也纳美泉宫

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。



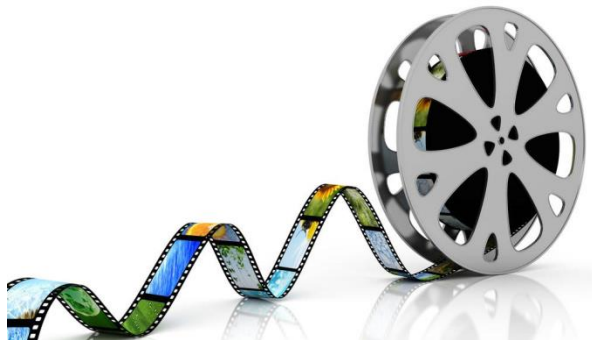
例题：电影节(百练4151)

输入

多组数据。每组数据开头是 n ($n \leq 100$)，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



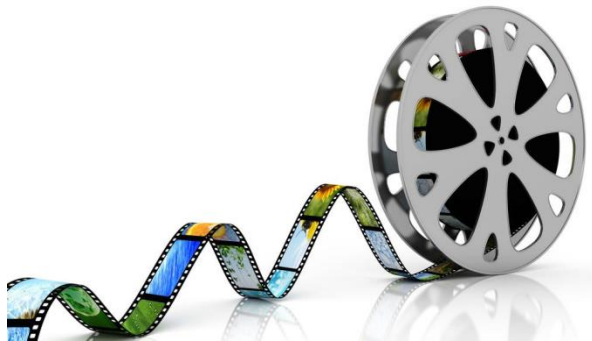
例题：电影节(百练4151)

Sample Input

```
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
0
```

Sample Output

```
5
```

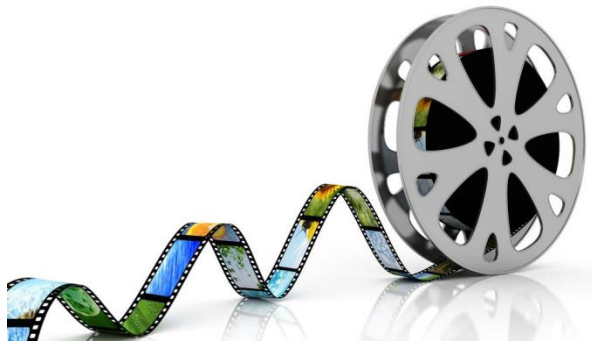


例题：电影节(百练4151)

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： $O(n\log n)$



例题：电影节(百练4151)

证明：

替换法。假设用贪心法挑选的电影序列为：

a_1, a_2, \dots

不用此法挑选的最长的电影序列为：

b_1, b_2, \dots

现可证明，对任意 i ， b_i 均可以替换成 a_i

例题：电影节(百练4151)

用 $S(x)$ 表示 x 开始时间， $E(x)$ 表示 x 结束时间，则：

- 1) b_1 可以替换成 a_1 ，因为 $E(a_1) \leq E(b_1)$
- 2) 若可以找到 a_i ，满足 $E(a_i) \leq E(b_i)$ 且 a_i 可以替换 b_i ，则 $E(a_{i+1}) \leq E(b_{i+1})$ 且 a_{i+1} 可以替换 b_{i+1}

证：

因为 $E(a_i) \leq E(b_i)$ 且 $E(b_i) \leq S(b_{i+1})$

则： $E(a_i) \leq S(b_{i+1})$

a_{i+1} 是所有 $S(x) \geq E(a_i)$ 的 x 中， $E(x)$ 最小的

$S(b_{i+1}) \geq E(b_i) \geq E(a_i)$ ，所以 $E(b_{i+1}) \geq E(a_{i+1})$

因此用 a_{i+1} 替换 b_{i+1} 不会对后续造成影响，替换可行



北京大学
PEKING UNIVERSITY

信息科学技术学院

例题

Stall Reservations



瑞士苏黎世

例题: Stall Reservations(POJ 3190)

有 n 头牛 ($1 \leq n \leq 50,000$) 要挤奶。给定每头牛挤奶的时间区间 $[A, B]$ ($1 \leq A \leq B \leq 1,000,000$, A, B 为整数)。

牛需要呆畜栏里才能挤奶。一个畜栏同一时间只能容纳一头牛。问至少需要多少个畜栏, 才能完成全部挤奶工作, 以及每头牛都放哪个畜栏里 (Special judged)

去同一个畜栏的两头牛, 它们挤奶时间区间哪怕只在端点重合也是不可以的。



例题： Stall Reservations(POJ 3190)

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 $E(x) \geq S(i)$, 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。

例题： Stall Reservations(POJ 3190)

证明：

由于按开始时间的顺序处理奶牛是必然，且按该算法，为奶牛 i 分配新畜栏时，确实是不得不分配的，所以算法正确。

复杂度： $O(n\log n)$

例题: Stall Reservations(POJ 3190)

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <vector>

using namespace std;

struct Cow {
    int a,b; //挤奶区间起终点
    int No; //编号
    bool operator<(const Cow & c) const {
        return a < c.a;
    }
} cows[50100];

int pos[50100]; //pos[i]表示编号为i的奶牛去的畜栏编号
```

```
struct Stall {
    int end;    //结束时间
    int No;     //编号
    bool operator<(const Stall & s) const {
        return end > s.end;
    }
    Stall(int e,int n):end(e),No(n) { }
};

int main()
{
    int n;
    scanf("%d",&n);
    for(int i = 0;i <n; ++i) {
        scanf("%d%d",&cows[i].a,&cows[i].b);
        cows[i].No = i;
    }
    sort(cows,cows+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, total));
        pos[cows[i].No] = total;
    }
    else {
        Stall st = pq.top();
        if( st.end < cows[i].a ) { //端点也不能重合
            pq.pop();
            pos[cows[i].No] = st.No;
            pq.push(Stall(cows[i].b, st.No));
        }
    }
}
```

```
        else { //对应 if( st.end < cows[i].a )
                ++total;
                pq.push(Stall{cows[i].b,total});
                pos[cows[i].No] = total;
            }
    }
    printf("%d\n",total);
    for(int i = 0;i < n;++i)
        printf("%d\n",pos[i]);

    return 0;
}
```



北京大学
PEKING UNIVERSITY

信息科学技术学院

例题 Radar Installation



瑞士卢塞恩

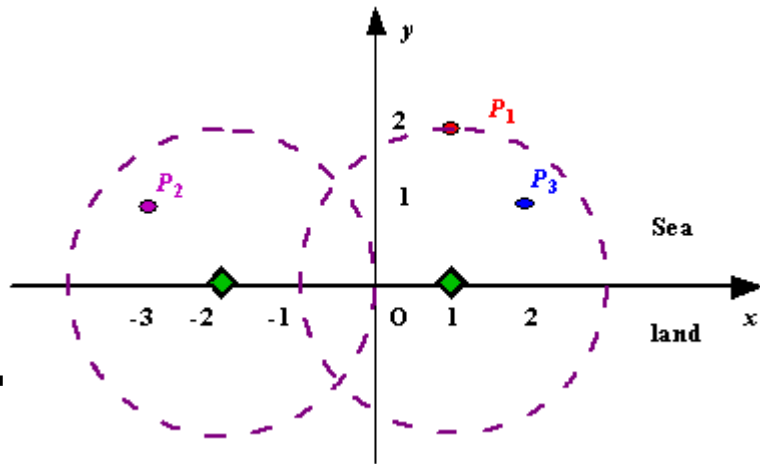
例题: Radar Installation(百练1328)

x 轴是海岸线, x 轴上方是海洋。海洋中有 n
($1 \leq n \leq 1000$)个岛屿, 可以看作点。

给定每个岛屿的坐标 (x, y) , x, y 都是整数。

当一个雷达 (可以看作点) 到岛屿的距离不超过 d (整数), 则认为该雷达覆盖了该岛屿。

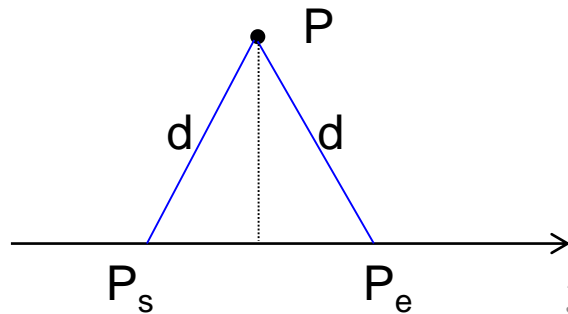
雷达只能放在 x 轴上。问至少需要多少个雷达才可以覆盖全部岛屿。



例题： Radar Installation(百练1328)

对每个岛屿 P ，可以算出，覆盖它的雷达，必须位于 x 轴上的区间 $[P_s, P_e]$ 。

如果有雷达位于某个 x 轴区间 $[a, b]$ ，称该雷达覆盖此区间。问题转换为，至少要在 x 轴上放几个雷达（点），才能覆盖全部区间 $[P1_s, P1_e], [P2_s, P2_e] \dots [Pn_s, Pn_e]$



例题：Radar Installation(百练1328)

重要结论：

如果可以找到一个雷达同时覆盖多个区间，那么把这多个区间按起点坐标从小到大排序，则最后一个区间（起点最靠右的） k 的起点，就能覆盖所有区间

证明：如果它不能覆盖某个区间 x ，那么它必然位于 1) x 起点的左边，或者2) x 终点的右边。

情况1) 和 k 的起点是最靠右的矛盾

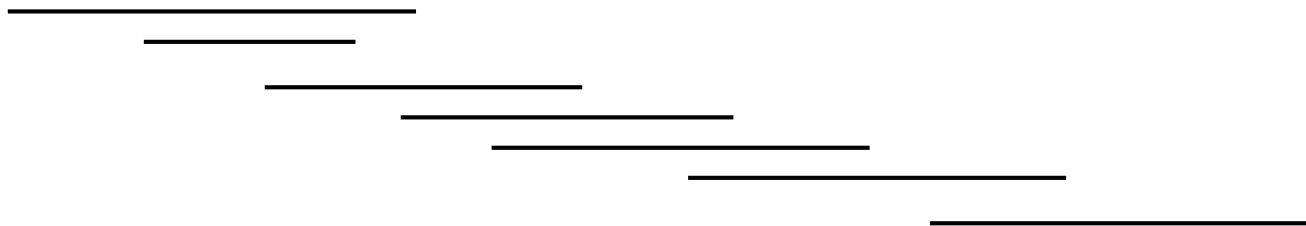
情况2) 如果发生，则不可能找到一个点同时覆盖 x 和 k ，也和前提矛盾

有了这个结论，就可以只挑区间的起点来放置雷达了。

例题： Radar Installation(百练1328)

贪心算法：

- 1) 将所有区间按照起点从小到大排序，并编号0 - (n-1)
- 2) 依次考察每个区间的起点，看要不要在那里放雷达。开始，所有区间都没被覆盖，所以目前编号最小的未被覆盖的区间的编号 `firstNoConverd = 0`



例题：Radar Installation(百练1328)

3) 考察一个区间 i 的起点 x_i 的时候，要看从 firstNoCovered 到区间 $i-1$ 中是否存在某个区间 c ，没有被 x_i 覆盖。如果没有，则先不急于在 x_i 放雷达，接着往下看。如果有，那么 c 的终点肯定在 x_i 的左边，因此不可能用同一个雷达覆盖 c 和 i 。即能覆盖 c 的点，已经不可能覆盖 i 和 i 后面的区间了。此时，为了覆盖 c ，必须放一个雷达了，放在区间 $i-1$ 的起点即可覆盖所有从 firstNoCovered 到 $i-1$ 的区间。因为当初考察 $i-1$ 的起点 z 时候，并没有发现 z 漏覆盖了从 firstNoCovered 到 $i-2$ 之间的任何一个区间

4) 放完雷达后，将 firstNoCovered 改为 i ，再做下去。

例题：Radar Installation(百练1328)

3) 考察一个区间 i 的起点 x_i 的时候，要看从 firstNoCovered 到区间 $i-1$ 中是否存在某个区间 c ，没有被 x_i 覆盖。如果没有，则先不急于在 x_i 放雷达，接着往下看。如果有，那么 c 的终点肯定在 x_i 的左边，因此不可能用同一个雷达覆盖 c 和 i 。即能覆盖 c 的点，已经不可能覆盖 i 和 i 后面的区间了。此时，为了覆盖 c ，必须放一个雷达了，放在区间 $i-1$ 的起点即可覆盖所有从 firstNoCovered 到 $i-1$ 的区间。因为当初考察 $i-1$ 的起点 z 时候，并没有发现 z 漏覆盖了从 firstNoCovered 到 $i-2$ 之间的任何一个区间

4) 放完雷达后，将 firstNoCovered 改为 i ，再做下去。

复杂度： $O(n^2)$

例题： Radar Installation(百练1328)

证明：

替换法。考虑不用贪心法获得的最佳雷达摆放方案。将其所有雷达按坐标从小到大排序得到 x_1, x_2, \dots 。用贪心法得到的雷达坐标从小到大排序则为 y_1, y_2, \dots 。

可证明每个 x_i 都可以被 y_i 替换，且 y 序列不会比 x 序列长。

先证明 x_1 可以用 y_1 替换。

用 $S(x)$ 表示贪心法中区间 x 的起点, 假设 $y_1 = S(i)$

a) 若 $x_1 < y_1$, 则用 y_1 替换 x_1 没问题, 因 y_1 覆盖了区间0到 i , x_1 覆盖的区间更少

b) 若 $x_1 > y_1$, 则分两种情况讨论:

1) $x_1 < S(i+1)$: 因 x_1 不能覆盖 $i+1$ 及以后区间, 且 i 及以前的区间已经被 y_1 覆盖, 所以将 x_1 用 y_1 替换, 不会有损失。

2) $x_1 \geq S(i+1)$

贪心法中, 在区间 i 起点放雷达, 是因为如果不放而在 $S(i+1)$ 放雷达, 则该雷达不能覆盖 i 及其前面的某个区间 C 。

若 $x_1 \geq S(i+1)$, 则 x_1 也不能覆盖 C 。 $x_i (i > 1)$ 更加不能。因此 $x_1 \geq S(i+1)$ 是不可能的。

类似地证明, 假设 x_i 可以被 y_i 替换, 则 x_{i+1} 也可以被 y_{i+1} 替换。



北京大学
PEKING UNIVERSITY

信息科学技术学院

例题 Gone Fishing



瑞士苏黎世

例题： Gone Fishing(百练1042)

有 n ($2 \leq n \leq 25$) 个湖从左到右一字排开。从第 i 个湖走到第 $i+1$ 个湖要耗时 $t[i]$ 个时间片（每个时间片5分钟）。

John有 h ($1 \leq h \leq 16$) 个小时可以用在这些湖钓鱼（包括湖间行走时间）。在每个湖待的时间必须是整数个时间片或0。就算钓不着鱼了，也可以在湖边呆着。

对于湖 i ，John在那里的第一个时间片可以钓到鱼 $f[i]$ 条，且后续每个时间片，能钓到的鱼数量都比上一个时间片少 $d[i]$ 。

注意John只能从第一个湖出发，从左往右走，不能回头。最后John要停在哪里都可以。问John最多能钓多少条鱼。还要输出钓鱼方案，即在每个湖各呆多长时间。如果有多种方案，则优先选择在第一个湖呆时间最长的。如果还有多种，则优先选择在第二个湖呆的时间最长的.....

例题： Gone Fishing(百练1042)

难点：

走路时间可多可少，不知道到底该花多长时间纯钓鱼才最好
(可能有好湖在很右边)。



例题： Gone Fishing(百练1042)

难点：

走路时间可多可少，不知道到底该花多长时间纯钓鱼才最好(可能有好湖在很右边)。

解决：

枚举最终停下来的湖，将方案分成 n 类。每类方案的走路时间就是确定的。在每类方案里找最优解，然后再优中选优。

例题： Gone Fishing(百练1042)

贪心：

在确定停下来的湖是 x 的情况下，假定纯钓鱼时间是 k 个时间片。

用三元组 (F, i, j) ($1 \leq i \leq x$, $1 \leq j \leq k$) 表示湖 i 的第 j 个时间片能够钓的鱼的数目是 F

将所有的 (F, i, j) (共 $x * k$ 个) 按 F 值从大到小排序，选前 k 个，就构成了最佳钓鱼方案

例题： Gone Fishing(百练1042)

$x = 3, k = 4$

1

18
15
12
9

2

17
16
15
14

3

27
7
0
0