程序设计思维与实践

Thinking and Practice in Programming

贪心、二分 | 内容负责：魏安然/苗顺源

1 贪心算法初试

The beginning of the greedy algorithm

贪心热身

- 部分背包问题
  - 有 n 个物品，第 i 个物品的重量为 w_i，价值为 v_i。
  - 你有一个容量为 c 的背包，可以往里面放物品，尽量让总价值最高。
  - 每个物品可以只取走一部分，价值和重量按比例计算。
  - 样例：

    n = 5

    v = [5, 1, 3, 2, 5]

    w = [6, 3, 2, 1, 4]

    C = 5

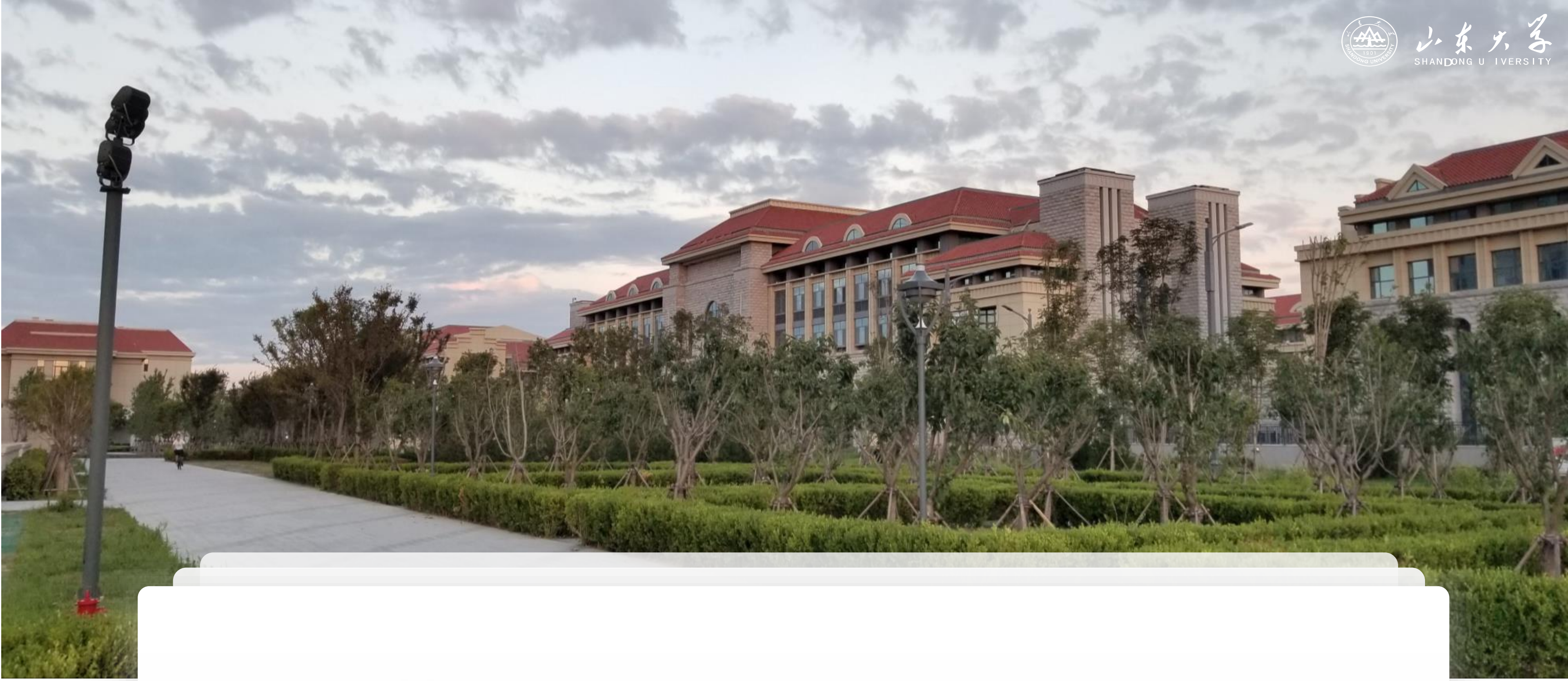
    答案为：？
  - 策略：？
    - 优先拿“单位重量价值”最大的，直到重量和正好为 c。
    - v/w = [0.83, 0.33, 1.5, 2, 1.25]

贪心热身

- 乘船问题
  - 有 n 个人，第 i 个人重量为 w_i。
  - 每艘船的最大载重量均为 C，且最多只能乘 2 个人。
  - 用最少的船装载所有的人。
  - 策略?
    - 最轻的人应该要选能和他一起坐船的人中最重的一个

贪心热身

- 乘船问题
  - 考虑最轻的人 u，他应该和谁一起坐呢？如果每个人都无法和他一起坐船，则唯一的方法就是每人各坐一艘船（想想为什么？）
    否则，他应该选择能和他一起坐船的人中最重的一个 v。
  - 因为剩下的人越轻越好



2 贪心算法

The greedy algorithm

贪心算法

贪心算法（英语：greedy algorithm），是用计算机来模拟一个“贪心”的人做出决策的过程。这个人十分贪婪，每一步行动总是按某种指标选取最优的操作。而且他目光短浅，总是只看眼前，并不考虑以后可能造成的影响。

可想而知，并不是所有的时候贪心法都能获得最优解，所以一般使用贪心法的时候，都要确保自己能证明其正确性。

贪心算法

证明方法¶

贪心算法有两种证明方法：反证法和归纳法。一般情况下，一道题只会用到其中的一种方法来证明。

- 反证法：如果交换方案中任意两个元素/相邻的两个元素后，答案不会变得更好，那么可以推定目前的解已经是最优解了。

- 归纳法：先算得出边界情况（例如 n=1 ）的最优解F_1，然后再证明：对于每个F_{n+1}，都可以由 F_n 推导出结果。

贪心算法

最常见的贪心题型有两种。
1.我们将 XXX 按照某某顺序排序，然后按某种顺序（例如从小到大）选择。
2.我们每次都取 XXX 中最大/小的东西，并更新 XXX。

二者的区别在于一种是离线的，先处理后选择；一种是在线的，边处理边选择。

贪心算法

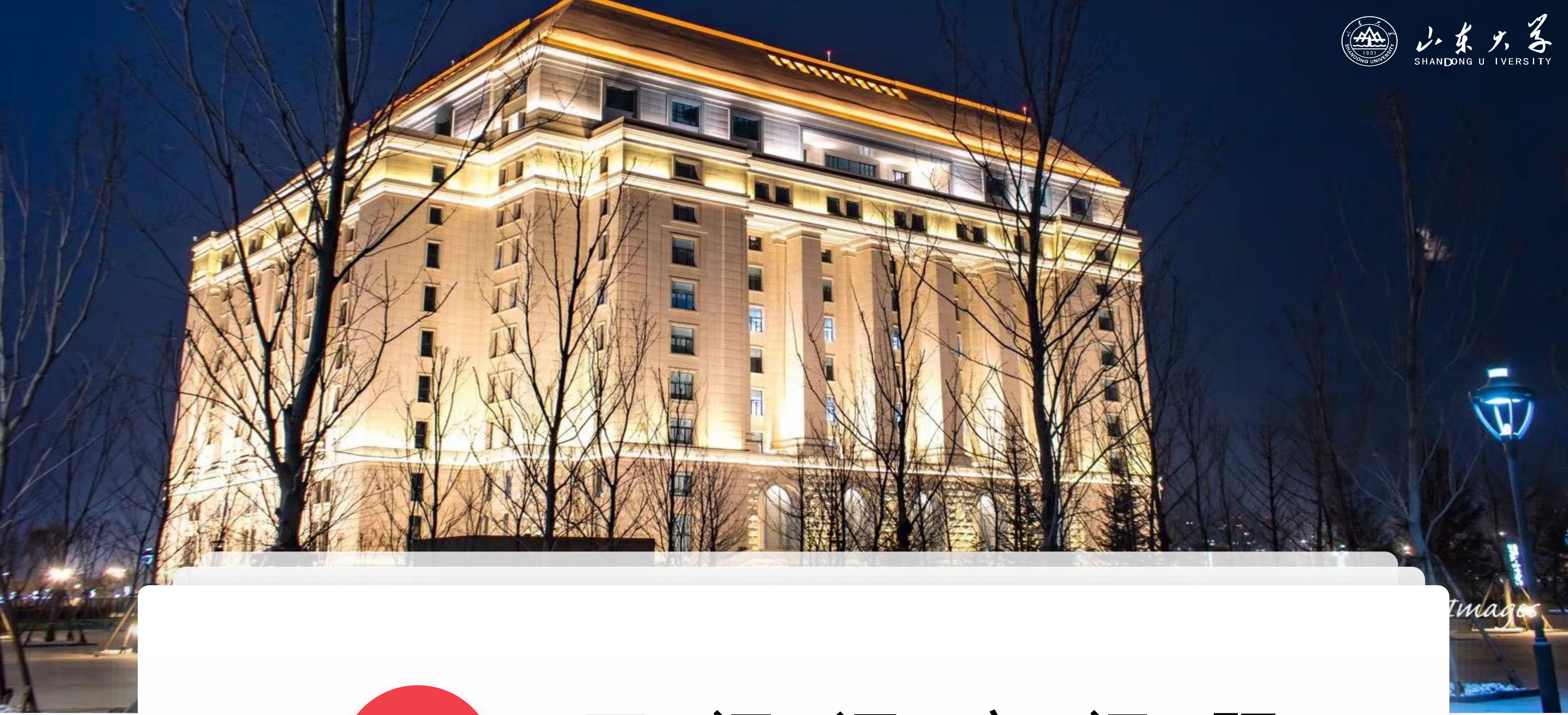
最常见的贪心解法有两种。

排序解法

用排序法常见的情况是输入一个包含几个权值的数组，通过排序然后遍历模拟计算的方法求出最优值。

后悔解法

思路是无论当前的选项是否最优都接受，然后进行比较，如果选择之后不是最优了，则反悔，舍弃掉这个选项；否则，正式接受。如此往复。

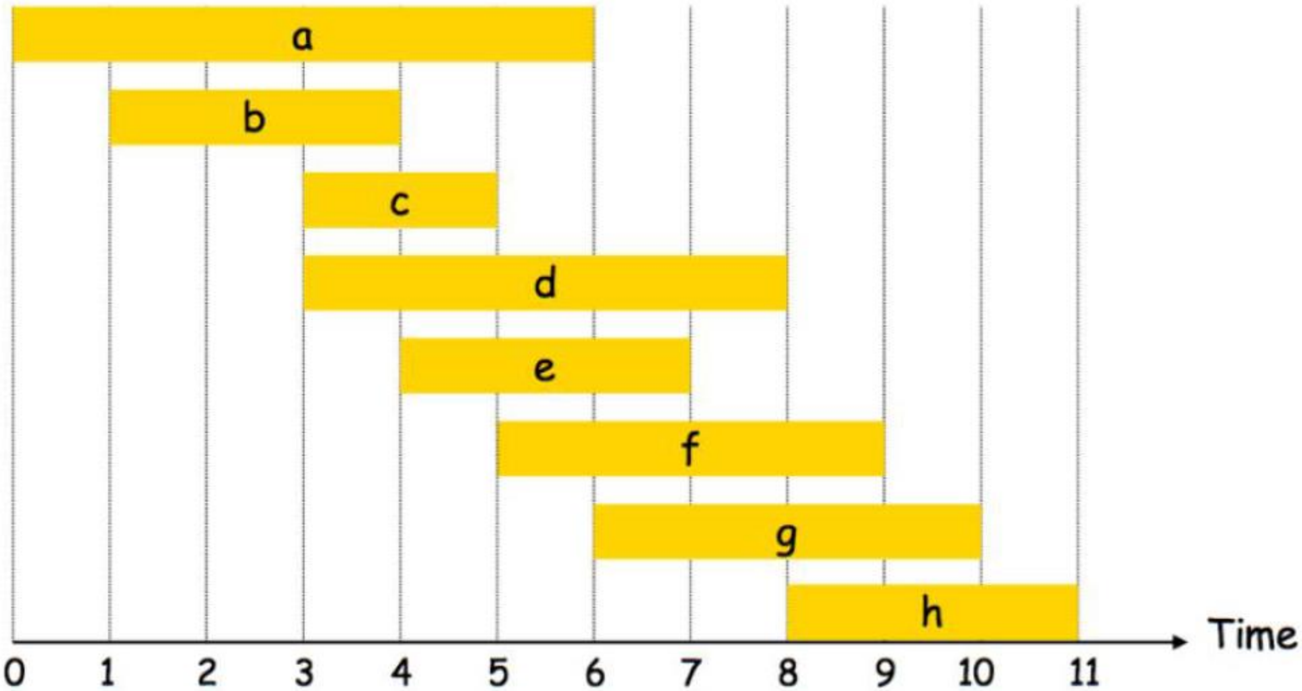


3 区间调度问题

Interval scheduling problem

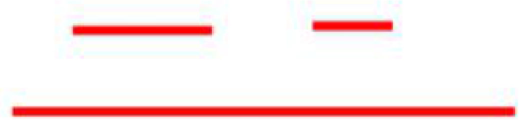
区间调度问题

- 区间调度问题
  - 区间 j 从 S_j 时刻开始，到 F_j 时刻结束 （$F_j > S_j$）
  - 两个区间 i 和 j 相容当且仅当它们不重叠 （$F_i <= S_j$ 或者 $F_j <= S_i$）
  - 目标：找到最大的由互相相容的区间组成的子集的大小



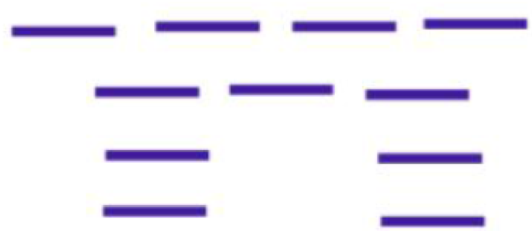
区间调度问题

- 区间调度问题 —— 贪心算法
  - 前面提到了贪心算法要有一个指标，现在我们要选择一个贪心指标，根据它，来优先选择一些区间
  - 我们应该用哪些指标?
  - 最早的开始时间 S_i ?
    - 反例



  - 最短的区间长度 F_i - S_i ?
    - 反例

  - 最少冲突的区间 ?
    - 反例



  - 选择 —— 最早的完成时间 F_i 或者最晚的开始时间S_i

区间调度问题

- 区间调度问题 —— 贪心算法
  - 选择 —— 最早的完成时间 F_i
  - 怎么证明?

Init：令 R 是所有区间的集合，A 为空

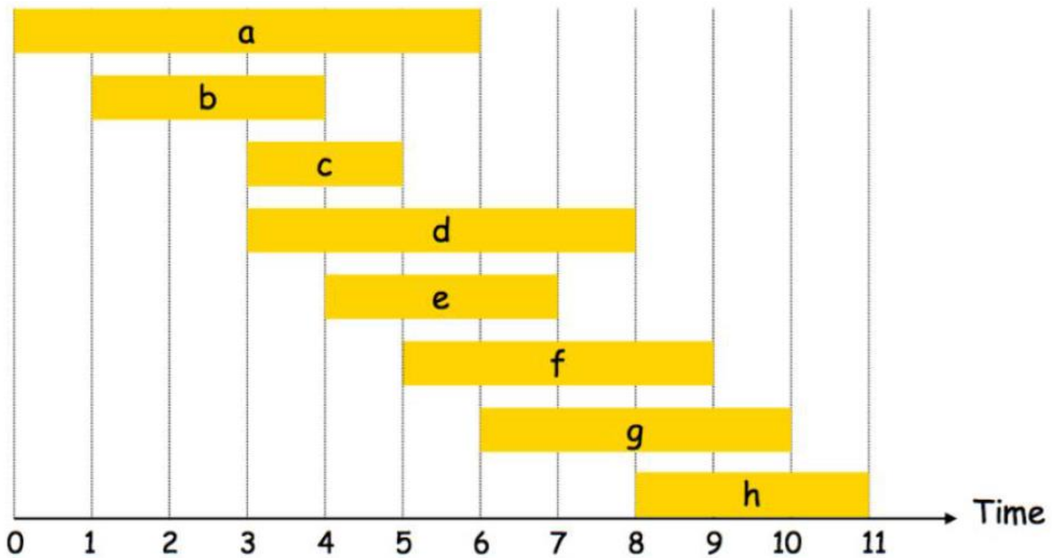
While R 不空：

    选择 R 中最小结束时间的区间 i

    把 i 加到 A 中

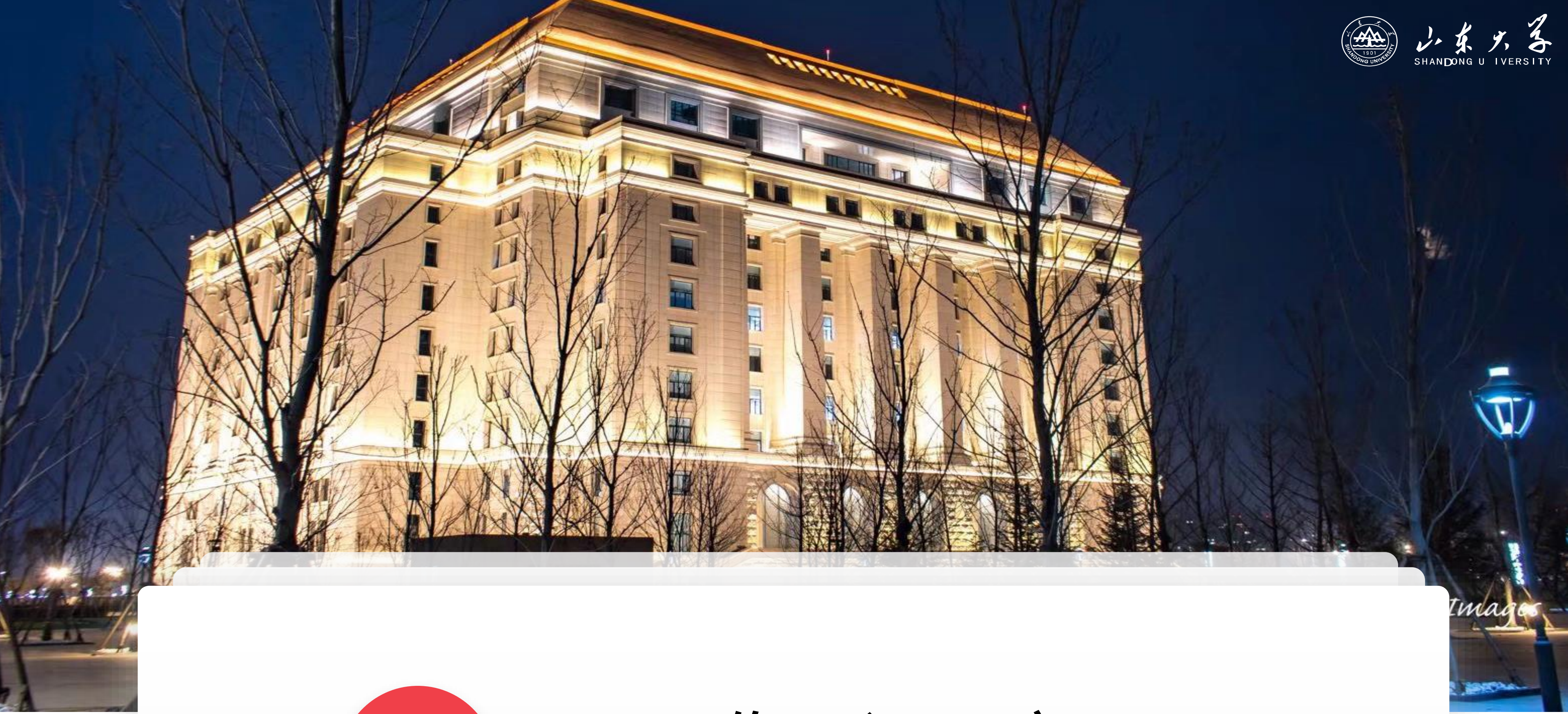
    从 R 中删除与区间 i 不相容的所有区间

Return A



区间调度问题

- 区间调度问题 —— 贪心算法
  - 选择 —— 最早的完成时间 F_i
  - 从左向右选，一开始的范围是(-∞，+∞），每选择一个区间后就变成了 [F_i,+∞），让这个范围尽可能越大越好，所以每次要选结束时间最早的，这样选完之后剩下的范围最大



5 工 作 调 度

Greedy Algorithm Proof

工作调度

- 问题引入
  - 约翰的工作日从 0 时刻开始，有 1e9 个单位时间。在任一单位时间，他都可以选择编号 1 到 n(1,1e5) 工作中的任意一项工作来完成。工作 i 的截止时间是 D_i(1,1e9)，完成后获利是 P_i(1,1e9)。在给定的工作利润和截止时间下，求约翰能够获得的利润最大为多少。

工作调度

做法：

- 先假设每一项工作都做，将各项工作按截止时间排序后入队；

- 在判断第 i 项工作做与不做时，若其截至时间符合条件，则将其与队中报酬最小的元素比较，若第 i 项工作报酬较高（后悔），则 ans += a[i].p - q.top()。
  用优先队列（小根堆）来维护队首元素最小。

6 国王游戏

Example One

国王游戏

- 问题引入
  - 恰逢 H 国国庆，国王邀请 n 位大臣来玩一个有奖游戏。首先，他让每个大臣在左、右手上面分别写下一个整数，国王自己也在左、右手上各写一个整数。然后，让这 n 位大臣排成一排，国王站在队伍的最前面。排好队后，所有的大臣都会获得国王奖赏的若干金币，每位大臣获得的金币数分别是：排在该大臣前面的所有人的左手上的数的乘积除以他自己右手上的数，然后向下取整得到的结果。 国王不希望某一个大臣获得特别多的奖赏，所以他想请你帮他重新安排一下队伍的顺序，使得获得奖赏最多的大臣，所获奖赏尽可能的少。注意，国王的位置始终在队伍的最前面。
  -

$$1 \le n \le 1,000, 0 < a, b < 10000$$

国王游戏

- 解法
  - 相邻的两个位置交换不会影响其他人的答案，只会改变交换位置的两个人
  - 设排序后第 i 个大臣左右手上的数分别为 a_i, b_i 。考虑通过邻项交换法推导贪心策略。
  - 用 s 表示第 i 个大臣前面所有人的 左手数字 的乘积，那么第 i 个大臣得到的奖赏就是 s/b_i，第 i+1 个大臣得到的奖赏就是 $s*a_i/b_{i+1}$。
  - 如果我们交换第 i 个大臣与第 i+1 个大臣，那么此时的第 i 个大臣得到的奖赏就是 $s*a_{i+1}/b_i$ ，第 i+1 个大臣得到的奖励就是 s/b_{i+1}。

国王游戏

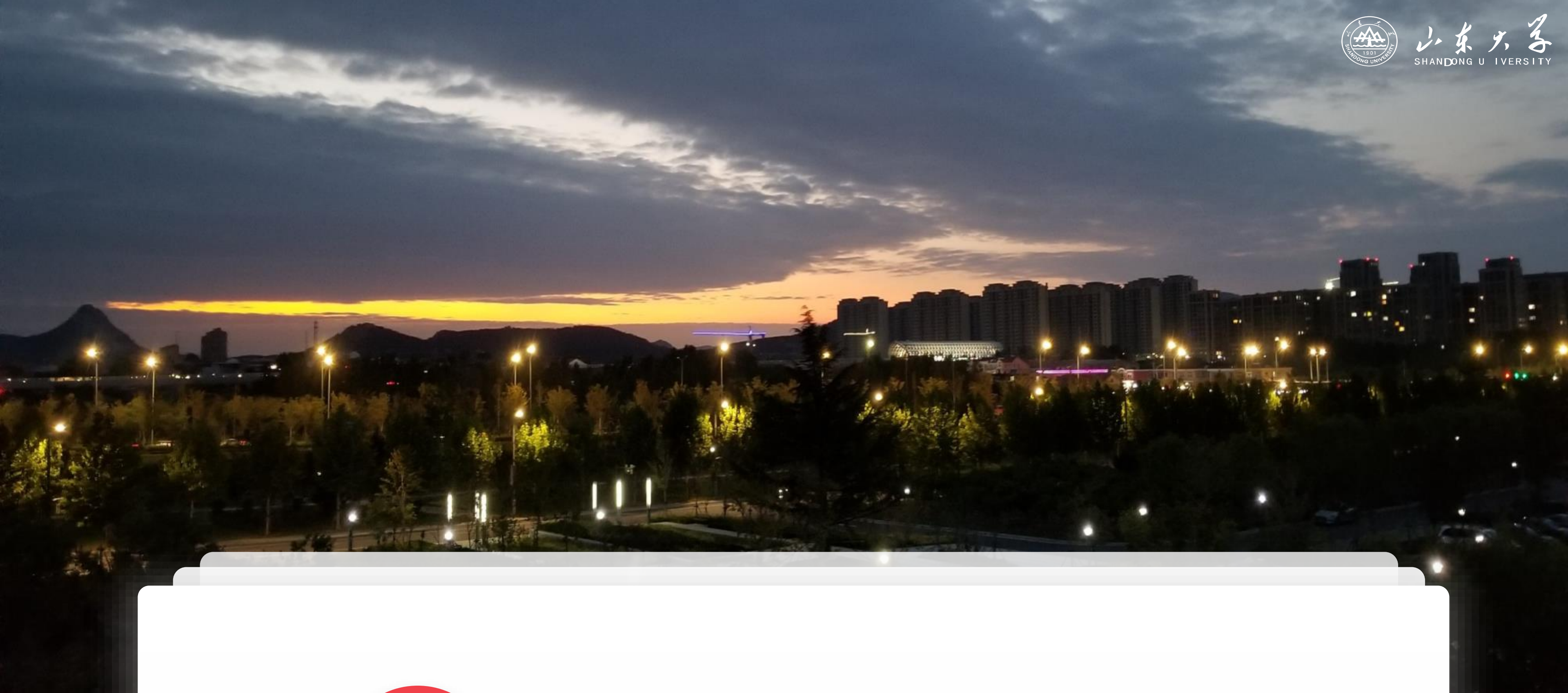
- 解法
  - 如果交换前更优当且仅当

$$max(\frac{s}{b_i}, \frac{s*a_i}{b_{i+1}}) < max(\frac{s}{b_{i+1}}, \frac{s*a_{i+1}}{b_i})$$

  - 提取出相同的 s 并同时都乘上b_i*b_{i+1}

$$max(b_{i+1}, a_i * b_i) < max(b_i, a_{i+1} * b_{i+1})$$

```
struct uv {
  int a, b;
  bool operator<(const uv &x) const {
    return max(x.b, a * b) < max(b, x.a * x.b);
  }
};
```



4 整 数 二 分

Integer Binary

整数二分

- 任务1
  - 设计一个函数$find(x)$，能够在给定的一组升序数列A中，找到第一个$\geq x$的位置

$$A = \{3, 4, 4, 4, 6\}$$

$$find(4) = 2$$

$$find(5) = 5$$

整数二分

1	2	3	4	5
3	**4**	**4**	**4**	**6**

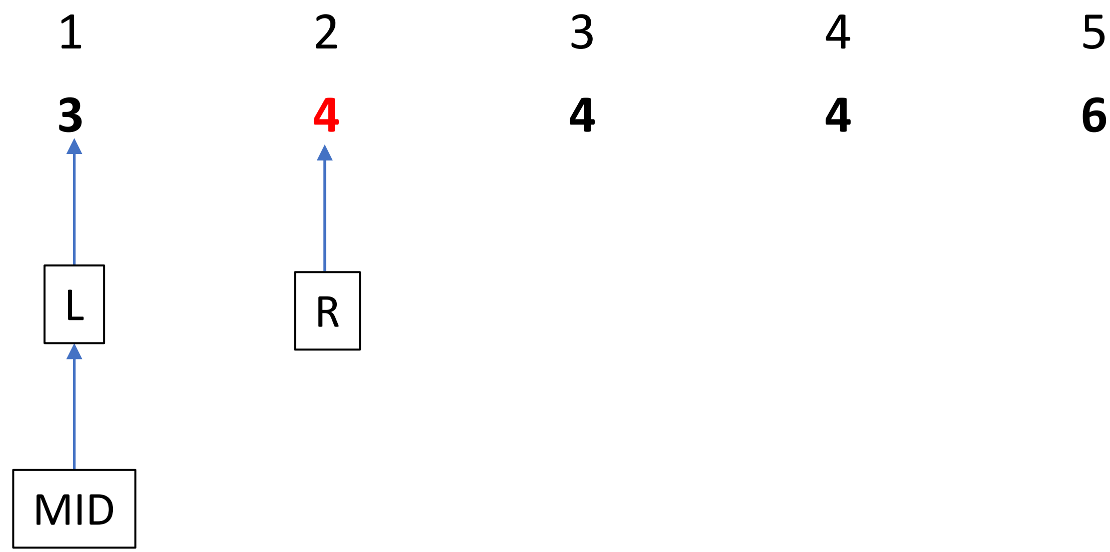
整数二分

1	2	3	4	5
3	**4**	**4**	**4**	**6**
L		MID		R

整数二分

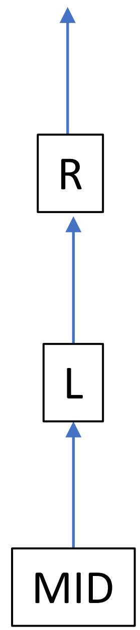
1	2	3	4	5
3	**4**	**4**	**4**	**6**
L	MID	R		

整数二分



整数二分

1	2	3	4	5
3	**4**	**4**	**4**	**6**



整数二分

代码实现

```
int find(int k){
    int l=1,r=n,mid;
    while(l<r){ //查找大于等于k的第一个位置
        mid=(l+r)>>1;
        if(a[mid]≥k) r=mid;
        else l=mid+1;
    }
    return l;
}
```

整数二分

- 任务2
  - 设计一个函数$find(x)$，能够在给定的一组升序数列A中，找到x最后一次出现的位置。如果x不存在输出-1
  - 留给同学们思考~

$$A = \{3, 4, 4, 4, 6\}$$

$$find(4) = 4$$

$$find(5) = -1$$

整数二分

- 分治算法的基本思想是将一个规模为N的大问题分解为K个规模较小的问题（一般来说是2个），这些子问题相互独立并且与原问题的性质相同，求解的过程一致。然后分治算法就是合并子问题的解求得原问题的解。

- 二分法属于分治算法，普遍意义下的二分法为二分查找法

整数二分

- 在一个单调有序(递增或递减) 的区间[a_1,a_n] 中查找元素x，每次将区间分为左右长度相等的两部分，判断解在哪个区间中并调整区间上下界，不断重复直至找到相应的解。
- 时间复杂度 $O(logn)$
- 优于顺序查找 $O(n)$

作用：

1. 查找元素（位置或值）
2. 求满足条件的最值（最大值/最小值）

5 浮点二分

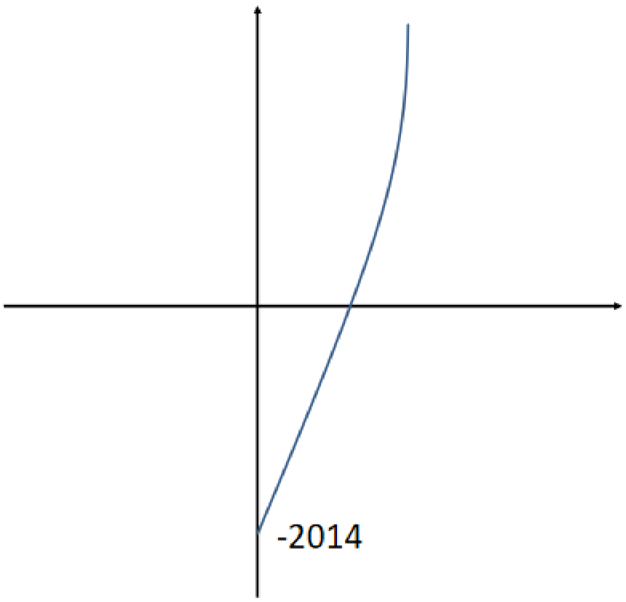
Real Binary

浮点二分

- 1.求方程：
- $x^6 + x - 2014$
- 在$(0, +\infty)$ 的一个解。
- 小数点后精确5位。

浮点二分

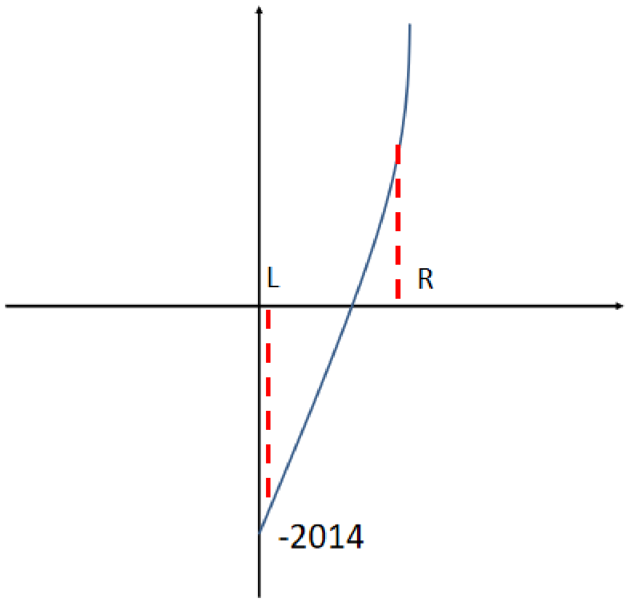
- $f(x) = x^6 + x - 2014$ 在$(0,+\infty)$单调递增



浮点二分

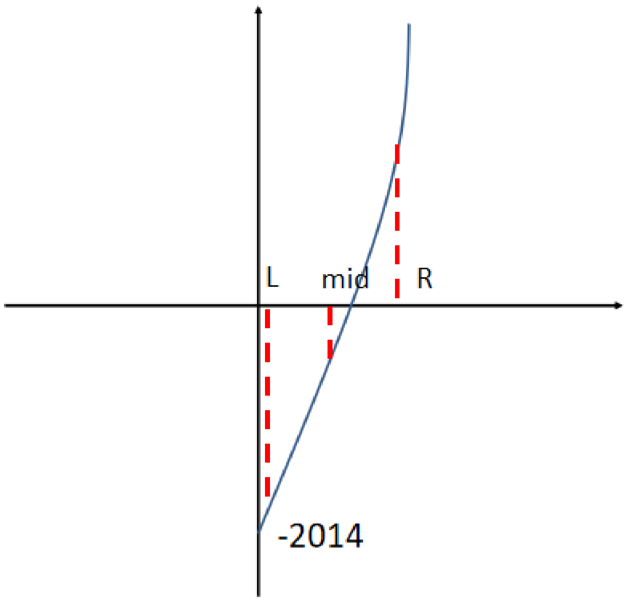
- $f(x) = x^6 + x - 2014$ 在$(0,+\infty)$单调递增

- 估算：
- L = 1.0;   //f(1.0)<0
- R = 10.0;   //f(10.0)>0
- f(L) * f(R) < 0
- 解的区间：[1.0, 10.0]



浮点二分

- L = 1.0;
- R = 10.0;
- mid = (L + R) / 2;
- if (f(mid) * f(R) <= 0)
-   答案在[mid,R];
- else
-   答案在[L,mid];



浮点二分

代码实现1

eps用来控制精度误差，题目要求答案保留小数点后5位，也就是10^{-5}

```
const double eps = 1e-5;
double find() {
    double l = 1, r = 10;
    while(abs(r - l) > eps) {
        double mid = (l + r) / 2;
        if(f(mid) * f(r) <= 0) {
            l = mid;
        } else {
            r = mid;
        }
    }
    return l;
}
```

浮点二分

代码实现2

每一次迭代区间长度减小一半，当区间长度小于10^{-5}时即可停止，需要迭代多少次?

区间长度从10减少到10^{-5}，变化幅度为10^{6}　$\log(10^{6}) \sim 20$

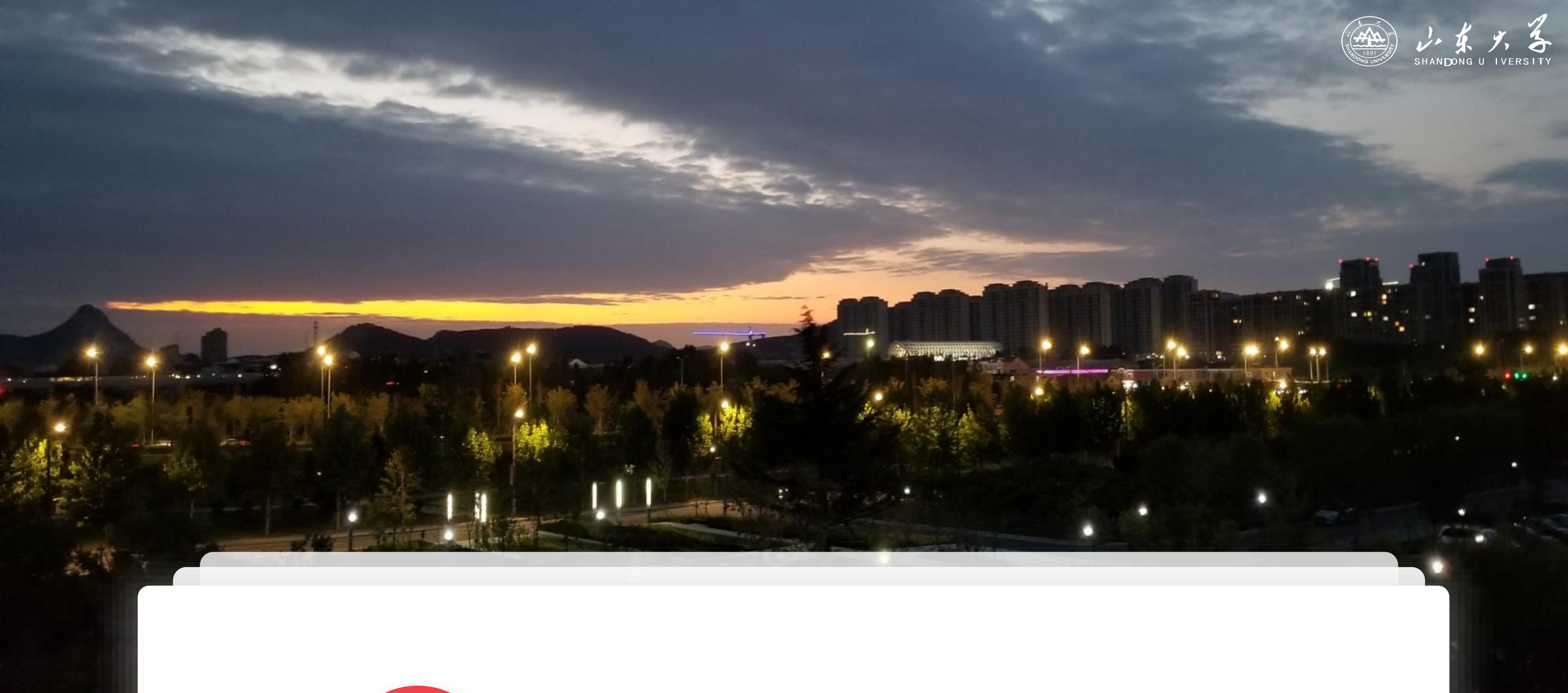
浮点二分

代码实现2

每一次迭代区间长度减小一半，当区间长度小于10^{-5}时即可停止，需要迭代多少次?

区间长度从10减少到10^{-5}，变化幅度为10^{6} $\log(10^{6})\sim 20$

```
double find() {
    double l = 1, r = 10;
    for(int i = 1; i <= 20; ++i) {
        double mid = (l + r) / 2;
        if(f(mid) * f(r) <= 0) {
            l = mid;
        } else {
            r = mid;
        }
    }
    return l;
}
```



6 二 分 答 案

B i s e c t i o n

二分答案

解题的时候往往会考虑枚举答案然后检验枚举的值是否正确。

若满足单调性，则满足使用二分法的条件。

把这里的枚举换成二分，就变成了“二分答案”。

例题

- 小熊有一个W(width)×L(length)×D(depth)大小的空游泳池，他想知道往泳池里放V m^3水和n个重量为w_i和半径为r_i的小球之后的水的高度。

- The first line contains five integers n, W, L, D, V $(1 \leq n \leq 10^3, 2 \leq W, L \leq 10^3, D \leq 5, V \leq W \cdot L \cdot D)$.
- The i^{th} line of the n following lines contains two positive real numbers r_i and w_i indicating the specific weight and radius of the i^{th} ball $(0 < w_i, r_i \leq 2)$. The real values have at most two digits after the decimal point.

例题

如果直接算，非常困难，因为放一个球之后，水位的变化也会影响之前放入的球，不如，二分最终的水位高度，这样每个球的贡献就有了

课后作业

- 题目链接会放到群里

感谢收听

Thank You For Your Listening