



武松打虎



西南大学附属中学
High School Affiliated to Southwest University

问题描述:

曾经因打虎而文明的武松在x年后接到了景阳冈动物园的求助信,信上说:最近我们动物园逃跑了几只老虎,请您把它们抓回来, thank you!武松接到信后立刻上了山。

正当他到半山腰是, suddenly! 跳出n只猛虎来。每只老虎都有一块虎牌,牌上写着的是每一只虎最大拥有的体力,当武松与老虎PK时,若老虎的体力先用完,那么老虎over,否则武松over。求武松在over之前最多能干掉几只老虎?

(注:老虎是一只只上的)

输入:

第一行两个数字n($n < 50000$), t0(武松的体力)。第二行n个数字,分别代表每只老虎的体力。所有变量都不超过long int 范围。

输出:

一行,最多能干掉的老虎数。

样例输入:

```
6 10
1 5 3 2 4 6
```

样例输出:

```
4
```

Q: 如何打虎，才能干掉最多的老虎？

1. 以我们考试解题为例：

在考试时，如果每道题分值相同的话，时间一定的情况下为了得到更高的分数我们会倾向于先做简单题再做难题，因为难题比简单题费时更长、更难得分。

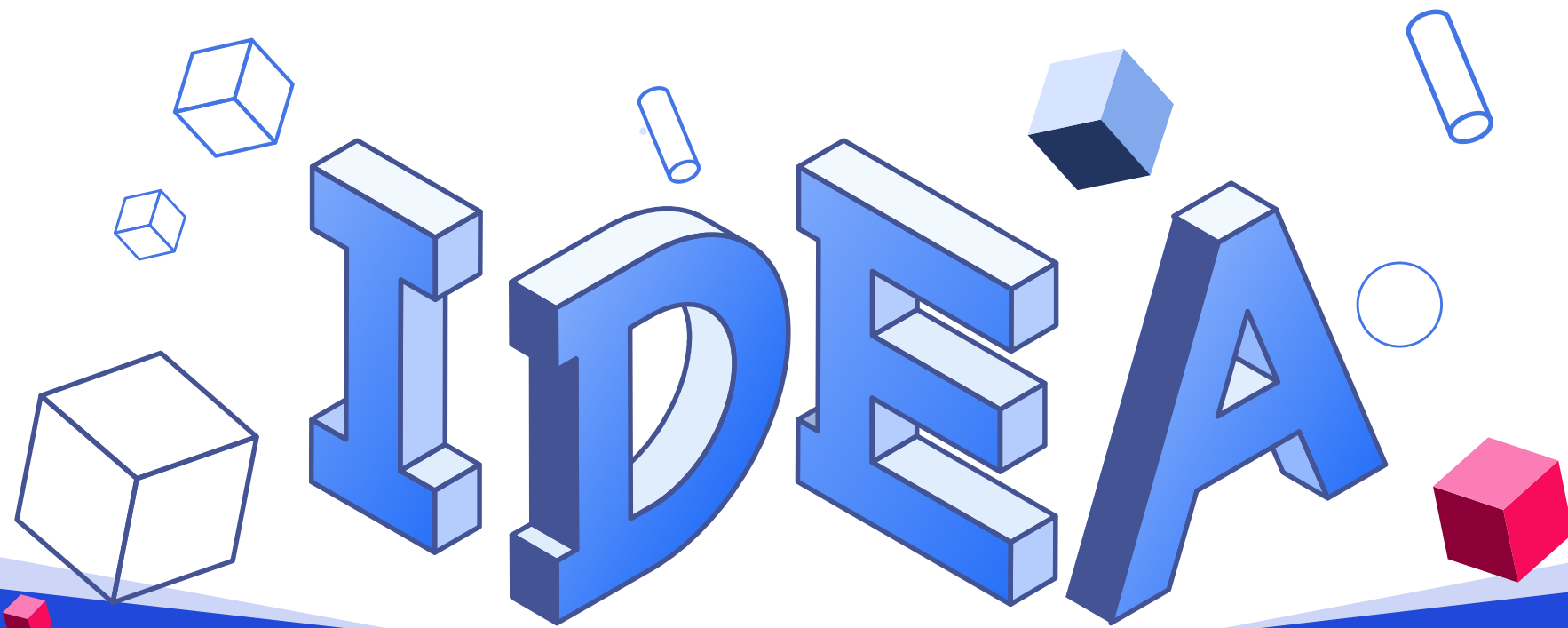
2. 那么假设打死一只老虎得分为1，它的体力表示难易程度，那么我们很容易得到这样的策略：

每次打体力最小的老虎 → 武松能剩余更多体力，打更多的老虎



贪心算法

在现实生活中，我们常常会想着以目前看起来“最好，最优，最有效率”的方式来解决问题，其实这就是贪心的体现。



信息学 贪心算法

西南大学附属中学校
信息奥赛教练组



基本思想：

将问题的求解过程看作是一系列选择，每次选择都是**当前状态下的最好选择**(**局部最优解**)，不断地做出选择，直到问题求解完成。

以武松打虎为例：

1. $t_0=10$ ，干掉体力为1 5 3 2 4 6的老虎 **选择干掉1**

↓ 问题转化为

2. $t_0=9$ ，干掉体力为5 3 2 4 6的老虎 **选择干掉2**

↓ 问题转化为

3. $t_0=7$ ，干掉体力为5 3 4 6的老虎 **选择干掉3**

↓ 问题转化为

4. $t_0=4$ ，干掉体力为5 4 6的老虎 **选择干掉4**

↓ 问题转化为

5. $t_0=0$ ，干掉体力为5 6的老虎 **over**

每次都是当前问题最好的选择，即打死体力值最小的老虎

贪心选择前，先排序

贪心解题需满足的两个性质：

1.最优子结构性质

2.贪心选择性质



1.最优子结构性质



西南大学附属中学
High School Affiliated to Southwest University

当一个问题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。

Q: 什么是子问题?

1. $t_0=10$, 干掉体力为1 5 3 2 4 6的老虎 **选择干掉1**

↓ 问题转化为

2. $t_0=9$, 干掉体力为5 3 2 4 6的老虎 **选择干掉2**

↓ 问题转化为

3. $t_0=7$, 干掉体力为5 3 4 6的老虎 **选择干掉3**

↓ 问题转化为

4. $t_0=4$, 干掉体力为5 4 6的老虎 **选择干掉4**

通过每次最优的选择，问题慢慢变成一个**问题描述**和**所求答案与原问题相同**但**数据规模更小**的问题，称为**子问题**



1.最优子结构性质



西南大学附属中学
High School Affiliated to Southwest University

问题1. $t_0=10$, 干掉体力为1 5
3 2 4 6的老虎



答案的解为: 1 2 3 4

子问题



问题2. $t_0=9$, 干掉体力为5 3 2
4 6的老虎



答案的解为: 2 3 4



左边大问题的解包含右边子问题的解



2.贪心选择性质



西南大学附属中学
High School Affiliated to Southwest University

所求问题的**整体最优解可以通过一系列局部最优的选择得到**，
也就是说，当前的选择只依赖于之前的选择，不依赖于后面的选择

武松打虎的贪心策略：每次都打死体力值最小的老虎

我们知道，武松剩余的体力越多能够打死的老虎就越多。
如果我们采用上述的贪心方式，能够保证每次打死老虎后武松剩余的体力最大，
最终能保证武松打死的老虎最多。
即通过一系列局部最优解得到整体最优解，这种贪心策略满足贪心选择性质。

贪心选择性质的关键是**选取怎样的贪心策略**



疑问

贪心算法如何保证每一步的最优解最终能逐步达到整体的最优解呢？

需要进行正确性证明



贪心算法求解的一般步骤：

- 1.通过问题信息找规律、分析归纳(**一般要结合排序找规律**)
- 2.进行贪心猜想
- 3.对猜想进行正确性证明(严格证明和一般证明)
 - (1) 严格证明：数学归纳法、反证法、矩阵胚理论
 - (2) 一般证明：列举反例即可
- 4.设计并实现程序

经典应用： 活动安排、乘船问题、区间问题等



例1：独木舟的旅行



西南大学附属中学
High School Affiliated to Southwest University

进行一次独木舟的旅行活动，独木舟可以在港口租到，并且之间没有区别。一条独木舟最多只能乘坐两个人，且乘客的总重量不能超过独木舟的最大承载量。我们要尽量减少这次活动中的花销，所以要找出可以安置所有旅客的最少的独木舟条数。现在请写一个程序，读入独木舟的最大承载量、旅客数目和每位旅客的重量。根据给出的规则，计算要安置所有旅客必须的最少的独木舟条数，并输出结果。

输入

第一行输入s,表示测试数据的组数;

每组数据的第一行包括两个整数w, n,

$80 \leq w \leq 200, 1 \leq n \leq 300$, w为一条独木舟的最大承载量,n
为人数;

接下来的一组数据为每个人的重量（不能大于船的承载
量）;

输出

每组人数所需要的最少独木舟的条数。

样例输入

2

85 6

5 84 85 80 84 83

90 3

90 45 60

样例输出

5

3

题意

每条船的承载量
为C,怎么安排让
船的数量最少

Q: 如何才能使船的数目最少?

每艘船都尽可能达到最大承载量C



- 策略:
- 1) 将所有人的重量 w 进行排序;
 - 2) 从当前最轻的重量 w_i 开始考虑, 找能跟其坐一只船的最重的人 w_j
 - 3) 比最重的人 j 都重的人都单独坐一个船;



反证法：

1. 假设 i 不与任何人同船，如果将 j 拉来与其同船，用的船数会小于等于原来的船数；
 2. 假设 i 与 k 同船，因为 j 是与 i 匹配的最重的，所以 $w(k) \leq w(j)$ ，则 j 加入其它船可能会使其它船超重，用的船数会变多；
- 综上，说明这样的贪心法不会丢失最优解。



例二、最大整数



西南大学附属中学
High School Affiliated to Southwest University

设有 n 个正整数 ($n \leq 20$)，将它们连接成一排，组成一个最大的多位整数。

例如： $n=3$ 时，3个整数13,312,343,连接成的最大整数为：34331213

又如： $n=4$ 时，4个整数7,13,4,246连接成的最大整数为：7424613

输入

第一行：正整数 n ($n \leq 20$)

第二行： n 个整数，两个数之间用一个空格隔开

输出

连接成的多位数

样例输入

3

13 312 343

样例输出

34331213

目的：使得连接起来的数字所代表的整数最大

通过分析样例猜想：

按数字的字典序比较， $342 > 312 > 13$ ，最终结果是按照字典序从大到小进行连接。

正确性证明：

反例：

例如 $A = "12"$ ， $B = "121"$ ， $121 > 12$ ，而明显 $12121 > 12112$ ，即 $AB > BA$ 。

进一步思考：

可以发现，字符串A和B的大小关系并不是他们本身来决定，而是他们组合后的数来决定。

这里应该**判断AB和BA的大小关系**来判断A和B的大小关系：
若是 $AB > BA$ 则 $A > B$ ，若是 $AB < BA$ 则 $A < B$ ，否则 $A == B$ 。

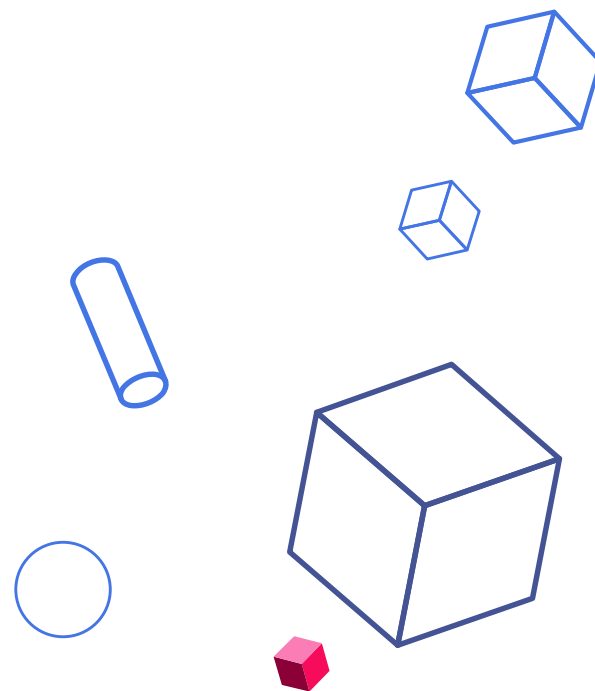
最终的贪心策略：按照上述的比较规则对所有字符串从大到小排序，依次输出即可。



贪心算法在解决问题的策略上**目光短浅**，只根据当前已有的信息就做出选择，而且一旦做出了选择，不管将来有什么结果，这个选择都不会改变。换言之，贪心法并不是从整体最优考虑，它所做出的选择只是在某种意义上的局部最优。

如果贪心算法被证明是不正确的，最后得到的解只能是一个**较优解**

今天：贪心经典 模型-区间问题





例：会场安排问题(选择不相交区间)



西南大学附属中学
High School Affiliated to Southwest University

学校的小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小礼堂的活动，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，他尽可能的安排更多的活动，请问他该如何安排。

输入

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 B_i, E_i ($0 \leq B_i, E_i < 10000$), 分别表示第 i 个活动的起始与结束时间 ($B_i \leq E_i$) 输出对于每一组输入，

输出

最多能够安排的活动数量。每组的输出占一行

样例输入

3

1 10

11 20

10 15

样例输出

2

题意

选择安排策略，在某个时间区间尽可能地多安排活动

如何来选择活动？

猜想1：以活动开始时间来采取贪心策略，选择安排较早开始的活动

验证猜想：



西 | 大 | 附 | 中 | 信 | 息 | 学 | 竞 | 赛 |
High School Affiliated to Southwest University

很明显，如果按照开始时间来选择，在上面这种情况下，只能选择活动1。而更好的选择是活动2和3。也可以发现，影响我们选择活动的关键是**活动什么时候结束**。

猜想2：以活动结束时间来采取贪心策略，选择安排较早结束的活动



如果按照活动结束时间的先后来选择，我们会选择到活动2和3

正确性：当一个活动越早结束剩余的时间就越多，相应地能够选择更多的活动的可能性就更大。

最终贪心策略的选择:

活动按结束时间**从小到大**排序，每次都选择安排结束时间较早的活动

这类问题也叫做**选择不相交区间**问题，开始时间就是区间的左端点，结束时间就是区间的右端点

核心代码:

```
//记得先排序，再贪心
int t, ans=1; //t记录当前的右端点，ans统计选择的区间数
t=a[1].r;    //初始化为第一个区间的右端点
for(int i=2; i<=n; i++){
    if(a[i].l>=t){ //如果当前区间与下面的区间没有重合
        ans++;    //选择该区间
        t=a[i].r; //更新当前的右端点
    }
}
```




例：区间选点问题



西南大学附属中学
High School Affiliated to Southwest University

数轴上有 n 个闭区间 $[a_i, b_i]$ ，选取尽量少的点，使得每区间都至少包含一个点（同一个点，可被多个区间包含）

输入

第一行 n ($n < 1000000$) 之后 n 行，每行两个数分别为 a_i, b_i

输出

最少需要的点的个数

样例输入

3

1 3

2 4

3 5

样例输出

1

我们的目标是每次选取的点能覆盖到最多区间



1. 首先满足每个区间至少有一个点
2. 其次，那么我们在选取点的时候就可以尽量往后选取，让这个点能覆盖到下面更多的区间，所以这个点是右端点。



贪心策略：按照右端点从小到大排序，若右端点相同，按左端点从大到小排序

正确性证明：<https://blog.csdn.net/dgq8211/article/details/7534776>



算法流程：

- 输入区间 $[a_i, b_i]$
- 对区间进行排序，先按右端点从小到大排序，若右端点相同，按左端点从大到小排序
- end 保存当前选择点的位置，遍历所有区间如果 $end < a_i$ ，更新当前 end ，选择的点数+1



例：区间覆盖问题



西南大学附属中学
High School Affiliated to Southwest University

数轴上有 n 个闭区间 $[a_i, b_i]$ ，选取尽量少的区间覆盖一条指定线段 $[s, t]$ 。

输入

第一行 n ($n < 1000000$)

之后 n 行，每行两个数分别为 a_i, b_i

最后一行为 $s\ t$

输出

最少需要选择的区间个数，如果无解则输出“no solution!”

样例输入

3

1 3

2 4

3 5

2 2

样例输出

1



区间情况

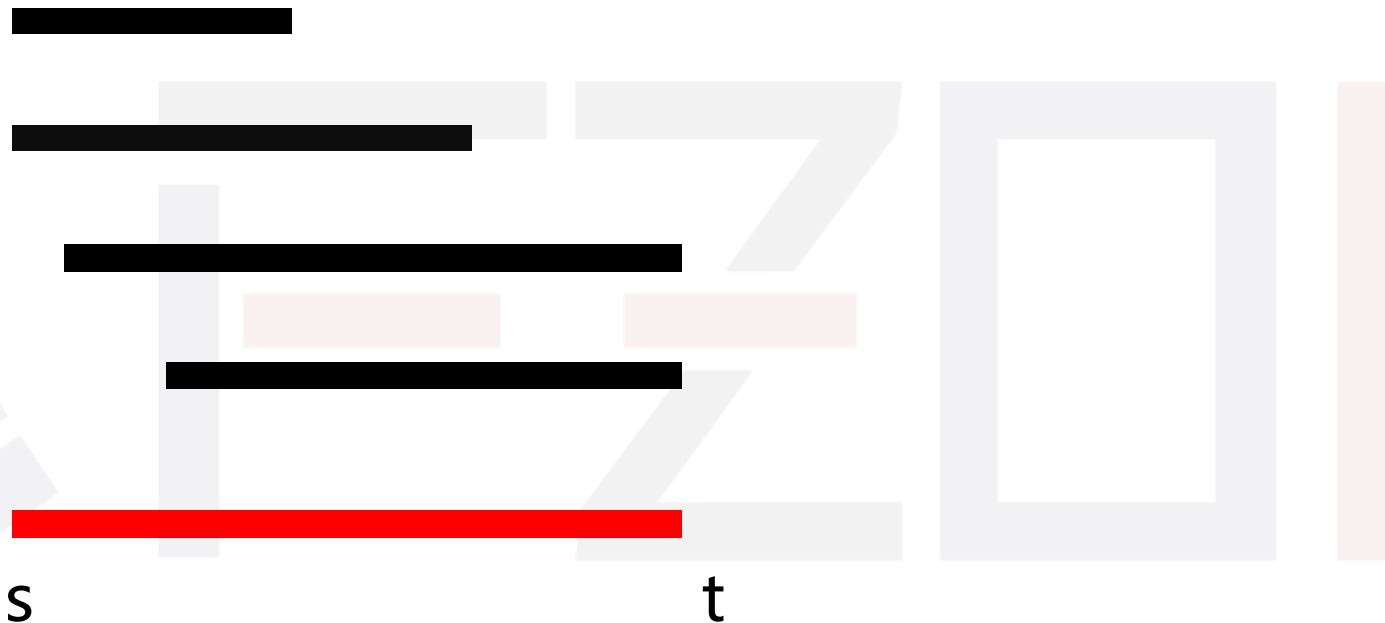


给定的区间超出了目标区间 $[s, t]$ 的部分都是无效的



区间情况

目标区间:



删去无效区间后，可以发现左端点相同时，能使区间覆盖更长的是右端点，所以选择右端点尽可能大的区间

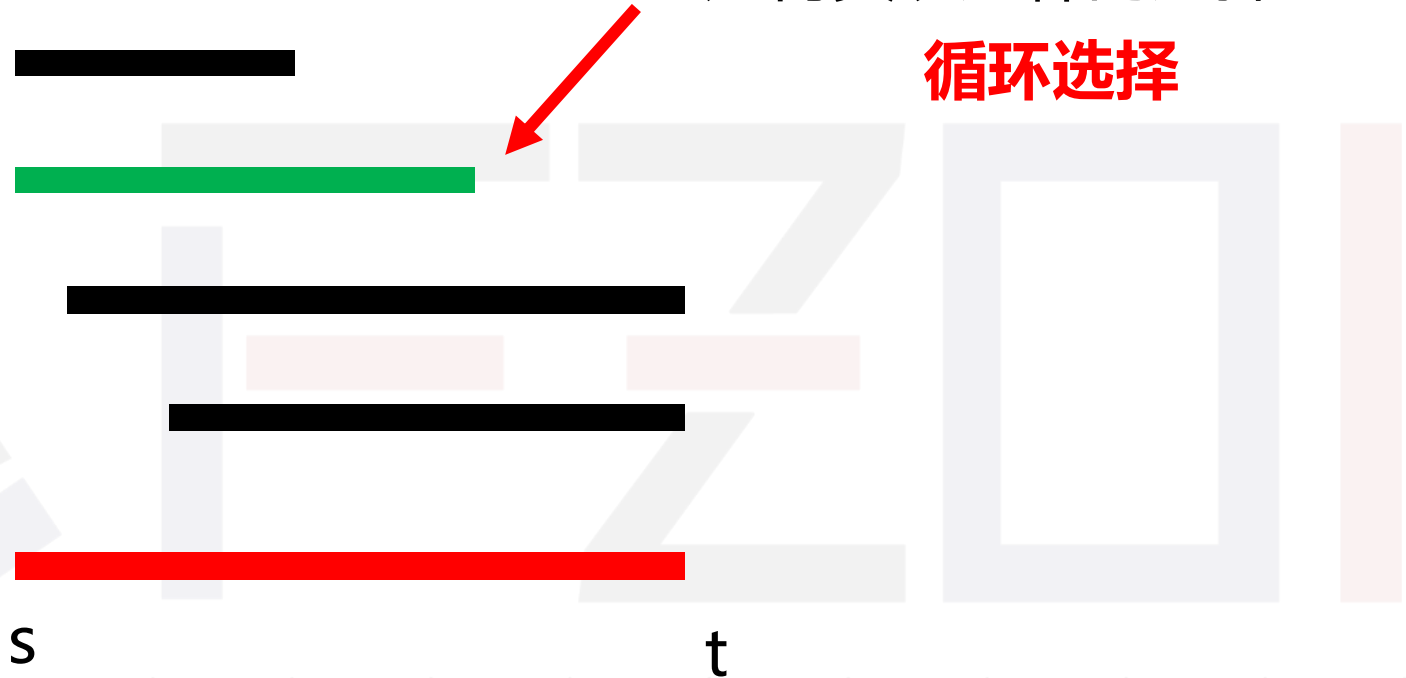


区间情况

如何实现这样的选择？

循环选择

目标区间：



删去无效区间后，可以发现左端点相同时，能使区间覆盖更长的是右端点，所以选择右端点尽可能大的区间



区间情况

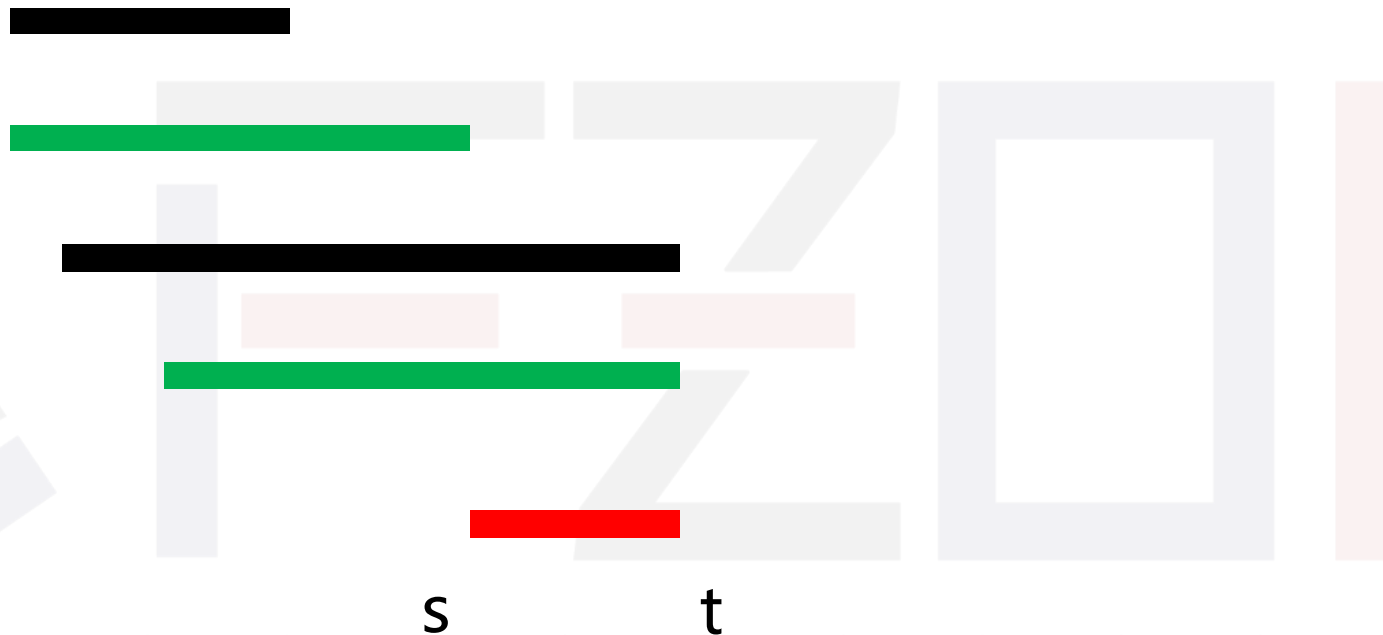
目标区间:



缩小目标区间



区间情况



目标区间：

选择小于左端点 s ，且右端点最长的区间，直到区间覆盖完成

问题：区间是否需要排序？以什么标准来排序？

为了保证起点被覆盖的情况下选择最长区间，**以左端点从小到大来排序**



1. 如何选择右端点最长的区间？

```
//定义了c是区间结构体数组，left是左端点，right是右端点
while(c[i].left<=s&& i<=n){ //如果当前区间左端点小于s
    end=max(end,c[i].right); //end保存最长区间的右端点
    i++; //继续判断下一个区间
}
```

2. 什么时候题目无解？

所有区间都无法覆盖完整个目标区间

3. 什么时候结束覆盖？

我们默认是从左端点开始覆盖，当右端点也被覆盖到了就结束。



算法流程：

- 输入需要覆盖的区间 $[s, t]$ ，及其他区间 $[a_i, b_i]$
- 对所有区间按照左端点排序
- 如果第一个区间的起点大于 s ，无解，否则选择 $a_i \leq s$ ， b_i 最大的区间。
- 选择此区间 $[a_i, b_i]$ 后，新的起点应该设置为 b_i
- 如果 n 条线段的都已选择，但还是无法覆盖 t ，无解。
- 如果某次选择已经把 t 覆盖，即 $b_i \geq t$ ，结束覆盖，输出答案

A 3D rendering of the word "IDEA" in large, blue, blocky letters. Surrounding the letters are various geometric shapes: a large white cube with blue outlines on the left, a small red cube at the bottom left, a blue cube on the right, and several blue cylinders and smaller cubes floating around. Above the letters is the logo for FZOI (Fuzhou Zhongxin Information School). The logo consists of a stylized "FZOI" in black and red, with the Chinese text "西大附中信息学竞赛" and the English text "High School Affiliated to Southwest University" below it. The background is a light blue gradient with a faint, large-scale geometric pattern.



A 3D rendering of the word "IDEA" in large, blue, blocky letters. The letters are surrounded by various geometric shapes, including cubes and cylinders, some in white and some in red. Above the letters is the logo for FZOI (Fuzhou Zhongxin Information Science School), which includes the text "西大附中信息学竞赛" and "High School Affiliated to Southwest University". The background is a light blue gradient with a faint circular pattern.