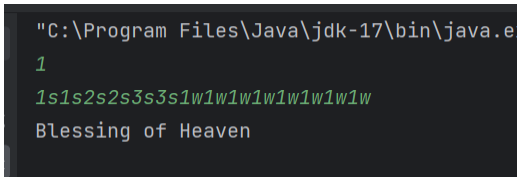
DSAA Lab 爱恨情仇

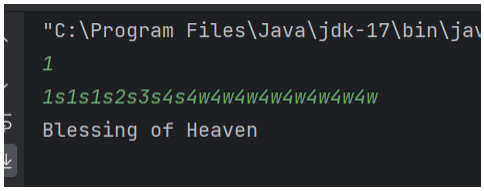
lab 0

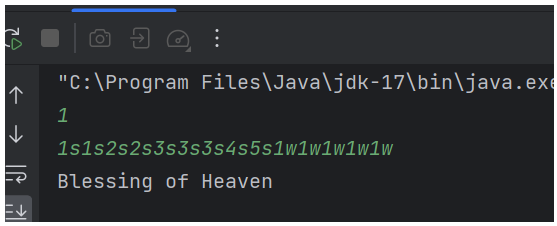
B ① 使用了快读快写

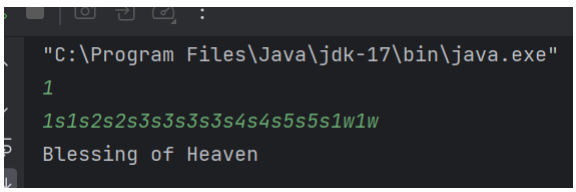
② 和第一题相比 用空间换时间 数组最大只能开到10^6左右

F 麻将！一些想破脑袋想出的珍贵特殊样例：

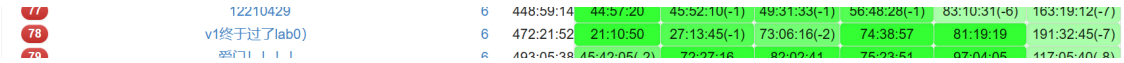








------->



lab 1

A 想想清楚每次循环再次开始时有哪些指针变量需要更新回默认值！ boolean exist = false;

B 打表！
不开long long见祖宗！开long不能直接在超界结果上改数据类型，(long)j*(long)j 而不是(long)(j*j)

C 什么是输出超限:本应为 NO 的地方输出了 YES 0
WA WA WA—一些样例：

测试输出超限
```
6 1
1 2 2 2 4 4
2 4
NO
```
测试结果WA
```
5 1
1 2 3 4 5
1 6
YES 4
```
```
5 4
1 3 4 5 7
2 6
2 5
3 6
3 5
YES 3
YES 2
YES 2
YES 1
```
```
5 1
5 5 5 5 5
5 5
NO
```
```
2 1
1 3
2 4
YES 1
```

对拍程序

D ① 最一开始的第一个算法调用了Combination函数 函数内部运算也有很大复杂度！
② 出现了内存超限的报错，不能乱开很大的数组 128MB的空间限制预估可以放下int
qaq记录一个不小心想到的nb样例：

```
9 8
1 2 2 2 3 3 3 4 5
```

这个样例会在运行的时候mid取到三个2或者三个3中间的那个，这时候就要对左右的重复数字同时进行处理

E 二分时间

F 右边界使用MAX_VALUE 会导致mid=left时数据溢出 因此要适当缩小右边界值 pow(10,16)

lab 2

B 归并排序

可以将数组设为全局static 或者将b作为参数传入merge 这样就避免了在merge里不断new新的数组

C 计算冒泡排序中的交换次数--模拟冒泡排序？×
--所求的交换次数等价于满足i<j, ai>aj的(i,j)数对的个数(这种数对的个数叫做**逆序数**
通过记录intial和final数组，并通过index数组找到两次下标差值？×
--128MB不能开index对应的int范围大小的数组

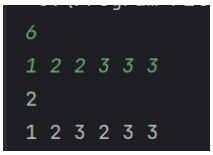
D 更适合使用快排！！

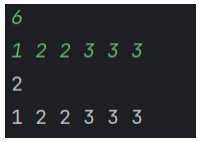
E 让一个最小值不成为中位数--让它前两个后两个都比它大

`ArrayList`在插入和删除操作（尤其是在列表的中间位置）上的性能相对较差，因为在这些情况下需要移动元素

```
ArrayList<Long> seat = new ArrayList<>();
        for (int i = k_index; i < n; i++) {
            seat.add(height[i]);
        }
        for(int i=0;i<k_index;i++){
            seat.add(0+3*i,height[i]);
        }
```

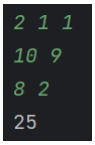
```
for (int i = 0; i < n && big < n; i++) {
            if (i % 3 == 0 && small<k_index) {
                seat[i] = height[small++];
            } else {
                seat[i] = height[big++];
            }
        }
```

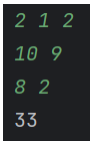


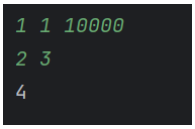
×


√

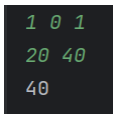
F 首先可以确定的基本原则--一定是对同一个植物超级加倍
但是不能够通过某个策略直接决定要对哪个植物释放加倍魔法







by 见见[考虑n=1]



by Jasmer[考虑h加倍之后依然没有s大]

lab3

最后一个输入是r可能导致数组越界

B char不能用(int)转换 直接就是char数据类型不会报错 但输出结果不正确

C 要开10^5大小的数组 方便访问 而不是N+2大小
一定要有假头假尾 在这题里会很方便 不用处理到边界的情况
如果两个子数组是连在一起的：插入而非交换

运行错误之后：（西巴 要开10001大小的数组 而不是10000！ 所以你干脆用N好了啊！还说什么开10^5大小数组方便访问

D mid 是node 而非int下标

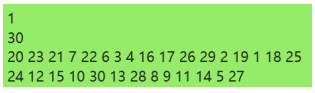
E 一个数组 2 4 1 5 3
对应的地址假设为 0 1 2 3 4 即为a
那么对a复制得到的b按照值大小排序 得到新的b为 2 0 4 1 3

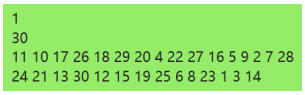
F 整个递减序列同时删除
不能只判断两个节点 而要一直遍历到递减结束
要把递减序列头的前一个存到数组里

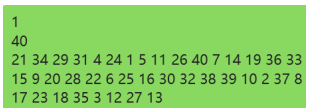
9 10 13 9 5 9 12 10 3 4 8

不要单项比较找发生变化的位置--和其他单调区间没办法区分 直接双向比较找到极值点

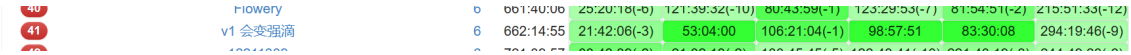
关于F的尾声：你到底也没有放弃 真的很棒







三个大长度样例（纯手搓orz



lab4

A 括号匹配

B 利用时间打标签 从而判断是否已经拿到票
在最后一次循环访问到p/q 但是A[p] B[q]又会数组越界--为AB各多开一个0

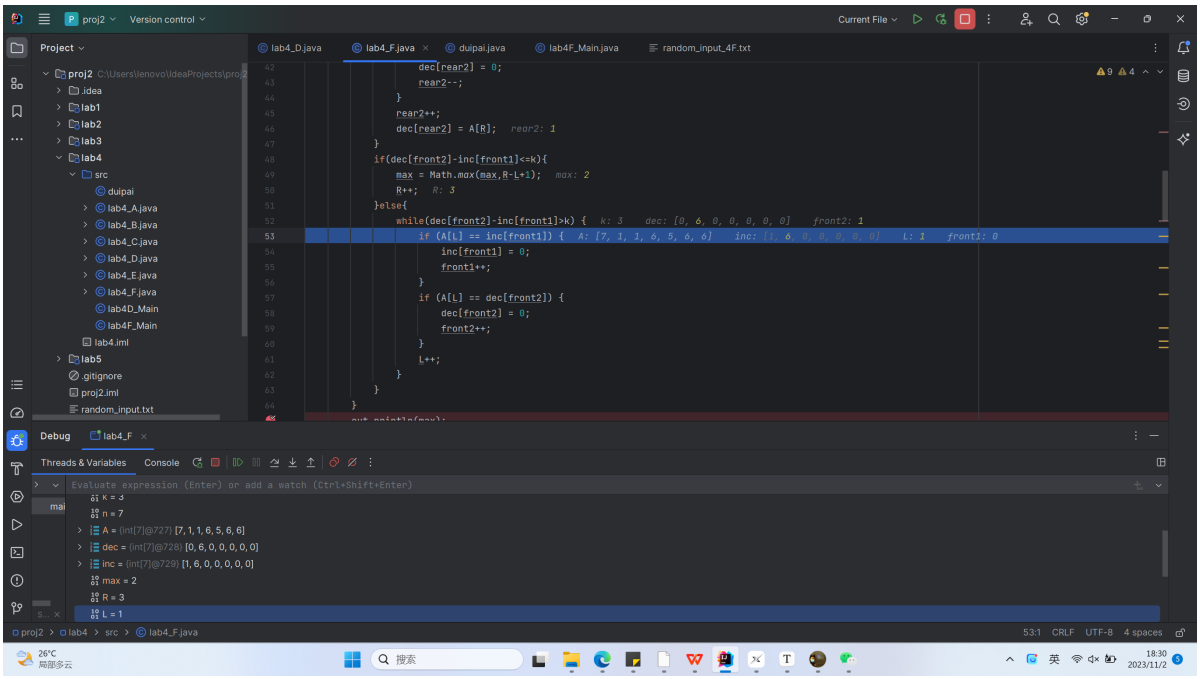
C 要在每次加法和乘法后取余 不能等爆long才取余

D 24135--13425
1 3 4 2 5

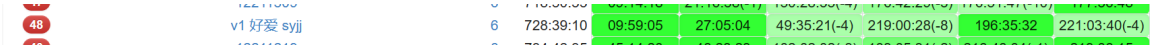
下一张能出的是手里最小的（桶里看是true false） 或者栈顶的（记录一个min）

E 单调队列 按位异或
2 5 3 1 4 2 6 2 7 3 队头扔-过期 队尾扔-破坏单调递减
每个数最多被踢走一次 O(n)复杂度
开node数组 存index和value

F 降序队列求最大值 升序队列求最小值
小于k L往右 大于k R往右
两个指针只会从头至尾走一次 每个数也只会进队出队一次 O(n)复杂度
【L可能在循环中多次左移】



对于一个队列，初始rear指在0 后面访问也要访问rear-1
即rear指的是末尾最后一个数还是末尾第一个空值需要前后保持一致



lab5

A 好做爱做

B 寻找next数组的标准算法

C Transition function的标准算法

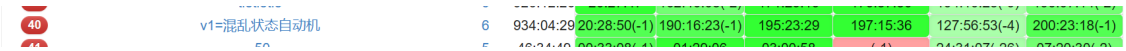
D 循环串的充要条件 len/len-next[len-1]==0 -->len-next[len-1]是总长度减去最长公共前后缀 即可能是循环的部分
注意：next[len-1]=0时也会被整除

E 二分答案
最好不用合数做底数
(双哈希√)

F 将原字符串按照规则编码，在拼接到原字符串后寻找拼接的字符串的最长公共前后缀 （中间可以加个* 从而实现寻找到是小于原串一般长度的最大公共前后缀，避免越过中间点）

```
b c d e f g h i j k l m n o p q r s t u v w x y z a
abababababab
10
```

其实你要找的max是在总长时的最长公共前后缀，而不是next数组中所有值的最大值



lab6

1、有没有根
2、是几叉树
3、父子关系（边的方向）

B 最初读取的时候不知道谁是父谁是子 使用队列确定树的方向 path存储总权重 w存储子节点的所有权重
在没有子结点的结点中访问path选出等于num的

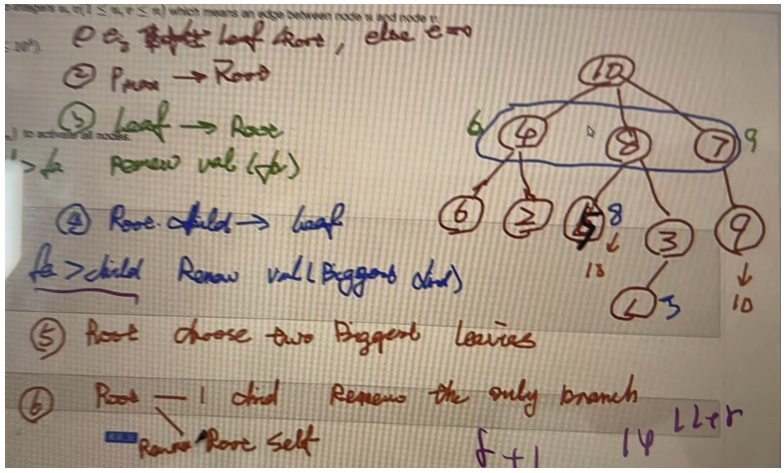
C 二叉树才有前中后序

D

E 每个子树之间互不干扰

F

```
10
1 2
1 3
1 4
2 5
2 6
3 7
3 8
4 9
8 10
10 4 8 7 6 2 5 3 9 1
31
```




```
5
1 2
2 3
2 4
3 5
5 4 3 2 1
12
```

root只有一个子结点

根左右子树都是大顶堆　根左右子树都是小顶堆　根左右子树一个大顶堆一个小顶堆　根只有一个子树且是小顶堆

```
9
1 2
1 3
2 4
2 5
3 6
3 7
4 8
4 9
9 8 7 6 5 4 3 2 1
27
```

```
9
1 2
1 3
2 4
2 5
3 6
3 7
4 8
4 9
9 1 3 2 5 4 7 6 8
33
```

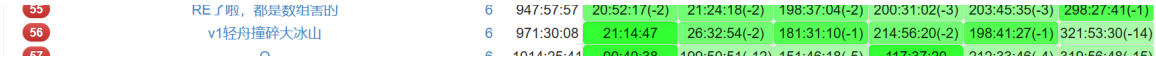
```
9
1 2
1 3
2 4
2 5
3 6
3 7
4 8
4 9
9 1 7 2 5 4 3 6 8
32
```

```
6
1 2
2 3
2 4
3 5
3 6
6 1 2 3 4 5
19
```

根有两个大顶堆子树 但是两个最大叶子在同一个子树

```
9
1 2
1 3
2 4
2 5
3 6
3 7
4 8
4 9
9 8 7 3 4 5 6 2 1
27
```

最不容易的一集——谢谢syjj



lab7

A 如何找根 本题给了边的方向 没有当过儿子的就是root
二叉--完全--父子大小

```
1
4
2 1 4 3
2 1
2 4
1 3
Case #1: YES
```

```
1
4
2 1 4 3
2 1
2 4
4 3
Case #1: NO
```

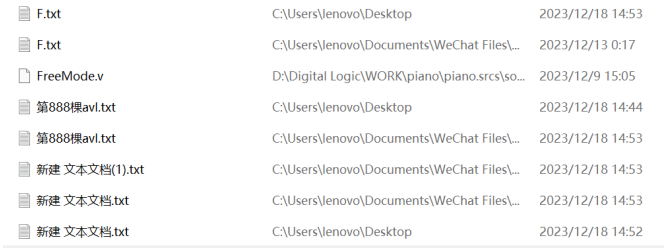
B 数组下标从1开始 大顶堆

C 创建结点堆

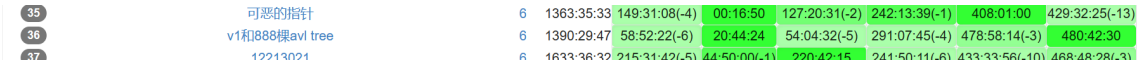
D 从视觉感受上 胡萝卜堆逐渐压缩 故形成的新胡萝卜堆放在形成新堆的两堆位置上
建立小顶堆后 使用堆顶最小值和其index相邻的数字进行运算（建小顶堆时index同时为第二筛选标准）
-- 这会遇到相邻数字在堆内难以删除的问题--为了解决这个问题 使用标签flag打标记
-- 仅使用标记查找下次作为相邻数字运算是否有效 会遇到最坏时间复杂度O(n)的问题 --为了解决这个问

题 使用linklist O(1)查找前后相邻数字
-- 而标记用来做什么呢？它使用于堆顶元素是否有效 --如果无效 那么把它丢掉 而不是略过！--这确保我能每次直接覆盖堆顶为新胡萝卜堆的大小 而不需要delete再insert（因为这时堆顶一定是参加运算的一个胡萝卜堆）

E F **AVL Tree**



这一集是 没有见哥我就死了：（



lab8

有向图还是无向图 有环图还是无环图 有权图还是无权图00

B BFS

C 注意星星不能超过n/2 否则输出心心

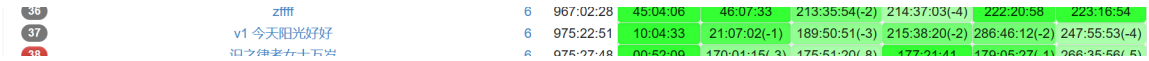
```
1
8 7
1 2
1 3
2 4
2 5
2 6
3 7
3 8
2
2 3
```

注意奇数 count<=n/2

D 先找树根 DFS
利用栈计算在栈时长 从而判断父（祖）子关系
得到出入栈的时间 O(n) 查询m个节点的时间O(m)

E 通过选与不选构建满二叉树 通过不相邻的条件砍掉不能走的子树 递归

F 有向无环无权
路径条数：逆着拓扑序找
拓扑序：对每一个点定义入度 入度为0时进入队列
父节点出队时，子节点入度-- 直至为零入队【保证了每个节点入队时所有的父亲一定已经入队】



lab9

A Dijkstra 只能用于正权（贪心算法求最小值） 单源最短路（确定两点才可以求）

建立堆维护最小dist
截至条件：堆空掉

B prim 算法 可用于负权边 ----最小生成树 其他的边除去负权边求和【可以随便选一个点作为起点】
先让堆顶出队再读入孩子 不然新插入的孩子可能会顶替当前的root

C 最大生成树：访问的点尽量多 尽量找最大的边

D 无向无权图求最小路径 BFS 本题是多源无向无权最短路：初始把所有同色节点丢到队列里 dist[k]

E 有向图 缩点操作
能到全图所有点的点： 一定要从缩点之后入度为零的找
大部分情况下 缩点之后入度为零的点的个数就是答案 cnt
如果S所在的强连通分量入度是零 到自己这个强连通分量不需要加边 cnt-1
随便选一个节点对反图做DFS

F 无权边传送门 使用传送门的个数有限制