07-03-Algorithm

Created on 20220605.

Last modified on 2025 年 4 月 20 日.

目录

Chapter 1　Study

1.1　Progress

1.2　Website

一道题目从 0 开始思考

NOI 级别

Petrozavodsk Camp

知识框架 + 刷题。或者边做边学。

```
 1. OI倾向的OJ:

 a) 洛谷: https://www.luogu.com.cn

 b) BZOJ: https://www.lydsy.com/JudgeOnline

 c) UOJ: http://uoj.ac

 d) Libre OJ: https://loj.ac


CNOI风格, 知识点 ==》 题目。对知识和思维的提高帮助有限。

带权带花树板子


2. 个人比赛平台

a) Codeforces Round: https://codeforces.com

b) Atcoder Round: https://atcoder.jp

c) Topcoder: https://www.topcoder.com

d) Codechef: https://www.codechef.com
```

e) Random OJ Round，如牛客挑战赛（？）

3. 组队比赛平台

a) Codeforces Gym：https://codeforces.com/gyms

b) Opentrains：http://opentrains.snarknews.info

c) Yandex：https://contest.yandex.ru/?lang=en

d) Vjudge：https://vjudge.net

首先是Codeforces Gym，里面比较成套的题目有Andrew Stankevich Contest系列，就是ITMO教练在Petrozavodsk Camp每年出的题。我只能说，历史上大多数备战总决赛的队伍都会选择板刷，但因为Andrew的题也逐渐失去了时效性，所以我持保留意见。另一个就是World Finals的题基本都有了，建议在备战WF的时候做，榜上也会有很多不同时代的著名队伍，可以纵向、横向比较。还有就是有很多零散的各个地区的区域赛，都是出题人自愿放上来的，建议按前文提到的选题方式进行辨识。同时Gym里还包含一些Petrozavodsk Camp和OpenCup的题，搜索对应关键字就好了

Opentrains在很长一段时间里，国内都只有实力较强的学校在使用，其本质是就是一个放着大量往年Petrozavodsk Camp和OpenCup题目，可供VP的OJ，需要找Snark要账号。这个Snark的CF id是snarknews，可以私戳。我感觉他只会给出线WF学校/队伍账号，很迷。那实际上，我觉得也不需要非要做这个OJ里的题，一个是我之前说的，难度不适合，另一个是缺少题解和有效讨论，虽然我们历史上有很多人和队伍都在做着相关的题解记录和讨论工作，但是因为我们是做别人手里的题，我们的整理过于分散且太不系统了，我稍微列几个：

i. ICPC CAMP Wiki by 叉姐：http://icpc.camp

ii. ICPC CAMP Forum by 叉姐：https://forum.icpc.camp

iii. Claris ‘s Blog：https://www.cnblogs.com/clrs97/

所以，不要随便就去做这些题，一来可能旧，二来你可能签完到就懵逼了，也不知道找谁问。

关于Yandex，实际上我也是去年在zimpha学长的安利下才开始认真看contest.yandex上的东西，也是Snark放上去的。我着重提下Cup of Three Quarterfinals，在这里你可以vp历年东欧子赛区的题目，比如Northern Subregional，Moscow Subregional等等，这些题都非常适合练习。

接下来是Vjudge，通过这个壳，你可以拉去各个OJ的题目，而且还能配以相应的榜。通过vjudge，你可以复现区域赛、多校等国内的题目，可以说是备战区域赛的最重要的一个平台。不过，有些OJ可能自己炸了，也可能版本革新后不再支持复现了，就比较可惜，说的就是你们啊bnuoj和zoj。

4. 其他平台

a) Aizu：http://judge.u-aizu.ac.jp/onlinejudge/finder.jsp?course=CGL

b) Sgu: https://codeforces.com/problemsets/acmsguru

c) Project Euler: https://projecteuler.net

太多了，我就不多举了，Aizu的这个，其实我是在DDF的wiki里找到的，当初cls整理几何板子好像就刷了我给的这个网页里的东西。至于sgu，老牌OJ，据说楼教主刷完了整个OJ。至于PE，我认识的所有能用“板刷PE”形容的人都是卡密一样的人物，就算不是卡密，他出的题也是卡密一样的东西，没错说的就是19ecfinal的某个题。

5. 年度活动

a) OpenCup: http://opencup.ru

b) 多校: http://acm.hdu.edu.cn

c) 训练营: 国内如字节跳动和wls

关于OpenCup，我记得叉姐给我的比喻特别准确：她说这东西就是俄罗斯人的多校，只不过整年办而已，题目也来源于世界各地到处白嫖。我觉得，比起做opentrains，我更推荐有条件的情况下去要个opencup账号，周日4:00-9:00跟着做，就算做不来也可以进入相应页面补题。比赛后，都有人会在codeforces发讨论帖，有什么问题也可以在上面问，甚至会有人发题解的sketch，请务必把这个讨论帖mark下来，不然真的很难找。反过来说，opentrains上的一些OpenCup，你牛逼一点也是能反向在cf找到讨论帖的。我觉得这个活动最神奇的是，他根本就不是买题，就是个白嫖的存在，东拿一套camp的题，西要一套区域赛，而且大家都很喜欢送题去给OpenCup，比如前几天的gp of tokyo就是白送的。所以我一直觉得Snark真的很牛逼，真的是白嫖大师，不过这样的活动对全世界来说都是有益的。毕竟能提供给全世界的强队，包括退役传奇队伍和选手过招交流的机会，只此一家。

训练营

Chapter 2 计算模型合集

各种高效实用的计算模型

Chapter 3 思想

3.1 模拟

翻译

3.2 枚举

双指针、滑动窗口

3.3 递推

递推公式。实现方式：loop 或 recrusion

3.3.1 逆推

执果索因。

Question 3.1. 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 *P0*，*P0* 到 B 走 *1* 趟，需要存最少 cL，距离 *c/bkm*；*P1*，*P1* 到 *P0* 走 *1+2* 趟，需要存最少 *c+cL*，距离 *c/(3b)km*；

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

3.3.2 正推

3.4 递归

回溯。一种编程手段。stack 模拟递归。

master 公式

$$T(N) = a \times T(N/b) + O(n^c)$$

when $log_b a \neq c$, we have

$$O(n^{max(log_b a, c)})$$

when $log_b a = c$, we have $n^c \times (logn)$

$$T(N) = a \times T(N/b) + O(n \times logN)$$

we have $O(N \times (logN)^2)$

3.5 分治

3.5.1 归并分治

冯诺依曼发明的。

左半部分 + 右半部分 + 跨左右部分。

归并分治可解决的问题，也可用：线段树、树状数组等解决。

2D 任意 2 点最短距离（算法导论）

3.5.2 整块分治

3.6 搜索

3.7 贪心

3.7.1 key thoughts

k-optional

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。假设最优集是 T，挑选的是 B，证明 B=T。假设不相等，可从 B 中取出一个 x，并加入在 T 中不在 B 中的一个 y，由于 T 最优，所以 $w_x \leqslant w_y$, 但是构造 B 的时候选择了 x 而不是 y，说明 $w_y \leqslant w_x$，所以 $w_y = w_x$，即证。

Question 3.2. *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。

3.8 DP

有限集合的最值问题。

3.8.1 各个模型

选择问题：背包模型

状态表示：前 i 个物品、限制

状态计算

0-1 背包

完全背包

序列问题：最长上升子序列模型

状态压缩 DP

状态机

树形 DP

区间 DP

单调队列优化

斜率优化

背包问题：

a 是 10kg，60 元，单位价值 6；
b 是 20kg，100 元，单位价值 5；
c 是 30kg，120 元，单位价值 4；
背包 50kg，求如何放入总价值最大。

部分背包问题，按单位价值从大到小，(a,b,c)=(10,20,20);

0-1 背包问题，需要考虑有物体 x 和无物体 x 的情况 => 重叠子问题 => 存表、记忆、DP

Chapter 4 算法基本概念

一般难解问题的高效实用算法

有穷，确定，有效。

4.1 复杂度

【复杂度】时间，空间

时间复杂度: $1, log_2 n, n, nlog_2 n, n^2, n^3, ..., n^k, 2^n, k^n, n!$

常数复杂度中，优的顺序：位运算 > 数值 > 寻址 >hash

4.2 均摊

每次翻倍扩容的数组，数据-扩容的时间复杂度对应关系：1-1,2-2,4-4,8-8，例如当 8 的时候，需要经历扩容时间复杂度 1、2、4、8，即 2*8-1，约为 O(N), 均摊到每个元素为 O(1)

Chapter 5 数据结构

见 07-03-04-DataStructure

Chapter 6 半数值算法

6.1 随机数生成

6.1.1 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}

Initialization:$[w, n, m, r], a, f, (u, d), (s, b), (t, c), l,\ MT_0 \leftarrow$ seed;
for $i \leftarrow 1$ **to** $n - 1$ **do**
    $MT_i \leftarrow f \cdot \{MT_{i-1} \oplus [MT_{i-1} >> (w-2)] + i\}$
end
for $i \leftarrow 0$ **to** $n - 2$ **do**
    $M_c \leftarrow$ the commposition of the higest $w - r$ bits of MT_i and the lowest r bits of MT_{i+1};
    $M_c \leftarrow M_c >> 1$;
    if *the lowest bit of* M_c *is* 1 **then**
        $M_c \leftarrow M_c \oplus a$
    end
    $MT_i \leftarrow MT_{i+m} \oplus M_c$
end
$x \leftarrow MT_{x_{in}}$;
$x_{out} \leftarrow x \oplus [(x >> u)\ \&\ d]$;
$x_{out} \leftarrow x \oplus [(x << s)\ \&\ b]$;
$x_{out} \leftarrow x \oplus [(x << t)\ \&\ c]$;
$x_{out} \leftarrow x \oplus (x >> l)$;
return x_{out};

6.2 算数

Chapter 7　排序与查找

7.1　排序

稳定、不稳定。【一样的数，保持原顺序，叫稳定】

7.1.1　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

7.1.2　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。

7.1.3　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。

7.1.4　选择式: 堆排序

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

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。

7.1.5 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。

7.1.6 交换式: 快速

快速: $O(nlogn)$，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。

7.1.7 归并排序

归并: $O(nlogn)$，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。

7.1.8 基数排序

7.2 查找

7.2.1 顺序查找、线性查找

平均查找长度: $\frac{n+1}{2}$ time,O(N)

7.2.2 Binary Search 二分查找

有序数组中查找某数

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。

有序数组中查找大于等于 x 的最左位置

相邻无重复数组中寻找峰值

检查完某个点 P 的峰值状态后，能够一定确定继续查找 P 的某侧。

LeetCode:278,744

7.2.3 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 $t \notin S, out = -1$

Initialization:$out = -1$;
while $l < r$ **do**
  $m = \lfloor \frac{r-l}{2} \rfloor$;
  if $t < m$ **then**
    $r = m$;
  end
  else if $t > m$ **then**
    $l = m + 1$;
  end
  else
    $out =$ index of $m \in S$;
    return;
  end
end
return out;

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 $t \notin S, out = -1$

Initialization:$out = -1$;
while $l \leqslant r$ **do**
  $m = \lfloor \frac{r-l}{2} \rfloor$;
  if $t < m$ **then**
    $r = m - 1$;
  end
  else if $t > m$ **then**
    $l = m + 1$;
  end
  else
    $out =$ index of $m \in S$;
    return;
  end
end
return out;

Chapter 8　组合算法

8.1　组合检索

8.2　递推

Chapter 9　语法算法

9.1　词法扫描

9.2　语法分析

Chapter 10 特定领域算法

10.1 数论

10.2 图论

10.2.1 广度优先

广度优先: FIFO（先入先出)，用队列。

10.2.2 深度优先

深度优先: LIFO（后入先出)，用栈。

10.2.3 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。

10.2.4 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。

10.2.5 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”。

10.3 计算几何

10.4 组合数学

10.5 线性代数

10.6 博弈论

Chapter 11 算法合集-特殊 1

11.1 分布式算法

11.2 并行算法

11.3 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，为了确认公开秘钥的制作者，使用“数字证书”技术。

11.3.1 利用秘钥加密

共享秘钥加密

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

公开秘钥加密

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 对秘钥。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 不知道收到的秘钥是否来自 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。数据用共享秘钥加密，发送的秘钥信息用公开密钥加密。

Diffie-Hellman 密钥交换

1) 构造秘钥合成算法 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 公开秘钥 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 $(G^x)modP$ 给 B；B 选一个数 y，B 发送 $(G^y)modP$ 给 A；A 和 B 都用秘钥 $(G^{xy})modP$。根据质数 P、生成元 G、$(G^x)modP$，求 x 的问题，称为离散对数问题，该问题至今没找到解法。

11.3.2 消息认证码

认证、检测篡改 A 把秘钥 P 安全发给 B；A 用密文和秘钥 P 生成消息认证码如 ab12，称为 MAC(Message authentication code)；B 收到密文和 MAC 后，用密文和秘钥生成一份 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秘钥，不能确定 MAC 是谁生成的。解决方案: 数字签名。

11.3.3 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) 公开秘钥加密是 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。

11.3.4 数字证书

A 和 B 之间的事，找一个双方承认的中间人。A 把要公开的秘钥 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。

11.4 聚类算法

11.4.1 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。

11.4.2 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。

Chapter 12　AI 算法

该部分见 ArtificialIntelligence

Chapter 13 其他算法

13.0.1 欧几里得算法（又称辗转相除法）

1) A = k1*gcd(A, B), B = k2*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 L0 > R0; L1 = R0,R1 = L0 mod R0；一直到 Lk, Rk,Rk = 0,Lk = gcd(L0，R0)

13.0.2 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 mod 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 $(c^p)mod(p) = c$。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。

13.0.3 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。

13.0.4 汉诺塔问题

递归 1) 移动方法 F 满足: F（n) =F（F（n-1))

13.0.5 杨辉三角

Chapter 14 算法大全

14.1 数组

14.1.1 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

Brif: array; swap;
Initialization:$l = 0, r = size(S)$;
Notation 1: after the algorithm, r is the index of the last element that not equal to t;
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.
while $l \leqslant r$ **do**
  if $S[l] = t$ **then**
    $swap(S[l], S[r])$;
    $r \leftarrow r - 1$;
  end
  else
    $l \leftarrow l + 1$;
  end
end
$out \leftarrow r + 1$;
return out;

Algorithm 5: Remove Element-2

Brif: array; fast slow pointer;
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.
Initialization:$p_{slow} = -1$;
for $p_{fast} : [0, size(S))$ **do**
  if $S[p_{fast}] \neq t$ **then**
    $p_{slow} \leftarrow p_{slow} + 1$;
    $S[p_{slow}] = S[p_{fast}]$;
  end
end
$out \leftarrow p_{slow} + 1$;
return out;

14.1.2 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

Brif: array; two pointers;
Input: ordered set $S = [l, r), l = S[0]$.
Output: T. the squre set.
Initialization:$p_l = 0, p_r = size(S)$, out set T with the same size of S, all elements set to 0,$p_t = size(T)$;
for $i : [0, size(S))$ **do**
  $S[i] \leftarrow S[i]^2$
end
while $p_l \leqslant p_r$ **do**
  if $S[p_l] \leqslant S[p_r]$ **then**
    $T[p_t] = S[p_r]$;
    $p_t \leftarrow p_t - 1$;
    $p_r \leftarrow p_r - 1$;
  end
  else
    $T[p_t] = S[p_l]$;
    $p_t \leftarrow p_t - 1$;
    $p_l \leftarrow p_l + 1$;
  end
end
return T;

14.1.3 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]< bl){
                index = -1;
             }
            else if(p[0]== bl){
                 bl +=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;

  }
};
```

14.1.4 else

Proposition 14.1. 给定 *3* 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

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.
Initialization:$p_l = 0, p_r = 0, len = size(S) + 1, k = 0,$, the sum of the subarray $sum = 0$;
for $p_r : [0, size(S))$ **do**
    $sum \leftarrow sum + s[p_r]$;
    while $sum \geqslant t$ **do**
        $sum \leftarrow sum - s[p_l]$;
        $len = \min(len, p_r - p_l + 1)$;
        $k \leftarrow p_l$;
        $p_l \leftarrow p_l + 1$;
    end
end
return k, len;

Proposition 14.2. 小于 A 的所有质数:$a = \lfloor\sqrt{A}\rfloor . i \in \{2, 3, \cdots, a\}, Answer = \{i|a\}.$

Chapter 15 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 16　工程算法集成和相应软件体系结构

Chapter 17 工程算法分析和评价体系