知识点

前置

0. 快速幂
1. 质数

   质数个数 $O(n/\log n)$

   埃筛 $O(n\log\log n)$

   欧拉筛 $O(n)$

2. 逆元

   若 $x\times y\equiv 1 \mod p$，则称 x,y 互为模 p 意义下的逆元

   根据费马小定理我们给出一种简单的快速幂求逆元方式：$x^{-1}\equiv x^{p-2} \mod p$，$p$ 为质数时

   特别地，若需要预处理阶乘的逆元，我们可以递推的实现，$(i!)^{-1}=((i+1)!)^{-1}\times(i+1)$，令 $inv[i]$ 表示 $i!$ 的逆元，则 $inv[i-1]\equiv inv[i]\times i \mod p$，我们只需要 $O(\log n)$ 计算 $n!$ 的逆元后 $O(n)$ 递推即可得到所有数阶乘的逆元

3. 秦九韶算法（C1C）

   快速计算 $\Sigma_{i=0}^{n}a_ix^i$，转化为 $(((a_nx+a_{n-1})x+a_{n-2})x+a_{n-3})x+\cdots$

4. 离散化

   结合 `sort` 和 `unique`

5. 并查集

快速维护一些点是否属于同一个集合，几乎实现 $O(1)$ 的合并与 $O(1)$ 的查询

```
int f[maxn],siz[maxn];
int fa(int x) {return f[x]==x?x:f[x]=fa(f[x]);}
int merge(int x,int y) {
    int fx=fa(x),fy=fa(y);
    if(fx!=fy) {
        if(siz[fx]>siz[fy]) {
            f[fy]=fx;
            siz[fx]+=siz[fy];
        }
        else {
            f[fx]=fy;
            siz[fy]+=siz[fx];
        }
    }
}
```

6. 重载运算符/cmp

```
struct Node{
    int u,d;
    bool operator < (const Node rhs) const {
        return d<rhs.d;
    }
};
bool cmp(Node x,Node y) {
    return x.d<y.d;
}
```

基本

1. 卡特兰数（C1B，C3B）

   $f_n = \begin{cases} \Sigma_{i=1}^{n} H_{i-1}H_{n-i}, n \geq 2 \\ 1, n = 0,1 \end{cases}$

   $O(n^2)$ 递推

   $f_n = \frac{1}{n+1}C_{2n}^{n}$

   $O(n)$ 预处理阶乘， $O(1)$ 查询

2. 归并排序（C1H）

   归并过程中求逆序对

3. 桶排序（C2C）

高精度

最好用现成的模板

但是自己也得会写点最基本的，比如某次上机考了求 `2^n` 的高精度

DP

1. 简单DP及输出方案（C3A，C3C，C3D，C3G，C3J，E2G，C4D）

   背包问题

- 01背包，滚动数组优化，倒序
  - 完全背包，滚动数组优化，正序
  - 分组背包

  流水线调度问题

  - $n(1 \le n \le 2e3)$ 个金币在 $t = t_i$ 时刻恰好出现在 $x = x_i$，$y = y_i$ 处，每个单位时间至多移动一格，求最大金币数

2. 区间DP（C3E，C3H，C3I）

   矩阵乘法优化

   OBST

   石子合并

```
for(int l=2;l<=n;l++) {
    for(int i=1;i+l-1<=n;i++) {
        int j=i+l-1;
        for(int k=i;k<j;k++) {
            f[i][j]=min/max(f[i][j],f[i][k]+f[k+1][j]+w(i,j,k));
        }
    }
}
```

3. 最长公共子序列（E2I，C4F）

   $O(n^2)$

4. 最长不下降子序列（E2J）

   $O(n^2)$ 显然，$O(n\log n)$ 也必须会

5. DAG上的DP
   - DAG求最长路
   - 缩点*

6. 复杂一点的DP（属于是不会分类）（E3F）

贪心

感觉所有题目或许都可以叫做 贪心

但是使用贪心时需要证明，否则很有可能是把DP错误的写成了贪心

证明一般可以用最优子状态推全局，或者调整法（反证法）

以独木舟问题为例： n 个人，独木舟容量 m ，每次最多坐两人，每个人体重 $a_i \le m$ ，求最少次数

附一道变形题：

Nerovix 拾到了 $2n$ 个石子。他发现石子两两放在一起的时候格外地美丽，当两个大小分别为 x, y 的石子放在一起配对的时候，刚好可以产生 $(x+y) \bmod k$ 的和谐度。

Nerovix 沉浸在石子的美丽之中。他想请教你，这些石子能产生的最大和谐度是多少。注意每个石子只能被配对一次。

输入格式

第一行，三个正整数 n, m, k（$1 \le m \le 2 \times 10^5, 1 \le k \le 10^9, 1 \le nk \le 4.5 \times 10^{18}$），表示石子配对数，石子种类数，以及和谐度的参数 k。

第二行，m 个正整数 $a_1, \ldots, a_m$（$1 \le a_i \le 10^9$），表示第 i 种石子一共有 a_i 个。

第三行，m 个正整数 $v_1, \ldots, v_m$（$1 \le v_i \le 10^9$），表示第 i 种石子的大小是 v_i。

保证 $\sum a_i = 2n$ 。

输出格式

输出一行，一个整数，表示最大的和谐度。

样例

standard input	standard output
5 3 4 2 3 5 1 3 2	9
10 5 20 4 6 5 3 2 8 12 16 3 14	121

提示

对于第一个样例，选择 $(1,2)$，$(3,3)$，$(1,2)$，$(3,2)$，$(2,2)$ 配对时，取得答案 9。

图论

1. 图的存储
   - 链式前向星（链表）

     适用于稀疏图，即点很多边很少，空间复杂度 $O(m)$ ，但不能快速判断两点间是否有边
   - 邻接矩阵

     适用于稠密图，空间复杂度 $O(n^2)$
   - vector

     适用于所有情况，但常数略大（可能）
2. 图的遍历（E2C）
   - 给定 n 个点及 m 对相对位置关系，问能否确定所有点的位置（保证不冲突）
3. 矩阵BFS（E2E，E3I）
   - 给定 $\{a_{ij}\}$ ，每个位置可以向4个方向走 a_{ij} 格，碰到边界返回，求到达 (n, m) 的最小次数
   - 给定01矩阵，可以到达同一行相同颜色或同一列不同颜色，求到达 (n, m) 的最小次数
4. 最短路（E2H，C4E，E4D）

   Floyd，Dijkstra，Dijkstra堆优化，SPFA（×）
5. 负环（C4G，E4H）

   SPFA

6. 拓扑排序（C4H，C4I，E4A）

   优先推荐BFS

   在DAG上DP

   tarjan缩点*

7. 欧拉路径（C4J）

8. 最小生成树（E4B）

   克鲁斯卡尔 $O(m\log m)$

   克鲁斯卡尔重构树*

9. 网络流（E4C）

   一般来说Dinic当前弧优化 $O(n^2m)$ 足够

10. 二分图匹配（E4E）

不带权匈牙利算法 $O(n^3)$

带权KM算法 $O(n^3)$ （E5D）

计算几何

corner case特别多，考虑一定要全面，任何一种没想到的情况都有可能是测试点

有板子会用就行，不要自己写

字符串

1. 哈希（C6F，C6G，E5F）

   $O(1)$ 实现比较字符串相等

   结合二分答案可用于比较两个字符串公共前缀长度

   还是建议使用双哈希

2. KMP（C6A）

   $O(n)$ 构建 $kmp[i]$ 记录匹配到 i 失配后应跳到哪个位置

   用于字符串匹配，关键在于构建失配指针

3. Z函数（C6E）

   同样可以 $O(n)$ 构建 $z[i]$ 表示 $s[0\sim n-1]$ 与 $s[i\sim n-1]$ 的最长公共前缀

   基本可以实现和KMP相同的功能

4. manacher（C6D）

   $O(n)$ 求以每个字符为中心的最长回文串的半径

5. Trie

   给定 n 个串 s_i ，查询 t 是否与某一个 s_i 相等（或 t 的前缀是否与某一个 s_i 相等）

6. AC自动机*（C6I）

   给定 n 个串 s_i ，查询 t 中每个 s_i 的出现次数

FFT

最基本用于求解多项式乘法

基本原理就是系数表示和点值表示的转换（E5A）

高精度乘法（C6B，C6C）

所有类似于卷积的形式（E5E）

字符串匹配（C6J）

STL

auto

自动获取所需类型

包括基本数据类型，自定义结构体，迭代器等

```
struct Node{
    int x,y;
};
queue<Node> q;
while(!q.empty()) {
    auto u=q.front();
}
```


lower/upper_bound

返回a[]中第一个大于等于/（大于）x的下标

```
int pos1=lower_bound(a+1,a+n+1,x)-a;
int pos2=upper_bound(a+1,a+n+1,x)-a;
```

stack

栈，但不推荐使用STL，建议手写

```
stack<int> s;
s.push(x);
s.top();->int
s.pop();
s.empty();
s.size();
```


queue

（单向）队列，可代替手写循环队列

```
queue<int> q;
q.push(x);
q.front();->int
q.pop();
q.empty();
q.size();
```


dequeue*

双端队列

priority_queue

优先队列（堆）

注意重载运算符，堆默认为大根堆，但是用小于号实现的，如下定义小根堆

```
priority_queue<int,vector<int>,greater<int>> q;
```

pair

可以用来代替一些便捷的自定义struct

且pair自带小于号，可直接用于排序，第一关键字为第一维升序，第二关键字为第二维升序

```
pair<int,int> p1;
pair<int,string> p2;
pair<double,int> p3;
```

vector

```
vector<int> v;
v.push_back();
v.size();
for(int i=0;i<=v.size()-1;i++) {//错误
    cout<<v[i]<<" ";
}
for(int i=0;i<=(int)v.size()-1;i++) {//(正确)
    cout<<v[i]<<" ";
}
```


set

集合，无重复，有序

```
set<int> s;
//插入
for(int i=1;i<=n;i++)
    s.insert(gint());
//遍历
for(auto x : s)
    cout<<x<<" ";
s.begin():返回第一个元素的迭代器
s.end():返回最后一个元素后的迭代器
//故也可以这样遍历
for(auto it=s.begin();it!=s.end();it++)
    cout<<(*it)<<" ";
s.lower_bound(x):返回第一个大于等于x的值的迭代器
```


map

构建一个映射关系

复杂度为 $O(\log n)$

```
map<T1,T2> mp;
map<int,int> mp1;
map<string,int> mp2;
map<int,set<int> mp3;
map<int,vector<int>> mp4;
```


做题思想/技巧

0. 输入输出优化

   卡常是不合理，但是io优化不可忽略

   一般来说的话 `scanf/printf` 是足够的，如果数据量较大（1e6）建议还是使用快读快输（即转化为字符串输入输出）

1. 尺取法/双指针（C2F，E1J）

   维护一段连续区间的相关性质，要求可以快速地（ $O(1)/O(\log n)$ ）地处理修改信息

   - 求一个长为 n 的序列中恰好为 k 的一个排列的子段数
   - 求满足$x+y,x-y$ 为质数且 $x_1 \le x \le x_2, y_1 \le y \le y_2$ 的 (x,y) 的个数

2. 随机（C2H，E1G）

   - 给定 I_n^3 中的 n 个互不相同的点 $\{P_i\}$，选取一个点 A 使得 $A, P_i, P_j, (i \neq j)$ 不共线

     C_n^2 条直线每条最多经过 n 个点，占据空间内约 $\frac{1}{2}$ 的点，故随机即可找到 A

   - 多个随机数中选最多的数使其的 $\gcd > 1$

3. 二分（E3D，C5J）

   - 二分查找：找第一个大于等于的数
   - 二分答案：如何写check，注意可二分性，最小的满足条件的

     ```
     int l=1,r=n;
     while(l<=r) {
         int mid=(l+r)>>1;
         if(check(mid)) r=mid-1;
         else l=mid+1;
     }
     cout<<l<<endl;
     ```

4. 差分

   令 $b_i = a_i - a_{i-1}$ ，则 $a_i = \Sigma_{j=1}^{i} b_j$ ，这样区间 $[l,r]$ 修改 d 可以转化为 $b_l+=d, b_{r+1}-=d$

5. dfn序

6. 二进制枚举子集

   ```
   for(int s=0;s<(1<<n);s++) {
       cout<<s<<":\n";
       for(int i=0;i<n;i++) {
           if(s&(1<<i)) {
               cout<<i+1<<" ";
           }
       }
       cout<<"\n";
   }
   ```

7. 打表（E2D）

- 求各位数的 n 次方之和等于本身的数的个数

8. 非黑即白（？）（E4F）
   - $n \times n$ 的山，每个点高度两两不同，给出一条走遍所有点的路径，要求上坡次数不超过下坡次数

数据结构*

1. ST表

   查询静态区间信息，维护的信息需要满足如下性质，设运算为 f

   $f([l,r]) = f(f([l,\bar{r}]), f([\bar{l},r])), \bar{r} \le \bar{l}$ ，简单来说就是这种运算具有可重叠性（自己瞎起的名词）

   说的不那么抽象就是比如区间[1,5]的信息可由区间[1,4]和[2,5]直接合并而来

   易知 $\max, \min, \wedge, \vee$ 等运算具有此性质， $+$ 不具有此性质

   采用的是倍增的思想以区间最大值为例，设 $f[i,j]$ 表示区间 $[i, i+2^j-1]$ 的最大值，则 $f[i,j] = \max(f[i,j-1], f[i+2^{j-1}, j-1])$ ，以此可以 $O(n\log n)$ 地预处理

   关于查询，令 $x = \log_2(r-l+1)$ ，则 $[l, l+2^x-1], [r-2^x+1, r]$ 可以覆盖整个区间，也即 $ans = \max(f[l,x], f[r-2^x+1, x])$ ，实现了 $O(1)$ 查询

2. 树状数组

   维护前缀信息，支持 单点修改+区间查询 或 区间修改+单点查询（差分）

```
void modify(int x,int k) {
    for(;x<=n;x+=x&-x)
        c[x]+=k;
}
int query(int x) {
    int res=0;
    for(;x;x-=x&x-x)
        res+=x;
    return res;
}
```

   以上为单调修改，区间求和，凡是前缀信息都可以维护，比如说单点修改，求前缀 $\min$

   最基本的应用比如求逆序对，常结合离散化，下标存数

3. 线段树

   维护更复杂的区间信息，可以同时实现单点/区间修改/查询

注意事项

1. io优化
2. 数组大小

   如无向图双倍空间，FFT四倍空间，线段树四倍空间

   空间计算 `sizeof` ，如下方法计算 a 数组所占空间（MB）：

```
cout<<sizeof a/1024/1024;
```

3. corner case

   $n = 0, 1 \quad a_i = 0, 1e9, -1e9$

4. 初始化
5. 多测清空

   不要滥用 `memset` ，复杂度是 $O(n)$ 的，多测 $T \le 1e5, \Sigma n \le 1e5$ 是会被卡到 $O(n^2)$ 的，可这样用：

   ```
   memset(a,0,sizeof(int)*(n+1));//正确
   memset(a,0,sizeof a);//超时
   ```

   以及`memset` 是按字节填充，初始化最大值建议 `0x3f` ，这样相加不会爆 `int` ， `0xff` 意味着全部为1（包括符号位），故对应的是 `-1`
6. 不要卡题

例题

从去年上机题随便找了几道

1. E1F

   #### 题目描述

   小水獭正在学习二叉树，它希望世界上所有的树都是二叉树。

   为了实现这个目的，小水獭请求莫卡教自己二叉树魔法，使自己可以把世界上所有的树变成二叉树。

   为了检验小水獭的魔法天赋，莫卡给了小水獭一棵有 n 个节点的二叉树，且根节点的编号为 1。这棵二叉树任意一个节点要么是白色，要么是黑色。之后莫卡会对这棵二叉树释放 q 次魔法，每次莫卡会选择一个节点，将以这个节点为根的子树内所有节点的颜色反转，即黑色变成白色，白色变成黑色。在莫卡的 q 次魔法全部释放完成之后，小水獭需要回答每个节点的颜色。

   但是这个问题对于小水獭来说过于困难了，请聪明的你帮他回答这个问题吧。

   #### 输入格式

   第一行一个正整数 t（$1 \le t \le 5$），表示数据组数。

   对于每组数据，第一行一个正整数 n（$1 \le n \le 10^5$），表示二叉树的节点数量。

   第二行 $n-1$ 个正整数，第 i（$1 \le i \le n-1$）个数表示编号为 $i+1$ 的节点的父亲节点编号，数据保证是一棵二叉树。

   第三行一个长度为 n 的 01 串，从左到右第 i（$1 \le i \le n$）位如果为 0，表示编号为 i 的节点颜色为白色，否则为黑色。

   第四行一个正整数 q（$1 \le q \le 10^5$），表示魔法的释放次数。

   接下来 q 行每行一个正整数 a_i（$1 \le a_i \le n$），表示第 i 次魔法选择的节点编号。

   #### 输入格式

   对于每组数据，输出一行一个长度为 n 的 01 串，表示 q 次魔法全部释放完成之后每个节点的颜色。从左到右第 i（$1 \le i \le n$） 位如果为 0，表示编号为 i 的节点颜色为白色，否则为黑色。

   #### 输入样例

   ```
   1
   6
   3 1 1 3 4
   001110
   3
   1
   3
   2
   ```

   #### 输出样例

   ```
   111011
   ```

2. E1G

成功变成猫猫的 XIAO7 开设了一家猫娘乐园，但由于猫猫太多，XIAO7 的猫粮不够吃了。

于是天上传来一个声音：「如果你能答对以下若干个问题，就会有足够多的猫粮从天而降。」

请你救救 XIAO7 和她的猫猫们!

具体来说，给定一个整数 n，不妨将其看作一个十进制数字串，规定如下两种操作：

- 删除数字串中的任何一位数字（允许执行此操作而出现数字串为空的情况）。
- 在数字串中的最右侧添加一位数字。

以上操作可以**按任意顺序执行任意次数**。请问最少需要执行多少次操作才能使数字串对应的整数成为 2 的任意非负整数次幂（即 $2^i, i \in \mathbb{N}$）？

最终得到的数字串**不能有前导零**。

输入格式

第一行一个正整数 t（$1 \le t \le 10^4$），表示数据组数。

对于每组数据，一行一个正整数 n（$1 \le n \le 10^9$），含义同题目描述。

输出格式

对于每组数据，输出一行一个非负整数，表示最少操作次数。

输入样例

```
8
1256
701
12
95
512
7635
25765
100000
```

输出样例

```
1
2
1
3
0
4
2
5
```

3. C1F

题目描述

Zhoues 很喜欢研读《算法导论》，有一天在暗中观察该书第 24 页的时候，想到了一个问题并需要你来帮他解决，问题如下：

- 给定一个序列 $a_1, a_2, \dots, a_n$ 和一个正整数 k，如果 $1 \le i < j \le n$ 且 $a_i > k \cdot a_j$ 我们就将 (i, j) 称作一个**逆序 k 倍对**。请你计算序列中逆序 k 倍对的个数。

输入格式

第一行两个正整数 n, k（$1 \le n \le 10^5$，$1 \le k \le 10$），含义同题目描述。

第二个 n 个正整数 $a_1, a_2, \dots, a_n$（$1 \le a_i < 2^{31}$），含义同题目描述。

对于得分占比 10% 的测试点，保证 $1 \le n \le 100$。

输出格式

一行一个非负整数，表示序列中逆序 k 倍对的个数。

输入样例

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

输出样例

```
4
```

4. C1G

题目描述

Zhoues 这个同学估计大家都不怎么认识，但是这不重要，重要的是你只需要知道他有一项特殊的能力——**暗中观察**。他经常通过这个能力轻松看透一些事物背后的算法原理。

虽然学校因为疫情封校，但是却校园内却盛行一款游戏：

- 该游戏中有小 A 和小 B 两个角色，初始给定两个正整数 n 和 k，小 A 和小 B 轮流对 n 减去 k 的任意非负整数次幂（即 $k^i, i \in \mathbb{N}$），但需要保证 n 始终非负，最终在自己回合将 n 变为 0 的一方获胜。

每次游戏开始都是小 A 先手，且小 A 小 B 两个人都会采取最优策略，那么每轮谁会赢呢？

Zhoues 暗中观察了几场这两款游戏的对局，已经看透了背后的算法奥秘，现在你只要告诉他游戏的初始情况，他就可以直接告诉你最后的赢家是谁！

输入格式

第一行一个正整数 t（$1 \le t \le 50$），表示数据组数。

对于每组数据，一行两个正整数 n, k（$1 \le n, k \le 10^3$），含义同题目描述。

输出格式

对于每组数据，输出一行：

- 若小 A 获胜，输出 `~A~`。
- 若小 B 获胜，输出 `~B~`。

输入样例

```
2
3 2
3 4
```

输出样例

```
~B~
~A~
```

5. 计网课上有一道题：一条街道安装无线网络，需要放置M个路由器。整条街道上一共有N户居民，分布在一条直线上，每一户居民必须被至少一台路由器覆盖到。现在的问题是所有路由器的覆盖半径是一样的，我们希望用覆盖半径尽可能小的路由器来完成任务，因为这样可以节省成本。

输入

输入第一行包含两个整数M和N，以下N行每行一个整数Hi表示该户居民在街道上相对于某个点的坐标。

输出

输出仅包含一个数，表示最小的覆盖半径，保留一位小数。

输入样例

```
2 3
1
3
10
```

输出样例

```
1.0
```

HINT

【样例输出】（在2，10位置上各放一个）

【数据规模】

对于100%的数据，有1 ≤N, M ≤100000，-10000000 ≤Hi ≤10000000。

已知一个完全图唯一的最小生成树（即知道这个树所有边的端点和权值）,其余的边权值未知，问这个完全图所有边权值和的最小值。

完全图是每对顶点之间都恰连有一条边的简单图。

输入

第一行一个整数t表示数据组数($1 \leq t \leq 10$)

每组数据第一行一个正整数n，表示完全图的点数($2 \leq n \leq 10^5$)

接下来n-1行，每行三个整数x，y，z，表示x,y之间有一条权值为z的边（无向边）($1 \leq x, y \leq n$，$1 \leq z \leq 10000$)

输出

每组数据一行一个整数

6.
输入样例

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

输出样例

```
9
29
```