

## 07-03-Algorithm

Created on 20220605.

Last modified on 2024 年 8 月 9 日.



# 目录



# Chapter 1 计算模型合集

各种高效实用的计算模型



# Chapter 2 思想

## 2.1 递推

### 2.1.1 逆推

执果索因。

**Question 2.1.** 车从  $A$  到  $B$  共  $akm$ ，油耗  $bL$  每  $km$ ，载油容量  $cL$ ，问需要沿途如何建立储油点，耗油最少到  $B$ ？

**S1 solution 1**

从  $B$  倒推，插入的首个储油点  $P_0$ ， $P_0$  到  $B$  走 1 趟，需要存最少  $cL$ ，距离  $c/bkm$ ； $P_1$ ， $P_1$  到  $P_0$  走 1+2 趟，需要存最少  $c+cL$ ，距离  $c/(3b)km$ ；

$$\min k, \text{ with } \frac{c}{b} \sum \frac{1}{2i+1} > a, k \in \{\mathbb{Z}^+, 0\} \text{ and 油耗是 } a \times (i+1) + b \times (2i+1) \times (a - \sum \frac{1}{2i-1})$$

### 2.1.2 正推

## 2.2 贪心

局部最优是全局最优。

matroid，矩阵胚， $M = [S, I]$ ，如  $S$  是  $n$  个  $m$  维的行向量， $I$  是  $S$  中挑选的所有的那些线性无关的  $r$  个  $m$  维的行向量。给  $m$  维附加上权值维度信息，则贪心即是在  $I$  中找到最优的一组行向量，怎么找呢，每次收集最优的 1 个行向量即可。为什么每次收集最优的 1 个行向量即可呢？马上证明。

**Question 2.2.** 给定  $a$  是  $n$  位数，删掉  $s$  位后，余下的数最小。如  $178543$  删掉 4 位，最小是  $13$

**S1 solution 1**

从左向右遍历  $a$  各位  $s$  遍，每次做的处理：有递减区间  $A$ ，删除区间  $A$  的首；否则删除最后一位。继续下次遍历。

## 2.3 递归

## 2.4 分治

## 2.5 枚举

## 2.6 模拟



# Chapter 3 数据结构

## 3.1 AAA

### 3.1.1 数组

$a[i] = a + i * \text{len}$  ; // i from 0;  $a[i][j]$  按行存:  $a + (i * \text{len} + j) * \text{len}$  ;  $a[i][j]$  按列存:  $a + (j * \text{len} + i) * \text{len}$  ;

5\*5 的二维数组 a, 各元素 2 字节,  $a[2][3]$  行有限存, 地址?  $2*5+3 = 13$ ,  $13*2 = 26$ ,  $a+26$ ;

### 3.1.2 线性表、链表

### 3.1.3 栈

### 3.1.4 队列

优先队列

### 3.1.5 哈希表

哈希函数

哈希函数:  $y = H(x)$ , 输出长度不变; 相同输入每次得到相同输出; 输入差距小也会导致输出差距大, 输入差距大也可能导致输出相同。x 求 y 容易, y 求 x 困难。

MD5, message digest algorithm 5 SHA-1, SHA-2, source hash algorithm. MD5, SHA-1 存在安全隐患。

### 3.1.6 堆

上浮和下沉, 用于实现 priority queues

### 3.1.7 树

#### 二叉树

二叉树遍历: 前序、后续、中序。反向构造二叉树: 利用前 + 中, 或后 + 中遍历结果, 推出树的结构。只利用前 + 后不行。

树转二叉树: 第一个孩子在左, 兄弟都是右。

查找二叉树: 左 < 根 < 右 1) 左子树的值 < 根的值 < 右子树的值; 2) 一直向左达到最小值, 一直向右达到最大值; 3) 增加节点: 从根开始, 向末端方向, 插入值更小就左转, 否则右转, 到达末端增加一个叶子节点; 4) 删除节点 A: A 的左子树的最大节点替代删除的 A 的位置; 5) 扩展: 平衡二叉查找树; B 树 (m 个节点的形状平衡)。

最优二叉树、哈夫曼树: 带权路径长度最小。1,2,8,4 构造哈夫曼树: step1: 1,2->3; 3,8,4; step2: 3,4->7;7,8 so: 15 7 8 3 4 1 2 权值:  $1*3+2*3+4*2+8*1=25$

线索二叉树: 前序、后续、中序, 列举各元素后, 叶子 LR 指针指向前后元素。

平衡二叉树: 任意结点左右子树深度差不大于 1。平衡度 = 左子树深度-右子树深度。

### 3.1.8 图

有向图, 无向图。完全图。

存储: 邻接矩阵。n 个点,  $n*n$ 。i 到 j 有邻接边,  $R_{ij}=1$ , 否则为 0。邻接表, V1 --> [2,6,--] --> [4,1,--] --> [6,50,1] // V1 到 2 号结点距离 6, 到 4 号结点距离 1, 到 6 号结点距离 50

【遍历】深度优先, 广度优先。

【拓扑排序, AOV 网络】有向边表示活动之间开始的先后关系。

【图的最小生成树, 普里姆算法】留下的权值最小。树没有环路, n 个节点的树边最多 n-1 个。染色红, 逐个收集最短的一个元素进来。注意过程中不能形成环。

【图的最小生成树, 克鲁斯卡尔算法】从最短的边开始选边。

# Chapter 4 随机算法

## 4.1 随机数生成

### 4.1.1 Mersenne Twister

梅森旋转 (Mersenne Twister, MT) 算法, 常用是伪随机数生成算法。算法描述如算法??所示。

---

**Algorithm 1:** Mersenne Twister

---

**Input:** the index is noted as  $x_{in}$ , the seed number is noted as seed  
**Output:** random number  $x_{out}$

```
1 Initialization:  $[w, n, m, r], a, f, (u, d), (s, b), (t, c), l, MT_0 \leftarrow \text{seed};$   
2 for  $i \leftarrow 1$  to  $n - 1$  do  
3    $MT_i \leftarrow f \cdot \{MT_{i-1} \oplus [MT_{i-1} \gg (w - 2)] + i\}$   
4 end  
5 for  $i \leftarrow 0$  to  $n - 2$  do  
6    $M_c \leftarrow$  the composition of the highest  $w - r$  bits of  $MT_i$  and the lowest  $r$  bits of  
    $MT_{i+1};$   
7    $M_c \leftarrow M_c \gg 1;$   
8   if the lowest bit of  $M_c$  is 1 then  
9      $M_c \leftarrow M_c \oplus a$   
10  end  
11   $MT_i \leftarrow MT_{i+m} \oplus M_c$   
12 end  
13  $x \leftarrow MT_{x_{in}};$   
14  $x_{out} \leftarrow x \oplus [(x \gg u) \& d];$   
15  $x_{out} \leftarrow x \oplus [(x \ll s) \& b];$   
16  $x_{out} \leftarrow x \oplus [(x \ll t) \& c];$   
17  $x_{out} \leftarrow x \oplus (x \gg l);$   
18 return  $x_{out};$ 
```

---



# Chapter 5 算法合集

一般难解问题的高效实用算法

有穷，确定，有效。

【复杂度】时间，空间

时间复杂度:  $1, \log_2 n, n, n \log_2 n, n^2, n^3, \dots, 2^n$

## 5.1 计算几何

## 5.2 分布式算法

## 5.3 并行算法

## 5.4 查找

### 5.4.1 顺序查找、线性查找

平均查找长度:  $\frac{n+1}{2}$  time,  $O(N)$

### 5.4.2 Binary Search 二分查找

有序排列。对于有序数组，每次甩掉一半可能区间。比较次数最多  $\lfloor \log_2 n \rfloor + 1$  time,  $O(\log_2 n)$

算法描述如算法??所示。

### 5.4.3 散列表查找

例如，存储空间 10， $p=5$ ，散列函数  $h = key \% p$ ，存储 3,8,12,17,9: 线性探测: 3,4, 2, 5,6 冲突解决: 线性探测，伪随机数。

**Algorithm 2:** search-Binary-1

---

**Input:** ordered range set  $S = [l, r)$ , the search number  $t$   
**Output:**  $out$ . if  $t \in S$ , the index of  $t$ , index starts with 0. if  $t \notin S$ ,  $out = -1$

```

1 Initialization:  $out = -1$ ;
2 while  $l < r$  do
3    $m = \lfloor \frac{r+l}{2} \rfloor$ ;
4   if  $t < m$  then
5      $r = m$ ;
6   end
7   else if  $t > m$  then
8      $l = m + 1$ ;
9   end
10  else
11     $out = \text{index of } m \in S$ ;
12    return;
13  end
14 end
15 return  $out$ ;

```

---

## 5.5 排序

稳定、不稳定。【一样的数，保持原顺序，叫稳定】

### 5.5.1 插入式: 直接插入

插入:  $O(n^2)$ , 认为第 1 个已排序, 剩余的依次插入到合适位置。

新的一个与已经排好的比, 插入到位置

### 5.5.2 插入式: 希尔

数据少时插入排序效率可以。

例如 10 个元素, 先  $d=n/2=5$ , 隔 5 个一组, 插入排序;  $d = d/2 = 2$ , 取奇数是 3; 隔 3 个一组, 插入;  $d = d/2 = 1$ , 全体插入排序。

### 5.5.3 选择式: 直接选择

选择:  $O(n^2)$ , 每次从剩余数组中挑最小的。

每次选剩余最小的。

### 5.5.4 选择式: 堆排序

完全二叉树。堆:  $O(n \log n)$ , 构造堆, 不断取根-维护堆结构。小顶堆:  $k_i \leq k_{2i}, k_i \leq k_{2i+1}$   
 大顶堆:  $k_i \geq k_{2i}, k_i \geq k_{2i+1}$  所有孩子都更小

**Algorithm 3:** search-Binary-2

---

**Input:** ordered range set  $S = [l, r]$ , the search number  $t$   
**Output:**  $out$ . if  $t \in S$ , the index of  $t$ , index starts with 0. if  $t \notin S$ ,  $out = -1$

```

1 Initialization:  $out = -1$ ;
2 while  $l \leq r$  do
3    $m = \lfloor \frac{r+l}{2} \rfloor$ ;
4   if  $t < m$  then
5      $r = m - 1$ ;
6   end
7   else if  $t > m$  then
8      $l = m + 1$ ;
9   end
10  else
11     $out = \text{index of } m \in S$ ;
12    return;
13  end
14 end
15 return  $out$ ;

```

---

从小到大排列: 建小顶堆-》取顶-》建小顶堆-》。。。。

例如构造大顶堆: step1, 数组顺序构造完全二叉树。step2, 最后一个非叶子节点, 与其 2 个孩子调整为大顶堆; 倒数第 2 个非叶子节点, 依次调整。如果有子树, 要调整后继续调整子树。

### 5.5.5 交换式: 冒泡

冒泡:  $O(n^2)$ , 认为数组是从地板到天花板, 每轮都从地板开始冒泡, 每轮导致天花板降低; 天地相接或某一轮的所有冒泡没有产生相邻的交换, 认为排好序。

### 5.5.6 交换式: 快速

快速:  $O(n \log n)$ , 递归的分治法每次操作 F 是把当前处理的区间划分为 3 个部分: [小于基准的数区间] 基准数 [大于基准的数区间]。对左右 2 个区间递归执行操作 F。

### 5.5.7 归并排序

归并:  $O(n \log n)$ , 递归的分治法 1) 自顶而下: 不断细分, 然后归并。归并要保证两帧有序。

### 5.5.8 基数排序

## 5.6 图的搜索

### 5.6.1 广度优先

广度优先: FIFO (先入先出), 用队列。

### 5.6.2 深度优先

深度优先: LIFO (后入先出), 用栈。

### 5.6.3 Bellman-Ford

Bellman-Ford: 无向图中最短路径问题。从 A 节点到 B 节点, 节点间度量为正。设置初始权重起点为 0, 其余点无穷大。广度优先, 更新各个节点的权重值, 有更小的时更新权重值。

### 5.6.4 Dijkstra

Dijkstra: 无向图中最短路径问题。从 A 节点到 B 节点, 节点间度量为正。设置初始权重起点为 0, 其余点无穷大。计算与 A 连接是边中最短的节点 K1, 然后计算与 K1 连接是边中最短的节点 K2, 持续下去直到 B。

### 5.6.5 A-star

A-star: Dijkstra 并没有一个指向性保证一次性走到终点。增加一种引导, 如当前点与终点的估计距离, 引导每次对最短节点 K1 的选择。这样的算法称为“启发式算法”。

## 5.7 安全算法

A 向 B 数据传输 4 个问题: 1) 窃听: C 听到了; 【加密】2) 假冒: A 或 B 是假的; 【消息认证、数字签名】3) 篡改: B 收到的是 C 修改后的 A 发送的消息; 【消息认证、数字签名】4) 事后否认: A 事后不承认消息了。【数字签名】数字签名中, 为了确认公开密钥的制作者, 使用“数字证书”技术。

### 5.7.1 利用密钥加密

#### 共享密钥加密

【共享密钥加密。对称加密】DES, AES, 凯撒密码, 动态口令。1) 加密和解密用相同的密钥。密钥从 A 发送到 B 的过程中, 也可能被窃听。密钥分配问题。2) 密钥分配问题的解决方案: 密钥交换协议, 公开密钥加密。

#### 公开密钥加密

RSA 算法、椭圆曲线加密算法。1) B 生成公钥 P 和私钥 S, B 把 P 发给 A, A 用 P 加密后把密文发给 B, B 用 S 解密。2) 假设 n 个人互相传输, 需要  $n(n-1)/2$  对密钥。改进是想 B 对所有的人都保持 1 个 P 和 1 个 S。3) 安全问题: 【中间人攻击】在 B 把 P 发给 A 时, C 截获



P, 把 Q 发给 A, A 用 Q 加密后的密文再次被 C 截获, C 解密后修改, 再用 P 加密传给 B。问题是由于 A 不知道收到的密钥是否来自 B。用数字证书解决。4) 加密解密时间长, 不适合持续发送小数据的情形。用混合加密解决。5) RSA 算法中利用 Miller-Rabin 质数测试。

### 混合加密

SSL/TLS 协议。数据用共享密钥加密, 发送的密钥信息用公开密钥加密。

### Diffie-Hellman 密钥交换

1) 构造密钥合成算法  $F = [P, S]$ , 具有特点: 1) 可合成不可分解; 合成后可继续合成; 合成结果与合成顺序无关。2) A 和 B 公开密钥 P, A 准备 SA, A 传输  $[P, SA]$  给 B; B 准备 SB, A 传输  $[P, SB]$  给 A; A 和 B 各自组合出  $[P, SA, SB]$  用于加密和解密。窃听者无法组合出。2) A 和 B 公开大质数 P 和另外一个数 G; A 选一个数 x, A 发送  $(G^x) \bmod P$  给 B; B 选一个数 y, B 发送  $(G^y) \bmod P$  给 A; A 和 B 都用密钥  $(G^{xy}) \bmod P$ 。根据质数 P、生成元 G、 $(G^x) \bmod P$ , 求 x 的问题, 称为离散对数问题, 该问题至今没找到解法。

#### 5.7.2 消息认证码

认证、检测篡改 A 把密钥 P 安全发给 B; A 用密文和密钥 P 生成消息认证码如 ab12, 称为 MAC(Message authentication code); B 收到密文和 MAC 后, 用密文和密钥生成一份 MAC 和 A 发来的比较是否一样。MAC 算法: HMAC, OMAC, CMAC 缺点: 无法保证密文是 A 生成的还是 B 生成的。问题原因是两方都有相同的密钥, 不能确定 MAC 是谁生成的。解决方案: 数字签名。

#### 5.7.3 数字签名

希望: A 的签名发送给 B, B 可以验证签名, B 不能生成签名。1) 公开密钥加密是 P 加密 S 解密; 数字签名是 S 加密 P 解密。2) A 准备发签名, A 准备好 P 和 S; A 用 S 加密得到签名, 公开 P。能够用 A 发布的 P 解密的, 一定是 A 的 S 加密生成的。3) 求消息的哈希值 X, 对 X 加密得到签名。4) 问题: 需要知道公钥 P 的制作者, 防止 C 用自己的公钥冒充 A 的。解决方法: 数字证书。

#### 5.7.4 数字证书

A 和 B 之间的事, 找一个双方承认的中间人。A 把要公开的密钥 PA 和自己的个人信息提交给认证中心 (CA, Certification Authority); CA 确认后利用 CA 的私钥将 PA 和个人信息生成签名作为 A 的证书。B 收到证书后, 利用 CA 的公钥 PC 检测证书。1) 问题: 检测证书的公钥 PC 是来自 CA 的吗? CA 的 PC 是以数字证书的形式交付的, 有更高级别的 CA 署名。2) 根认证中

心,其正当性由自身证明,如大型企业。3) 网站的证书称为“服务器证书”,与域名信息对应。可确认域名和存储网站本身的服务器由同一个组织管理。4) PKI, public key infrastructure, 公钥基础设施。

## 5.8 聚类算法

### 5.8.1 k-means

随机选择凝聚中心,得到  $n$  个集合;利用集合重心作为新的凝聚中心,计算新一轮的簇。重复下去得到最终的  $n$  个集合。

### 5.8.2 层次聚类

初始时每个对象为 1 类;每次将最近的 2 类合并为 1 类,持续下去。

## 5.9 其他算法

### 5.9.1 欧几里得算法 (又称辗转相除法)

1)  $A = k_1 * \gcd(A, B)$ ,  $B = k_2 * \gcd(A, B)$ ,  $A$  和  $B$  看做相同刻度的不同数量的尺子,不断把长的重新赋值为长的减去短的,直到最后剩下长度之比为 1:2,得到了刻度。2) 令  $L_0 > R_0$ ;  $L_1 = R_0$ ,  $R_1 = L_0 \bmod R_0$ ; 一直到  $L_k, R_k, R_k = 0, L_k = \gcd(L_0, R_0)$

### 5.9.2 质数判断

1) 根据定义枚举: 计算  $A$  的平方根  $n$ ,  $i: [2, n]$ ,  $r[i] = A \bmod i$ ;  $r[i]$  中有 0 表示有公因数,即不是质数。2) 费马测试: 对于质数  $p$ , 任意小于  $p$  的数  $c$ , 有  $(c^p) \bmod(p) = c$ 。测试  $A$ , 随机找几个小于  $A$  的数, 判断通过费马测试, 大概率认为是质数。3) 存在满足费马测试的合数, 称为 Carmichael Numbers, 绝对伪质数, 如 561。4) AKS 算法, 多项式时间内进行质数测试。

### 5.9.3 PageRank

1) 利用网页间的链接关系判断网页的价值。2)  $A$  链接指向  $x$  个网页,  $x$  个网页评分  $A$  的权重;  $A$  被  $y$  个网页指向,  $A$  的评分等于来的各个网页的权重之和; 为了解决循环链接, 引入随机游走, 即有  $a$  的概率跳到其他的节点, 有  $1-a$  的概率沿着链接关系走。

### 5.9.4 汉诺塔问题

递归 1) 移动方法  $F$  满足:  $F(n) = F(F(n-1))$

### 5.9.5 杨辉三角



# Chapter 6 算法大全

## 6.1 数组

### 6.1.1 Remove Element 移除元素

给你一个数组 `nums` 和一个值 `val`，你需要 原地 移除所有数值等于 `val` 的元素，并返回移除后数组的新长度。

不要使用额外的数组空间，你必须仅使用  $O(1)$  额外空间并\*\*原地\*\*修改输入数组。

元素的顺序可以改变。你不需要考虑数组中超出新长度后面的元素。

示例 1:

给定 `nums = [3,2,2,3]`, `val = 3`,

函数应该返回新的长度 2, 并且 `nums` 中的前两个元素均为 2。

你不需要考虑数组中超出新长度后面的元素。

---

#### Algorithm 4: Remove Element-1

---

```
1 Brif: array; swap;
2 Initialization:  $l = 0, r = size(S)$ ;
3 Notation 1: after the algorithm,  $r$  is the index of the last element that not equal to  $t$ ;
4 Notation 2: swap(a,a) does nothing ;
   Input: set  $S = [l, r), l = S[0]$ , the removed number  $t$ .
   Output: out. the length of new set.
5 while  $l \leq r$  do
6   if  $S[l] = t$  then
7     | swap( $S[l], S[r]$ );
8     |  $r \leftarrow r - 1$ ;
9   end
10  else
11    |  $l \leftarrow l + 1$ ;
12  end
13 end
14  $out \leftarrow r + 1$ ;
15 return out;
```

---

**Algorithm 5:** Remove Element-2

---

```

1 Brif: array; fast slow pointer;
2 Notation 1: not change the order;
   Input: set  $S = [l, r)$ ,  $l = S[0]$ , the removed number  $t$ .
   Output:  $out$ . the length of new set.
3 Initialization:  $p_{slow} = -1$ ;
4 for  $p_{fast} : [0, size(S))$  do
5   if  $S[p_{fast}] \neq t$  then
6      $p_{slow} \leftarrow p_{slow} + 1$ ;
7      $S[p_{slow}] = S[p_{fast}]$ ;
8   end
9 end
10  $out \leftarrow p_{slow} + 1$ ;
11 return  $out$ ;
```

---

**6.1.2 有序数组的平方**

给你一个按 非递减顺序 排序的整数数组 `nums`，返回 每个数字的平方 组成的新数组，要求也按 非递减顺序 排序。

示例 1:

\* 输入: `nums = [-4,-1,0,3,10]`

\* 输出: `[0,1,9,16,100]`

\* 解释: 平方后, 数组变为 `[16,1,0,9,100]`, 排序后, 数组变为 `[0,1,9,16,100]`

**Algorithm 6:** Suqre of ordered sequence-1

---

```

1 Brif: array; two pointers;
   Input: ordered set  $S = [l, r)$ ,  $l = S[0]$ .
   Output:  $T$ . the squire set.
2 Initialization:  $p_l = 0$ ,  $p_r = size(S)$ , out set  $T$  with the same size of  $S$ , all elements set to
    $0$ ,  $p_t = size(T)$ ;
3 for  $i : [0, size(S))$  do
4    $S[i] \leftarrow S[i]^2$ 
5 end
6 while  $p_l \leq p_r$  do
7   if  $S[p_l] \leq S[p_r]$  then
8      $T[p_t] = S[p_r]$ ;
9      $p_t \leftarrow p_t + 1$ ;
10     $p_r \leftarrow p_r - 1$ ;
11  end
12  else
13     $T[p_t] = S[p_l]$ ;
14     $p_t \leftarrow p_t + 1$ ;
15     $p_l \leftarrow p_l + 1$ ;
16  end
17 end
18 return  $T$ ;
```

---

### 6.1.3 长度最小的子数组

给定一个含有  $n$  个正整数的数组和一个正整数  $s$ ，找出该数组中满足其和  $s$  的长度最小的连续子数组，并返回其长度。如果不存在符合条件的子数组，返回 0。

示例：

\* 输入:  $s = 7$ ,  $nums = [2,3,1,2,4,3]$

\* 输出: 2

\* 解释: 子数组  $[4,3]$  是该条件下的长度最小的子数组。

<https://leetcode.cn/problems/spiral-matrix-ii/submissions/518829536/>

```
class Solution {
public:
    vector<vector<int>> generateMatrix(int n) {
        vector<vector<int>> out(n, vector<int>(n,0));

        int s[2]{0,0};
        int p[2]{0,0};
        int cur[2]{0,0};
        int br = n-1,bb = n-1, bl=0,bt=0;
        int v[2]{1,0};
        bt -=1;

        out[0][0] = 1;
        for(int i=0;i<n*n;++i){

            int index = 1; // 0 ok, -1 error, 1 try

            while(index ==1){

                if((v[0]== 1) &&(v[1]== 0) ){ // go right
                    if(p[0]== br){

                        br -=1;
                        v[0]== 0;v[1] = 1;
                    } else{
                        index = 0;
                    }
                }
                else if((v[0]== 0) &&(v[1]== 1) ){ // go bottom
```

```

        if(p[1]== bb){
            bb -=1;
            v[0]== -1;v[1] = 0;
        } else{
            index = 0;
        }
    }

    else if((v[0]== -1) &&(v[1]== 0) ){ // go left
        if(p[0]< b1){
            index = -1;
        }
        else if(p[0]== b1){
            b1 +=1;
            v[0]== 0;v[1] = -1;
        } else{
            index = 0;
        }
    }

    else if((v[0]== 0) &&(v[1]== -1) ){ // go top
        if(p[1] == bt){
            bt +=1;
            v[0]== 0;v[1] = 1;
        } else{
            index = 0;
        }
    }

    }

    p[0] = p[0]+v[0];
    p[1] = p[1]+v[1];

    out[p[0]][p[1]] = i+1;
    std::cout <<p[0]<< ", " << i+1 << std::endl;

}

return out;

}

};

```

#### 6.1.4 else

---

**Proposition 6.1.** 给定  $\beta$  边边长, 判断三角形类型: $c = \max(a, b, c), T = c^2 - a^2 - b^2$ , 利用  $T$  的符号。



---

**Algorithm 7:** Shortest subarray-1

---

```

1 Brif: array; two pointers; Sliding window;
  Input: ordered set  $S = [S[0], S[size(S)]]$ , thenumbert.
  Output: the shortest subarray that sum greater than t, stats with index k, length is
           len.
2 Initialization:  $p_l = 0, p_r = 0, len = size(S) + 1, k = 0$ , the sum of the subarray  $sum = 0$ ;
3 for  $p_r : [0, size(S))$  do
4    $sum \leftarrow sum + s[p_r]$ ;
5   while  $sum \geq t$  do
6      $sum \leftarrow sum - s[p_l]$ ;
7      $len = \min(len, p_r - p_l + 1)$ ;
8      $k \leftarrow p_l$ ;
9      $p_l \leftarrow p_l + 1$ ;
10  end
11 end
12 return  $k, len$ ;

```

---

**Proposition 6.2.** 小于  $A$  的所有质数:  $a = \lfloor \sqrt{A} \rfloor, i \in \{2, 3, \dots, a\}, Answer = \{i|a\}$ .



# Chapter 7 算法综合案例

面向应用的大尺度难解问题的工程实用算法



## Chapter 8 工程算法集成和相应软件 体系结构



## Chapter 9 工程算法分析和评价体系