后缀数组的应用

寻找最小的循环移动位置

将字符串 S 复制一份变成 SS 就转化成了后缀排序问题。

在字符串中找子串

任务是在线地在主串 T 中寻找模式串 S。在线的意思是，我们已经预先知道知道主串 T，但是当且仅当询问时才知道模式串 S。我们可以先构造出 T 的后缀数组，然后查找子串 S。若子串 S 在 T 中出现，它必定是 T 的一些后缀的前缀。因为我们已经将所有后缀排序了，我们可以通过在 p 数组中二分 S 来实现。比较子串 S 和当前后缀的时间复杂度为 $O(|S|)$，因此找子串的时间复杂度为 $O(|S|\log|T|)$。注意，如果该子串在 T 中出现了多次，每次出现都是在 p 数组中相邻的。因此出现次数可以通过再次二分找到，输出每次出现的位置也很轻松。

从字符串首尾取字符最小化字典序

题意：给你一个字符串，每次从首或尾取一个字符组成字符串，问所有能够组成的字符串中字典序最小的一个。
　暴力做法就是每次最坏 $O(n)$ 地判断当前应该取首还是尾（即比较取首得到的字符串与取尾得到的反串的大小），只需优化这一判断过程即可。

由于需要在原串后缀与反串后缀构成的集合内比较大小，可以将反串拼接在原串后，并在中间加上一个没出现过的字符（如 `#`，代码中可以直接使用空字符），求后缀数组，即可 $O(1)$ 完成这一判断。

height 数组

LCP（最长公共前缀）

两个字符串 S 和 T 的 LCP 就是最大的 $x(x\le\min(|S|,|T|))$ 使得 $S_i=T_i\ (\forall\,1\le i\le x)$。

下文中以 $lcp(i,j)$ 表示后缀 i 和后缀 j 的最长公共前缀（的长度）。

O(n) 求 height 数组的代码实现

利用上面这个引理暴力求即可：

```
for (i = 1, k = 0; i <= n; ++i) {
  if (rk[i] == 0) continue;
  if (k) --k;
  while (s[i + k] == s[sa[rk[i] - 1] + k]) ++k;
  height[rk[i]] = k;
}
```

k 不会超过 n，最多减 n 次，所以最多加 $2n$ 次，总复杂度就是 $O(n)$。

height 数组的应用

两子串最长公共前缀

$lcp(sa[i], sa[j]) = \min\{height[i+1..j]\}$

感性理解：如果 $height$ 一直大于某个数，前这么多位就一直没变过；反之，由于后缀已经排好序了，不可能变了之后变回来。

严格证明可以参考[[2004] 后缀数组 by. 徐智磊][1]。

有了这个定理，求两子串最长公共前缀就转化为了 RMQ 问题。

比较一个字符串的两个子串的大小关系

假设需要比较的是 $A = S[a..b]$ 和 $B = S[c..d]$ 的大小关系。

若 $lcp(a, c) \ge \min(|A|, |B|)$，$A < B \iff |A| < |B|$。

否则，$A < B \iff rk[a] < rk[c]$。

不同子串的数目

子串就是后缀的前缀，所以可以枚举每个后缀，计算前缀总数，再减掉重复。

「前缀总数」其实就是子串个数，为 $n(n+1)/2$。

如果按后缀排序的顺序枚举后缀，每次新增的子串就是除了与上一个后缀的 LCP 剩下的前缀。这些前缀一定是新增的，否则会破坏 $lcp(sa[i], sa[j]) = \min\{height[i+1..j]\}$ 的性质。只有这些前缀是新增的，因为 LCP 部分在枚举上一个前缀时计算过了。

所以答案为：

$$\frac{n(n+1)}{2} - \sum_{i=2}^{n} height[i]$$

出现至少 k 次的子串的最大长度

出现至少 k 次意味着后缀排序后有至少连续 k 个后缀以这个子串作为公共前缀。

所以，求出每相邻 $k-1$ 个 $height$ 的最小值，再求这些最小值的最大值就是答案。

可以使用单调队列 $O(n)$ 解决，但使用其它方式也足以 AC。

是否有某字符串在文本串中至少不重叠地出现了两次

可以二分目标串的长度 $|s|$，将 h 数组划分成若干个连续 LCP 大于等于 $|s|$ 的段，利用 RMQ 对每个段求其中出现的数中最大和最小的下标，若这两个下标的距离满足条件，则一定有长度为 $|s|$ 的字符串不重叠地出现了两次。

连续的若干个相同子串

我们可以枚举连续串的长度 $|s|$，按照 $|s|$ 对整个串进行分块，对相邻两块的块首进行 LCP 与 LCS 查询，具体可见[[2009] 后缀数组——处理字符串的有力工具][2]。

结合并查集

某些题目求解时要求你将后缀数组划分成若干个连续 LCP 长度大于等于某一值的段，亦即将 h 数组划分成若干个连续最小值大于等于某一值的段并统计每一段的答案。如果有多次询问，我们可以将询问离线。观察到当给定值单调递减的时候，满足条件的区间个数总是越来越少，而新区间都是两个或多个原区间相连所得，且新区间中不包含在原区间内的部分的 h 值都为减少到的这个值。我们只需要维护一个并查集，每次合并相邻的两个区间，并维护统计信息即可。

结合线段树

某些题目让你求满足条件的前若干个数，而这些数又在后缀排序中的一个区间内。这时我们可以用归并排序的性质来合并两个结点的信息，利用线段树维护和查询区间答案。

检查字符串是否出现

> 给一个文本串 T 和多个模式串 P，我们要检查字符串 P 是否作为 T 的一个子串出现。

我们在 $O(|T|)$ 的时间内对文本串 T 构造后缀自动机。为了检查模式串 P 是否在 T 中出现，我们沿转移（边）从 t_0 开始根据 P 的字符进行转移。如果在某个点无法转移下去，则模式串 P 不是 T 的一个子串。如果我们能够这样处理完整个字符串 P，那么模式串在 T 中出现过。

对于每个字符串 P，算法的时间复杂度为 $O(|P|)$。此外，这个算法还找到了模式串 P 在文本串中出现的最大前缀长度。

不同子串个数

> 给一个字符串 S，计算不同子串的个数。

对字符串 S 构造后缀自动机。

每个 S 的子串都相当于自动机中的一些路径。因此不同子串的个数等于自动机中以 t_0 为起点的不同路径的条数。

考虑到 SAM 为有向无环图，不同路径的条数可以通过动态规划计算。即令 d_v 为从状态 v 开始的路径数量（包括长度为零的路径），则我们有如下递推方程：

$$d_v = 1 + \sum_{w:(v,w,c)\in DAWG} d_w$$

即，d_v 可以表示为所有 v 的转移的末端的和。

所以不同子串的个数为 $d_{t_0} - 1$（因为要去掉空子串）。

总时间复杂度为：$O(|S|)$。

另一种方法是利用上述后缀自动机的树形结构。每个节点对应的子串数量是 $\operatorname{len}(i) - \operatorname{len}(\operatorname{link}(i))$，对自动机所有节点求和即可。

所有不同子串的总长度

> 给定一个字符串 S，计算所有不同子串的总长度。

本题做法与上一题类似，只是现在我们需要考虑分两部分进行动态规划：不同子串的数量 d_v 和它们的总长度 ans_v。

我们已经在上一题中介绍了如何计算 d_v。ans_v 的值可以通过以下递推式计算：

$$ans_v = \sum_{w:(v,w,c)\in DAWG} d_w + ans_w$$

我们取每个邻接结点 w 的答案，并加上 d_w（因为从状态 v 出发的子串都增加了一个字符）。

算法的时间复杂度仍然是 $O(|S|)$。

同样可以利用上述后缀自动机的树形结构。每个节点对应的所有后缀长度是 $\frac{\operatorname{len}(i)\times(\operatorname{len}(i)+1)}{2}$，减去其 link 节点的对应值就是该节点的净贡献，对自动机所有节点求和即可。

字典序第 k 大子串

> 给定一个字符串 S。多组询问，每组询问给定一个数 K_i，查询 S 的所有子串中字典序第 K_i 大的子串。

解决这个问题的思路可以从解决前两个问题的思路发展而来。字典序第 k 大的子串对应于 SAM 中字典序第 k 大的路径，因此在计算每个状态的路径数后，我们可以很容易地从 SAM 的根开始找到第 k 大的路径。

预处理的时间复杂度为 $O(|S|)$，单次查询的复杂度为 $O(|ans| \cdot |\Sigma|)$（其中 ans 是查询的答案，$|\Sigma|$ 为字符集的大小）。

> 虽然该题是后缀自动机的经典题，但实际上这题由于涉及字典序，用后缀数组做最方便。

最小循环移位

> 给定一个字符串 S。找出字典序最小的循环移位。

容易发现字符串 $S+S$ 包含字符串 S 的所有循环移位作为子串。

所以问题简化为在 $S+S$ 对应的后缀自动机上寻找最小的长度为 $|S|$ 的路径，这可以通过平凡的方法做到：我们从初始状态开始，贪心地访问最小的字符即可。

总的时间复杂度为 $O(|S|)$。

出现次数

> 对于一个给定的文本串 T，有多组询问，每组询问给一个模式串 P，回答模式串 P 在字符串 T 中作为子串出现了多少次。

利用后缀自动机的树形结构，进行 dfs 即可预处理每个节点的终点集合大小。在自动机上查找模式串 P 对应的节点，如果存在，则答案就是该节点的终点集合大小；如果不存在，则答案为 0.

以下为原方法：

> 对文本串 T 构造后缀自动机。
>
> 接下来做预处理：对于自动机中的每个状态 v，预处理 cnt_v，使之等于 $\operatorname{endpos}(v)$ 集合的大小。事实上，对应同一状态 v 的所有子串在文本串 T 中的出现次数相同，这相当于集合 endpos 中的位置数。
>
> 然而我们不能明确的构造集合 endpos，因此我们只考虑它们的大小 cnt。
>
> 为了计算这些值，我们进行以下操作。对于每个状态，如果它不是通过复制创建的（且它不是初始状态 t_0），我们将它的 cnt 初始化为 1。然后我们按它们的长度 len 降序遍历所有状态，并将当前的 cnt_v 的值加到后缀链接指向的状态上，即：
>
> $$cnt_{\operatorname{link}(v)} += cnt_v$$
>
> 这样做每个状态的答案都是正确的。
>
> 为什么这是正确的？不是通过复制获得的状态，恰好有 $|T|$ 个，并且它们中的前 i 个在我们插入前 i 个字符时产生。因此对于每个这样的状态，我们在它被处理时计算它们所对应的位置的数量。因此我们初始将这些状态的 cnt 的值赋为 1，其它状态的 cnt 值赋为 0。
>
> 接下来我们对每一个 v 执行以下操作：$cnt_{\operatorname{link}(v)} += cnt_v$。其背后的含义是，如果有一个字符串 v 出现了 cnt_v 次，那么它的所有后缀也在完全相同的地方结束，即也出现了 cnt_v 次。

> 为什么我们在这个过程中不会重复计数（即把某些位置数了两次）呢？因为我们只将一个状态的位置添加到 **一个** 其它的状态上，所以一个状态不可能以两种不同的方式将其位置重复地指向另一个状态。
>
> 因此，我们可以在 $O(|T|)$ 的时间内计算出所有状态的 cnt 的值。
>
> 最后回答询问只需要查找值 cnt_t，其中 t 为模式串对应的状态，如果该模式串不存在答案就为 0。单次查询的时间复杂度为 $O(|P|)$。

第一次出现的位置

> 给定一个文本串 T，多组查询。每次查询字符串 P 在字符串 T 中第一次出现的位置（P 的开头位置）。

我们构造一个后缀自动机。我们对 SAM 中的所有状态预处理位置 $\operatorname{firstpos}$。即，对每个状态 v 我们想要找到第一次出现这个状态的末端的位置 $\operatorname{firstpos}[v]$。换句话说，我们希望先找到每个集合 endpos 中的最小的元素（显然我们不能显式地维护所有 endpos 集合）。

为了维护 $\operatorname{firstpos}$ 这些位置，我们将原函数扩展为 `sam_extend()`。当我们创建新状态 cur 时，我们令：

$$\operatorname{firstpos}(cur) = \operatorname{len}(cur) - 1$$

；当我们将结点 q 复制到 $clone$ 时，我们令：

$$\operatorname{firstpos}(clone) = \operatorname{firstpos}(q)$$

（因为值的唯一的其它选项 $\operatorname{firstpos}(cur)$ 显然太大了）。

那么查询的答案就是 $\operatorname{firstpos}(t) - |P| + 1$，其中 t 为对应字符串 P 的状态。单次查询只需要 $O(|P|)$ 的时间。

所有出现的位置

> 问题同上，这一次需要查询文本串 T 中模式串出现的所有位置。

利用后缀自动机的树形结构，遍历子树，一旦发现终点节点就输出。

以下为原解法：

> 我们还是对文本串 T 构造后缀自动机。与上一个问题相似，我们为所有状态计算位置 $\operatorname{firstpos}$。
>
> 如果 t 为对应于模式串 T 的状态，显然 $\operatorname{firstpos}(t)$ 为答案的一部分。需要查找的其它位置怎么办？我们使用了含有字符串 P 的自动机，我们还需要将哪些状态纳入自动机呢？所有对应于以 P 为后缀的字符串的状态。换句话说我们要找到所有可以通过后缀链接到达状态 t 的状态。
>
> 因此为了解决这个问题，我们需要为每一个状态保存一个指向它的后缀引用列表。查询的答案就包含了对于每个我们能从状态 t 只使用后缀引用进行 DFS 或 BFS 的所有状态的 $\operatorname{firstpos}$ 值。
>
> 这种变通方案的时间复杂度为 $O(answer(P))$，因为我们不会重复访问一个状态（因为对于仅有一个后缀链接指向一个状态，所以不存在两条不同的路径指向同一状态）。
>
> 我们只需要考虑两个可能有相同 endpos 值的不同状态。如果一个状态是由另一个复制而来的，则这种情况会发生。然而，这并不会对复杂度分析造成影响，因为每个状态至多被复制一次。
>
> 此外，如果我们不从被复制的节点输出位置，我们也可以去除重复的位置。事实上对于一个状态，如果经过被复制状态可以到达，则经过原状态也可以到达。因此，如果我们给每个状态记录标记 `is_clone` 来代表这个状态是不是被复制出来的，我们就可以简单地忽略掉被复制的状态，只输出其它所有状态的 $firstpos$ 的值。

> 以下是大致的实现：
>
> ```
> struct state {
> bool is_clone;
> int first_pos;
> std::vector<int> inv_link;
> // some other variables
> };
>
> // 在构造 SAM 后
> for (int v = 1; v < sz; v++) st[st[v].link].inv_link.push_back(v);
>
> // 输出所有出现位置
> void output_all_occurrences(int v, int P_length) {
> if (!st[v].is_clone) cout << st[v].first_pos - P_length + 1 << endl;
> for (int u : st[v].inv_link) output_all_occurrences(u, P_length);
> }
> ```

最短的没有出现的字符串

> 给定一个字符串 S 和一个特定的字符集，我们要找一个长度最短的没有在 S 中出现过的字符串。

我们在字符串 S 的后缀自动机上做动态规划。

令 d_v 为节点 v 的答案，即，我们已经处理完了子串的一部分，当前在状态 v，想找到不连续的转移需要添加的最小字符数量。计算 d_v 非常简单。如果不存在使用字符集中至少一个字符的转移，则 $d_v = 1$。否则添加一个字符是不够的，我们需要求出所有转移中的最小值：

$$d_v = 1 + \min_{w:(v,w,c)\in SAM} d_w$$

问题的答案就是 d_{t_0}，字符串可以通过计算过的数组 d 逆推回去。

两个字符串的最长公共子串

> 给定两个字符串 S 和 T，求出最长公共子串，公共子串定义为在 S 和 T 中都作为子串出现过的字符串 X。

我们对字符串 S 构造后缀自动机。

我们现在处理字符串 T，对于每一个前缀，都在 S 中寻找这个前缀的最长后缀。换句话说，对于每个字符串 T 中的位置，我们想要找到这个位置结束的 S 和 T 的最长公共子串的长度。显然问题的答案就是所有 l 的最大值。

为了达到这一目的，我们使用两个变量，**当前状态** v 和 **当前长度** l。这两个变量描述当前匹配的部分：它的长度和它们对应的状态。

一开始 $v = t_0$ 且 $l = 0$，即，匹配为空串。

现在我们来描述如何添加一个字符 T_i 并为其重新计算答案：

- 如果存在一个从 v 到字符 T_i 的转移，我们只需要转移并让 l 自增一。
- 如果不存在这样的转移，我们需要缩短当前匹配的部分，这意味着我们需要按照后缀链接进行转移：

$$v = \operatorname{link}(v)$$

与此同时，需要缩短当前长度。显然我们需要将 l 赋值为 $\operatorname{len}(v)$，因为经过这个后缀链接后我们到达的状态所对应的最长字符串是一个子串。

- 如果仍然没有使用这一字符的转移，我们继续重复经过后缀链接并减小 l，直到我们找到一个转移或到达虚拟状态 -1（这意味着字符 T_i 根本没有在 S 中出现过，所以我们设置 $v = l = 0$）。

这一部分的时间复杂度为 $O(|T|)$，因为每次移动我们要么可以使 l 增加一，要么可以在后缀链接间移动几次，每次都减小 l 的值。

代码实现：

```
string lcs(const string &S, const string &T) {
  sam_init();
  for (int i = 0; i < S.size(); i++) sam_extend(S[i]);

  int v = 0, l = 0, best = 0, bestpos = 0;
  for (int i = 0; i < T.size(); i++) {
    while (v && !st[v].next.count(T[i])) {
      v = st[v].link;
      l = st[v].length;
    }
    if (st[v].next.count(T[i])) {
      v = st[v].next[T[i]];
      l++;
    }
    if (l > best) {
      best = l;
      bestpos = i;
    }
  }
  return T.substr(bestpos - best + 1, best);
}
```

多个字符串间的最长公共子串

> 给定 k 个字符串 S_i。我们需要找到它们的最长公共子串，即作为子串出现在每个字符串中的字符串 X。

我们将所有的子串连接成一个较长的字符串 T，以特殊字符 D_i 分开每个字符串（一个字符对应一个字符串）：

$$T = S_1 + D_1 + S_2 + D_2 + \cdots + S_k + D_k.$$

然后对字符串 T 构造后缀自动机。

现在我们需要在自动机中找到存在于所有字符串 S_i 中的一个字符串，这可以通过使用添加的特殊字符完成。注意如果 S_j 包含了一个子串，则 SAM 中存在一条从包含字符 D_j 的子串而不包含以其它字符 $D_1, \ldots, D_{j-1}, D_{j+1}, \ldots, D_k$ 开始的路径。

因此我们需要计算可达性，即对于自动机中的每个状态和每个字符 D_i，是否存在这样的一条路径。这可以容易地通过 DFS 或 BFS 及动态规划计算。之后，问题的答案就是状态 v 的字符串 $\operatorname{longest}(v)$ 中存在所有特殊字符的路径。

赛前：

1. 到参赛地后要时刻不忘自己是来比赛的，**好好休息、备战**。

2. 参赛前一天要睡 **10 个小时以上，10 个小时以上，10 个小时以上，**前一天晚饭与当天早饭要吃好。

3. 到新环境，时刻注意远离疾病，感冒肠炎病不大，却是成绩的天敌。

对一个队伍比赛选手的一些要求：

4. 每次练习赛都要当作正式比赛来做，要确保所有的题都看过，赛后要把没做出来的题尽量补上。

5. 可能的话多看看以往比赛的总结、照片和录象，缩短与正式竞赛的距离，避免正式竞赛时紧张得做不出题等情况。

6. 最好的情况就是对于各种题目三个队员都能做，但是又各有侧重。 要保证出来一道题能够有人会做、敢做，至少也要知道做法。

* 比赛周忌讳学习新的算法
* 比赛周忌讳连续数日的高强度训练
* 比赛周推荐阅览 M67 等博客，翻阅小学奥数题目以保持思维活跃打开思路

赛时：

********可以紧张，杜绝慌张，慌张是出题的敌人，任何时候，如果发现自己或者队友出现慌张的情况，提醒深呼吸。**

除了签到题，尽量每个人都看，**并留下自己一眼可见的观察**。
具体而言：如果一个人**五分钟**不会做一道题，必须立即找队友讨论。如果**再过十分钟**还是不会，必须找第三个队友。如果队友在忙，**必须**把这道题放下，转而去别的题留下观察。

其他队友有题能写，如果当前 Coder 占着键盘发呆超过 3 分钟请理解滚下键盘，说明他根本没有想好那个题目。（离开键盘同时记得打印代码）

1. 交完每道题都要先**打印**，**不要**等评测结果，直接写下一题。
2. 比赛时发的饭不是让你当时就吃的，那是给你**赛后吃的**。
3. 减少代码复制粘贴，尽量写成函数方便修改
4. 时空效率危险且代码量大的程序完成优先度较低
5. 英语不好，看不懂的，要**勤查词典**，懒一次就少一道题。
6. 照着纸敲代码和 sample 数据时不要敲错，特别**注意文字信息。**
7. **第一道简单题交给队中最稳的人做**，万一遇到麻烦也不要慌，如果有很多队都出了就更不必着急了，它必定是简单题，必定是可以很快做出来的，晚几分钟也比罚掉 20 分好。
8. 最后一小时是出题高峰，谁松懈，谁落后。最后一小时出一道是正常，出两道更好。
9. 无论是否有人通过，所有题必须全读过，最好每道题都有两人以上读过，尽量杜绝讲题现象。要完全弄清题意，正确的判断出题目的难易，不要想当然。
10. 觉得一道题适合其他人做就转手。
11. 保持头脑灵活，在正常方法不行时想想歪门邪道，比如换种不常见的特殊的数据结构，加预处理，限时搜索等。效率是第一位的，如果觉得 DP 麻烦就用记忆化搜索，总之考虑清楚后就要在最短时间出题。
12. 竞赛中更需要比平时稳定，程序出来后要检查重点地方，尽量 1Y。对于 WA 的题，**不要改一处就交**，很可能还有错的地方，要稳，要懂得在压力下也要仔细。对 WA 的题测试时要完整，必须每个点都测到，但不一定特别复杂。要考虑到测试的各种边界情况，比如矩阵可能为 1*1 或 1*n 或 m*1。

11.除非做出的人很多，否则**最后**考虑复杂几何题，精度造成的问题太多了

12. **块复制要小心**，检查相应的部分是否已经正确修改。
13. **纸上写程序要尽量完整**，每道题上机时间（包括输入、测试和调试）不要超过一小时。程序出错如果一时无法排除就应该打印出来阅读而把机器让出来。
14. 尽可能想到题目可以用到的数学的东西。
15. 实在迫不得已才可换人做题。
16. 如果遇到卡水题（全场通过队伍数高于 60%的题目，当然这个是根据你的队伍的强度来决定的）超过 1h 的情况，可以尝试请求队伍重炼（不要沿用任何你的代码）
17. 没有思路或者头晕的时候可以尝试去洗手间洗脸以及在赛场上站立一会儿
18. 非签构造题往往需要很长时间去思考，在没有非常多队通过的情况下不要去开。

事件	Check List
卡题	不要急，通过队伍多的题不是难题，不要想得太复杂了 1.重新仔细阅读一遍题面，仔细思考每个可能有用的点（数据范围）。 2.尝试倒着思考 3.在题单上写下已有想法 4.寻求队友帮助，**不要转述题意！！！** 5.感觉对就多试试样例
提交前	恭喜你马上就要通过这一题了，但是不要太大意 1.手模至少3组样例，检查是否正确，样例须包含边界情况（花费不到5min，不要懒） 2.检查需要初始化的变量，数组大小。 3.时间空间复杂度能否优化 4.检查调试信息是否删除。 5.提交代码 6.打印代码
AC	恭喜，快去看下一题
WA	祝贺你有一份可以通过样例的代码了，这代表你距离通过不远了，但千万别大意 1.手模至少5组样例，检查是否正确，样例须包含边界情况 2.检查是否爆long long __int128 3.检查需要初始化的变量。 4.检查调试信息是否删除。 5.检查数组横纵是否反了 6.排序是从小到大还是从大到小 7.检查输出是否与题目要求匹配，数据范围，前导零 8.多对拍 9.重读一边题意，仔细读，不要以为自己懂了。 10.找队友出点样例 11.小黄鸭调试法，仔细关注变量是否打错。
TLE	祝贺你，你离通过不远了，这个时候千万别大意 1.优先检查是否存在死循环 2.检查复杂度是否正确 3.检查输入数据是否和题目匹配 4.检查是否可以将map，set用一些线性数据结构代替 5.调试信息有没有删除 6.有没有能预处理优化的时间复杂度 7.小黄鸭调试法，仔细关注变量是否打错。

RE　　祝贺你，你离通过不远了，这个时候千万别大意

1.检查数组是否开小
2.检查是否有除 0，负数开根号
3.检查输入数据是否和题目匹配
4.检查数组是否开太大（MLE 返回 RE）
5.检查循环边界
6.小黄鸭调试法，仔细关注变量是否打错。

2024.11.15

一、常用

头文件

```
/*

_/      _/    _/      _/    _/      _/   _/_/_/_/_/     _/_/       _/ _/
 _/   _/     _/      _/     _/    _/        _/       _/    _/     _/ _/
  _/  _/      _/      _/      _/  _/          _/      _/      _/    _/_/ _/_/
   _/_/       _/_/_/_/_/        _/            _/      _/      _/    _/  _/ _/
  _/  _/      _/      _/          _/            _/      _/      _/    _/ _/
```

```
  _/    _/     _/       _/         _/            _/        _/    _/      _/   _/
 _/       _/    _/       _/         _/            _/          _/_/         _/   _/

*/
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
#define rep(i,a,n) for(int i=a;i<n;i++)
#define per(i,a,n) for(int i=n-1;i>=a;i--)
#define fastio ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define multi int _;cin>>_;while(_--)
#define debug(x) cerr << #x << " = " << (x) << endl;
#define int long long
#define pb push_back
#define eb emplace_back
ll gcd(ll a,ll b){ return b?gcd(b,a%b):a;}
mt19937_64 mrand(chrono::steady_clock().now().time_since_epoch().count());
int rnd(int l,int r){ return mrand() % (r - l + 1) + l;}
void test() {cerr << "\n";}
template<typename T, typename... Args>
void test(T x, Args... args) {cerr << x << " ";test(args...);}
const ll MOD = 998244353;
// const ll MOD = 1e9+7;
ll ksm(ll x,ll y){ll ans=1;x%=MOD;while(y){if(y&1)ans=ans*x%MOD;x=x*x%MOD,y/=2;}return ans;}

const int P1 = 972152273, base1 = 809;
const int P2 = 905563261, base2 = 919;
const ll N = 200005;
//head



signed main()
{
#ifdef localfreopen
    // freopen("1.in","r",stdin);
#endif
    fastio

    return 0;
}
```

预编译优化命令

```
#pragma GCC optimize("O3,unroll-loops")
//这行告诉GCC编译器使用O3优化级别和循环展开。O3是GCC提供的最高优化级别，它会尝试使用所有的程序优化策略。"unroll-loops"是一个特定的优化选项，它会尝试将循环展开以减少循环的开销。

#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")

```

```
//这行告诉GCC编译器生成的代码应该针对支持AVX2，BMI，BMI2，LZCNT和POPCNT指令集的CPU。这些都是特定的CPU指令集，可以提高代码的性能，但是生成的代码可能无法在不支持这些指令集的CPU上运行。
#pragma GCC optimize("Ofast")
#pragma GCC target("avx", "sse2")
#pragma GCC optimize("inline")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-funroll-loops")
#pragma GCC optimize("-fwhole-program")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-fstrict-overflow")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-skip-blocks")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("-funsafe-loop-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
```

快读

```
inline int read()
{
    int x=0,f=1;char ch=getchar();
    while (ch<'0'||ch>'9'){if (ch=='-') f=-1;ch=getchar();}
    while (ch>='0'&&ch<='9'){x=x*10+ch-48;ch=getchar();}
    return x*f;
}
```

对拍

```
:loop
 data.exe > 1.in
 my.exe <1.in >my.out
 std.exe <1.in >std.out
 fc my.out std.out
if not errorlevel 1 goto loop
pause
goto loop
```

__int128

```
__int128 read()
{
    __int128 f=1,w=0;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-')
        f=-1;
        ch=getchar();
    }
    while(ch<='9'&&ch>='0')
    {
        w=w*10+ch-'0';
        ch=getchar();
    }
    return f*w;
}

void print(__int128 x)
{
    if(x<0)
    {
        putchar('-');
        x=-x;
    }
    if(x>9)print(x/10);
    putchar(x%10+'0');
}
```

对拍(linux)

```bash
#!/bin/bash
while true; do
./data>1.in
./std<1.in>std.out
./my<1.in>my.out
if diff std.out my.out; then
printf AC
else
echo WA
exit 0
fi
sleep 1
done
```

checker

```bash
set -e
[ $# == 2 ] || { echo invalid args ; exit 1 ; }
compileg++ $2.cpp || { echo CE ; exit 1 ; }
src=./samples-$1
dir=$1-test
mkdir -p $dir
cp $src/* $dir/
cd $dir
mv ../a.out ./$2
for input in *.in; do
    [ $input == "*.in" ] && exit 0
    cas=${input%.in}
    output=$cas.out
    answer=$cas.ans
    timeout 1 ./$2 < $input > $output 2> $cas.err || { echo Case $cas : TLE or RE ; continue ; }
    if diff -ZA $output $answer > $cas.dif ; then
        echo Case $cas : AC
    else
        echo Case $cas : WA
        cat $cas.dif $cas.err
    fi
done
```

check(windows)

```cpp
#include<bits/stdc++.h>
using namespace std;

int main (int argc, char *argv[]) {
    std::string s = argv[1];
    for (int i = 1; i <= 100; ++i) {
        string in = std::to_string(i) + ".in";
```

```
        string out = std::to_string(i) + ".out";
        system(("std.exe <" + s + "/" + in + " >my.out").c_str());
        std::cout << "std.exe <" + s + "/" + in + " >my.out" << "\n";
        string a = "fc my.out " + s + "/" + out + "";
        std::cout << a << "\n";
        int ans = system(a.c_str());
        if (ans == 0) {
            cout << "Test case " << i << ": AC" << endl;
        } else {
            cout << "Test case " << i << ": WA" << endl;
            break;
        }
    }
    system("pause");
    return 0;
}

```

check(linux)

```
#include<bits/stdc++.h>

using namespace std;

int main (int argc, char *argv[]) {
    std::string s = argv[1];
    for (int i = 1; i <= 100; ++i) {
        string in = std::to_string(i) + ".in";
        string out = std::to_string(i) + ".out";
        system(("./" + s + " <" + s + "_samples/" + in + " >my.out").c_str());
        string a = "diff my.out " + s + "_samples/" + out + "";
        int ans = system(a.c_str());
        if (ans != 0) {
            cout << "Test case " << i << ": WA" << endl;
            break;
        } else {
            cout << "Test case " << i << ": AC" << endl;
        }
    }

    return 0;
}

```

builtin函数

```
__builtin_ctz( ) / __buitlin_ctzll( )
返回括号内数的二进制表示数末尾0的个数
__buitlin_clz( ) / __buitlin_clzll( )
用法:返回括号内数的二进制表示数前导0的个数
__builtin_popcount( )
```

```
用法:返回括号内数的二进制表示数1的个数
__builtin_parity( )
用法:判断括号中数的二进制表示数1的个数的奇偶性(偶数返回0 , 奇数返回1)
__builtin_ffs( )
用法:返回括号中数的二进制表示数的最后一个1在第几位(从后往前算)
__builtin_sqrt( ) 8位
__builtin_sqrtf( ) 4位
用法:快速开平方, 需要硬件有浮点支持, 能快10倍
__builtin_abs( )
__builtin_fabs( )
__builtin_powi( )
__builtin_memset( )
__builtin_memcpy( )
__builtin_strlen( )
__builtin_sin( )
__builtin_cos( )
__builtin_tan( )
```

二、字符串

kmp

```
vector<int> kmp(string s)
{//string的形式为'#' + t1 + '#' + s
    int n = s.size() - 1;
    vector<int> nxt(s.size());
    int j = 0;
    for(int i = 2 ; i <= n ; i++ ){
        while(j && s[j + 1] != s[i]) j = nxt[j];
        if(s[j + 1] == s[i]) j++;
        nxt[i] = j;
    }
    return nxt;
}//从第lent + 2 位 到 lent + lens + 1位为 s
```

manacher

```
vector<int> manacher(string s)
{//string为#A#B#C#...#Z#
    int n = s.size();
    vector<int> d1(n);
    for (int i = 0, l = 0, r = -1; i < n; i++)
    {
        int k = (i > r) ? 1 : min(d1[l + r - i], r - i + 1);
        while (0 <= i - k && i + k < n && s[i - k] == s[i + k])  k++;
        d1[i] = k--;
        if (i + k > r)
        {
            l = i - k;
```

```
            r = i + k;
        }
    }
    return d1;
}
```

最小表示法

```
string minrep(string s)
{//s从s[0]开始存
    int k = 0, i = 0, j = 1, n = s.size();
    while (k < n && i < n && j < n) {
        if (s[(i + k) % n] == s[(j + k) % n]) {
            k++;
        } else {
            s[(i + k) % n] > s[(j + k) % n] ? i = i + k + 1 : j = j + k + 1;
            if (i == j) i++;
            k = 0;
        }
    }
    i = min(i, j);
    return s.substr(i, N) + s.substr(0, i);
}
```

Z函数

```
vector<int> exkmp(string s)
{
    vector<int> p(s.size());
    int n = s.size() - 1;
    int L = 1, R = 0;
    p[1] = 0;
    for(int i = 2 ; i <= n ; i++ )
    {
        if(i > R)
        {
            p[i] = 0;
        }else{
            int k = i - L + 1;
            p[i] = min(p[k], R - i + 1);
        }
        while(i + p[i] <= n && s[p[i] + 1] == s[i + p[i]])
        {
            ++p[i];
        }
        if(i + p[i] - 1 > R)
        {
            L = i;
            R = i + p[i] - 1;
        }
    }
    return p;
}//从lent + 2位到lent + lens + 1位为 s
```

```
//******p[1] = 0，但实际从第一位往后能匹配lent的总长
```

AC自动机

```
struct ACautomaton {
    vector<vector<int>> nxt, end;
    vector<int> fail;
    int vtot = 0;
    ACautomaton() : nxt(1, vector<int>(26, 0)), end(1), fail(1){

    }
    ACautomaton(vector<string> ss){
        ACautomaton();
        for (auto s : ss) {
            insert(s);
        }
        buildfail();
    }
    int newnode() {
        int cur = ++vtot;
        nxt.push_back(vector<int>(26, 0));
        end.push_back(vector<int>(0));
        fail.emplace_back(0);
        return cur;
    }
    void insert(string s, int id = 0) {
        int now = 0;
        for (auto c : s) {
            int x = c - 'a';
            if (!nxt[now][x]) {
                nxt[now][x] = newnode();
            }
            now = nxt[now][x];
        }
        end[now].emplace_back(id);
    }
    void buildfail() {
        queue<int> q;
        for (int i = 0; i <= 25; i++) {
            if (nxt[0][i]) {
                fail[nxt[0][i]] = 0;
                q.push(nxt[0][i]);
            }
        }
        while (!q.empty()) {
            int now = q.front();
            q.pop();
            for (int i = 0; i <= 25; i++) {
                if (nxt[now][i]) {
                    fail[nxt[now][i]] = nxt[fail[now]][i];
                    q.push(nxt[now][i]);
                } else {
                    nxt[now][i] = nxt[fail[now]][i];
                }
            }
```

```
        }
    }
    int query(string s) {
        int now = 0, ans = 0;
        for (int i = 0; i < s.size(); i++) {
            char c = s[i];
            int x = c - 'a';
            now = nxt[now][x];
            ///自定义
        }
        return ans;
    }
};// root = 0, ***记得buildfail
```

SA(nlogn)

```
sa[i]：排名为i的后缀的位置
rk[i]：第i个位置开始的后缀的排名，作为基数排序的第一关键字
struct SA{
    vector<int> sa, rk, oldrk, id, key1, cnt, ht;
    vector<vector<int>> st;
    int i, m = 127, p, w;
    bool cmp(int x, int y, int w) {
        return oldrk[x] == oldrk[y] && oldrk[x + w] == oldrk[y + w];
    }// key1[i] = rk[id[i]]（作为基数排序的第一关键字数组）
    int n;
    SA(string s)
    {
        n = s.size() - 1;
        oldrk.resize(2 * n + 5);
        sa.resize(n + 2);
        rk.resize(n + 2);
        id.resize(n + 2);
        key1.resize(n + 2);
        cnt.resize(max(n + 5, 130ll));
        for (i = 1; i <= n; ++i) ++cnt[rk[i] = s[i]];
        for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
        for (i = n; i >= 1; --i) sa[cnt[rk[i]]--] = i;
        for (w = 1;; w <<= 1, m = p) {  // m=p 就是优化计数排序值域
            for (p = 0, i = n; i > n - w; --i) id[++p] = i;
            for (i = 1; i <= n; ++i)
                if (sa[i] > w) id[++p] = sa[i] - w;
            fill(cnt.begin(), cnt.end(), 0);
            for (i = 1; i <= n; ++i) ++cnt[key1[i] = rk[id[i]]];
            // 注意这里px[i] != i，因为rk没有更新，是上一轮的排名数组

            for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
            for (i = n; i >= 1; --i) sa[cnt[key1[i]]--] = id[i];
            for(int i = 1 ; i <= n ; i++)
            {
                oldrk[i] = rk[i];
            }
            for (p = 0, i = 1; i <= n; ++i)
```

```
                rk[sa[i]] = cmp(sa[i], sa[i - 1], w) ? p : ++p;
            if (p == n) {
                break;
            }
        }
        // height数组构建
        ht.resize(n + 2);
        int k = 0;
        for(int i = 1 ; i <= n ; i++ )
        {
            k = max(k - 1, 0ll);
            if(rk[i] == 1) continue;
            int j = sa[rk[i] - 1];
            while(s[i + k] == s[j + k]) k++;
            ht[rk[i]] = k;
        }

        // LCPst表构建
        st.resize(24);
        st[0].resize(n + 5);
        for(int i = 1 ; i <= n ; i++ )
        {
            st[0][i] = ht[i];
        }
        for(int j = 1 ; j <= 22 ; j++ )
        {
            st[j].resize(n + 5);
            for(int i = 1 ; i + (1 << j) - 1 <= n ; i++ )
            {
                st[j][i] = min(st[j - 1][i], st[j - 1][i + (1ll << j - 1)]);
            }
        }
    }
    int LCP(int u, int v)
    {
        if(u == v) return n - u + 1;
        if(rk[u] > rk[v]) swap(u, v);
        int l = rk[u] + 1, r = rk[v];
        int len = __lg(r - l + 1);
        return min(st[len][l], st[len][r - (1 << len) + 1]);
    }
};
//字符串存在1~n
//如果要用vector<int>. 记得离散化
// sa[i] 表示字典序第 i 小的后缀起始点在sa[i]
// rk[i] 表示后缀起点在 i 的字符串字典序排 rk[i]
```

SA(offline)

```
// sa[i]：排名为i的后缀的位置
// rk[i]：第i个位置开始的后缀的排名，作为基数排序的第一关键字
// tp[i]：第二关键字中，排名为i的数的位置
// cnt[i]：有多少个元素排名为i
// s[i]：原输入数组

```

```cpp
// init: s[1..n], n = strlen(s + 1), m = SIGMA, makesa()
const int N = 1E6 + 5;
#define rep(i, s, t) for (int i = s; i <= t; ++i)
#define per(i, s, t) for (int i = t; i >= s; --i)

int m = 256;
char s[N];
int n, sa[N], rk[N], tp[N], cnt[N];
void init() {
    rep(i, 1, n) rk[i] = s[i], tp[i] = i;
}
void Qsort() {
    rep(i, 1, m) cnt[i] = 0;
    rep(i, 1, n) ++ cnt[rk[i]];
    rep(i, 1, m) cnt[i] += cnt[i - 1];
    per(i, 1, n) sa[cnt[rk[tp[i]]] --] = tp[i];
}
void get_sort() {
    for(int w = 1, p = 0; w <= n; m = p, p = 0, w <<= 1) {
        rep(i, n - w + 1, n) tp[++ p] = i;
        rep(i, 1, n) if(sa[i] > w) tp[++ p] = sa[i] - w;
        Qsort(), swap(rk, tp), p = rk[sa[1]] = 1;
        rep(i, 2, n) rk[sa[i]] = (tp[sa[i]] == tp[sa[i - 1]]
                                 && tp[sa[i] + w] == tp[sa[i - 1] + w]) ? p : ++ p;
        if(p == n) return;
    }
}
void makesa() {
    init();Qsort();get_sort();
}
int ht[N];
void makeht() {
    for(int k = 0, i = 1; i <= n; i++) {
        k = max(k - 1, 0);
        if(rk[i] == 1) continue;
        int j = sa[rk[i] - 1];
        while(s[i + k] == s[j + k]) k++;
        ht[rk[i]] = k;
    }
}


int st[25][N];
void makest() {
    rep(i, 1, n) st[0][i] = ht[i];
    rep(j, 1, 22) {
        for (int i = 1; i + (1 << j) - 1 <= n; i++) {
            st[j][i] = min(st[j - 1][i], st[j - 1][i + (1ll << j - 1)]);
        }
    }
}

int getLcp(int u, int v) {
    if(u == v) return n - u + 1;
    if(rk[u] > rk[v]) swap(u, v);
```

```
    int l = rk[u] + 1, r = rk[v];
    int len = __lg(r - l + 1);
    return min(st[len][l], st[len][r - (1 << len) + 1]);
}
```

SAIS

```
char str[1000010];
int n, a[2000100], sa[2000100], typ[2000100], c[1000100], p[2000100], sbuc[1000100], lbuc[1000100], name[1000100];
inline int islms(int *typ, int i)
{
    return !typ[i] && (i == 1 || typ[i - 1]);
}
int cmp(int *s, int *typ, int p, int q)
{
    do {
        if (s[p] != s[q]) return 1;
        p++;  q++;
    } while (!islms(typ, p) && !islms(typ, q));
    return (!islms(typ, p) || !islms(typ, q) || s[p] != s[q]);
}

void isort(int *s, int *sa, int *typ, int *c, int n, int m)
{
    int i;
    for (lbuc[0] = sbuc[0] = c[0], i = 1; i <= m; i++) {
        lbuc[i] = c[i - 1] + 1;
        sbuc[i] = c[i];
    }
    for (i = 1; i <= n; i++)
        if (sa[i]>1 && typ[sa[i] - 1])
            sa[lbuc[s[sa[i] - 1]]++] = sa[i] - 1;
    for (i = n; i >= 1; i--)
        if (sa[i]>1 && !typ[sa[i] - 1])
            sa[sbuc[s[sa[i] - 1]]--] = sa[i] - 1;
}

void build_sa(int *s, int *sa, int *typ, int *c, int *p, int n, int m)
{
    int i;
    for (i = 0; i <= m; i++) c[i] = 0;
    for (i = 1; i <= n; i++) c[s[i]]++;
    for (i = 1; i <= m; i++) c[i] += c[i - 1];
    typ[n] = 0;
    for (i = n - 1; i >= 1; i--)
        if (s[i]<s[i + 1]) typ[i] = 0;
        else if (s[i]>s[i + 1]) typ[i] = 1;
        else typ[i] = typ[i + 1];
    int cnt = 0;
    for (i = 1; i <= n; i++)
        if (!typ[i] && (i == 1 || typ[i - 1])) p[++cnt] = i;
    for (i = 1; i <= n; i++) sa[i] = 0;
```

```cpp
    for (i = 0; i <= m; i++) sbuc[i] = c[i];
    for (i = 1; i <= cnt; i++)
        sa[sbuc[s[p[i]]]--] = p[i];
    isort(s, sa, typ, c, n, m);
    int last = 0, t = -1, x;
    for (i = 1; i <= n; i++)
    {
        x = sa[i];
        if (!typ[x] && (x == 1 || typ[x - 1]))
        {
            if (!last || cmp(s, typ, x, last))
                name[x] = ++t;
            else name[x] = t;
            last = x;
        }
    }
    for (i = 1; i <= cnt; i++)
        s[n + i] = name[p[i]];
    if (t<cnt - 1) build_sa(s + n, sa + n, typ + n, c + m + 1, p + n, cnt, t);
    else
        for (i = 1; i <= cnt; i++)
            sa[n + s[n + i] + 1] = i;
    for (i = 0; i <= m; i++) sbuc[i] = c[i];
    for (i = 1; i <= n; i++) sa[i] = 0;
    for (i = cnt; i >= 1; i--)
        sa[sbuc[s[p[sa[n + i]]]]--] = p[sa[n + i]];
    isort(s, sa, typ, c, n, m);
}

int main()
{
    scanf("%s", str);
    n = strlen(str);
    int i;
    for (i = 1; i <= n; i++)
        a[i] = str[i - 1];
    a[++n] = 0;
    build_sa(a, sa, typ, c, p, n, 200);
    for (i = 2; i <= n; i++)
        printf("%d%s", sa[i], i<n ? " " : "\n");
    return 0;
}
```

SAIS

```cpp
void induced_sort(const vector<int> &vec, int val_range, vector<int> &SA, const vector<bool> &sl, const vector<int> &lms_idx) {
    vector<int> l(val_range, 0), r(val_range, 0);
    for (int c : vec) {
        if (c + 1 < val_range) ++l[c + 1];
        ++r[c];
    }
    partial_sum(l.begin(), l.end(), l.begin());
    partial_sum(r.begin(), r.end(), r.begin());
```

```cpp
    fill(SA.begin(), SA.end(), -1);
    for (int i = lms_idx.size() - 1; i >= 0; --i)
        SA[--r[vec[lms_idx[i]]]] = lms_idx[i];
    for (int i : SA)
        if (i >= 1 && sl[i - 1]) {
            SA[l[vec[i - 1]]++] = i - 1;
        }
    fill(r.begin(), r.end(), 0);
    for (int c : vec)
        ++r[c];
    partial_sum(r.begin(), r.end(), r.begin());
    for (int k = SA.size() - 1, i = SA[k]; k >= 1; --k, i = SA[k])
        if (i >= 1 && !sl[i - 1]) {
            SA[--r[vec[i - 1]]] = i - 1;
        }
}
vector<int> SA_IS(const vector<int> &vec, int val_range) {
    const int n = vec.size();
    vector<int> SA(n), lms_idx;
    vector<bool> sl(n);
    sl[n - 1] = false;
    for (int i = n - 2; i >= 0; --i) {
        sl[i] = (vec[i] > vec[i + 1] || (vec[i] == vec[i + 1] && sl[i +
1]));
        if (sl[i] && !sl[i + 1]) lms_idx.push_back(i + 1);
    }
    reverse(lms_idx.begin(), lms_idx.end());
    induced_sort(vec, val_range, SA, sl, lms_idx);
    vector<int> new_lms_idx(lms_idx.size()), lms_vec(lms_idx.size());
    for (int i = 0, k = 0; i < n; ++i)
        if (!sl[SA[i]] && SA[i] >= 1 && sl[SA[i] - 1]) {
            new_lms_idx[k++] = SA[i];
        }
    int cur = 0;
    SA[n - 1] = cur;
    for (size_t k = 1; k < new_lms_idx.size(); ++k) {
        int i = new_lms_idx[k - 1], j = new_lms_idx[k];
        if (vec[i] != vec[j]) {
            SA[j] = ++cur;
            continue;
        }
        bool flag = false;
        for (int a = i + 1, b = j + 1;; ++a, ++b) {
            if (vec[a] != vec[b]) {
                flag = true;
                break;
            }
            if ((!sl[a] && sl[a - 1]) || (!sl[b] && sl[b - 1])) {
                flag = !((!sl[a] && sl[a - 1]) && (!sl[b] && sl[b - 1]));
                break;
            }
        }
        SA[j] = (flag ? ++cur : cur);
    }
    for (size_t i = 0; i < lms_idx.size(); ++i)
        lms_vec[i] = SA[lms_idx[i]];
```

```
    if (cur + 1 < (int)lms_idx.size()) {
        auto lms_SA = SA_IS(lms_vec, cur + 1);
        for (size_t i = 0; i < lms_idx.size(); ++i) {
            new_lms_idx[i] = lms_idx[lms_SA[i]];
        }
    }
    induced_sort(vec, val_range, SA, sl, new_lms_idx);
    return SA;
}
template <class T>
vector<int> suffix_array(const T &s, const int LIM = 128) {
    vector<int> vec(s.size() + 1);
    copy(begin(s), end(s), begin(vec));
    vec.back() = 0 ;
    // vec.back() = '$';
    auto ret = SA_IS(vec, LIM);
    ret.erase(ret.begin());
    return ret;
}
vector<int> getRank(const vector<int> &sa) {
    vector<int> rk(sa.size());
    for (int i = 0 ; i < sa.size(); i++) {
        rk[sa[i]] = i;
    }
    return rk;
}
template <class T>
vector<int> getHeight(const T &s, const vector<int> &sa) {
    int n = s.size(), k = 0;
    vector<int> ht(n), rank(n);
    for (int i = 0; i < n; i++) rank[sa[i]] = i;
    for (int i = 0; i < n; i++, k ? k-- : 0) {
        if (rank[i] == n - 1) {
            k = 0;
            continue;
        }
        int j = sa[rank[i] + 1];
        while (i + k < n && j + k < n && s[i + k] == s[j + k]) ++ k;
        ht[rank[i] + 1] = k;
    }
    ht[0] = 0;
    return ht;
}
template <class T>
vector<vector<int>> buildLCP(const T &s, const vector<int> ht) {
    vector<vector<int>> st;
    int n = s.size() - 1;
    int LOG = __lg(n) + 1;
    st.resize(LOG);
    st[0].resize(n + 1);
    for(int i = 1 ; i <= n ; i++ )
    {
        st[0][i] = ht[i];
    }
    for(int j = 1 ; j <= LOG ; j++ )
    {
```

```
        st[j].resize(n + 1);
        for(int i = 1 ; i + (1 << j) - 1 <= n ; i++ )
        {
            st[j][i] = min(st[j - 1][i], st[j - 1][i + (1ll << j - 1)]);
        }
    }
    return st;
}
void use() {
    vector<vector<int>> st;
    vector<int> rk;
    int n;
    int u, v;
    function<int(int, int)> lcp = [&](int u, int v)
    {
        if(u == v) return n - u + 1;
        if(rk[u] > rk[v]) swap(u, v);
        int l = rk[u] + 1, r = rk[v];
        int len = __lg(r - l + 1);
        return min(st[len][l], st[len][r - (1 << len) + 1]);
    };
}
```

SAM

```
struct SuffixAutomaton
{
    int tot, last;
    vector<int> len, link, sz;
    vector<vector<int>> nxt;
    //vector<pii> order;
    int n;
    SuffixAutomaton(int _n) :n(_n), sz(2 * _n + 5), len(2 * _n + 5), link(2 * _n + 5), nxt(2 * _n + 5, vector<int>(33, 0))
    {
        len[1] = 0;
        link[1] = -1;
        nxt[1].clear();
        nxt[1].resize(33);
        tot = 2;
        last = 1;
    }
    void extend(int c)
    {
        int cur = tot++, p;
        len[cur] = len[last] + 1;
        nxt[cur].clear();
        nxt[cur].resize(33);
        for (p = last; p != -1 && !nxt[p][c]; p = link[p])
            nxt[p][c] = cur;
        if (p == -1) link[cur] = 1;
        else
        {
```

```
                int q = nxt[p][c];
                if (len[p] + 1 == len[q]) link[cur] = q;
                else
                {
                    int clone = tot++;
                    len[clone] = len[p] + 1;
                    link[clone] = link[q];
                    nxt[clone] = nxt[q];
                    for (; p != -1 && nxt[p][c] == q; p = link[p])
                        nxt[p][c] = clone;
                    link[q] = link[cur] = clone;
                }
            }
            last = cur;
            sz[cur] = 1;
        }
        vector<vector<int>> adj;
        void buildLinkTree()
        {
            adj.resize(tot + 1);
            for (int i = 2; i <= tot; i++ )
            {
                adj[link[i]].push_back(i);
            }
        }
};//sam的root为1
```

ExSAM

```
struct EXSAM
{
    const int CHAR_NUM = 30;   // 字符集个数，注意修改下方的 (-'a')
    int tot;                   // 节点总数：[0, tot)
    int n;
    vector<int> len, link;
    vector<vector<int>> nxt;
    EXSAM (int _n) : n(_n), len(_n * 2 + 5), link(_n * 2 + 5), nxt(n * 2 +
5, vector<int>(CHAR_NUM + 1, 0))
    {
        tot = 2;
        link[1] = -1;
    }
    int insertSAM(int last, int c)    // last 为父 c 为子
    {
        int cur = nxt[last][c];
        if (len[cur]) return cur;
        len[cur] = len[last] + 1;
        int p = link[last];
        while (p != -1)
        {
            if (!nxt[p][c])
                nxt[p][c] = cur;
            else
                break;
            p = link[p];
```

```
        }
        if (p == -1)
        {
            link[cur] = 1;
            return cur;
        }
        int q = nxt[p][c];
        if (len[p] + 1 == len[q])
        {
            link[cur] = q;
            return cur;
        }
        int clone = tot++;
        for (int i = 0; i < CHAR_NUM; ++i)
            nxt[clone][i] = len[nxt[q][i]] != 0 ? nxt[q][i] : 0;
        len[clone] = len[p] + 1;
        while (p != -1 && nxt[p][c] == q)
        {
            nxt[p][c] = clone;
            p = link[p];
        }
        link[clone] = link[q];
        link[cur] = clone;
        link[q] = clone;
        return cur;
    }

    int insertTrie(int cur, int c)
    {
        if (nxt[cur][c]) return nxt[cur][c];  // 已有该节点 直接返回
        return nxt[cur][c] = tot++;            // 无该节点 建立节点
    }

    void insert(const string &s)
    {
        int root = 1;
        for (auto ch : s) root = insertTrie(root, ch - 'a');
    }

    void insert(const char *s, int n)
    {
        int root = 1;
        for (int i = 0; i < n; ++i)
            root =
                insertTrie(root, s[i] - 'a');  // 一边插入一边更改所插入新节点的父节点
    }

    void build()
    {
        queue<pair<int, int>> q;
        for (int i = 0; i < 26; ++i)
            if (nxt[1][i]) q.push({i, 1});
        while (!q.empty())    // 广搜遍历
        {
            auto item = q.front();
```

```
            q.pop();
            auto last = insertSAM(item.second, item.first);
            for (int i = 0; i < 26; ++i)
                if (nxt[last][i]) q.push({i, last});
        }
    }
};
```

PAM

```
const int N = 5e5 + 10, Sigma = 26;
char s[N];
int lastans, n;
struct Palindrome_Automaton {
    int ch[N][Sigma], fail[N], len[N], sum[N], cnt, last;
    Palindrome_Automaton() {
        cnt = 1;
        fail[0] = 1, fail[1] = 1, len[1] = -1;
    }
    int getfail(int x, int i) {
        while(i - len[x] - 1 < 0 || s[i - len[x] - 1] != s[i]) x = fail[x];
        return x;
    }
    void insert(char c, int i) {
        int x = getfail(last, i), w = c - 'a';
        if(!ch[x][w]) {
            len[++cnt] = len[x] + 2;
            int tmp = getfail(fail[x], i);
            fail[cnt] = ch[tmp][w];
            sum[cnt] = sum[fail[cnt]] + 1;
            ch[x][w] = cnt;
        }
        last = ch[x][w];
    }
} PAM;
```

PAM(new)

```
struct PAM {
    int sz, tot, last;
    vector<int> cnt, len, fail;
    vector<vector<int>> ch;
    vector<char> s;
    PAM(int n) : cnt(n + 5), ch(n + 5, vector<int>(30)), len(n + 5), fail(n
+ 5), s(n + 5) {
        clear();
    }
    int node(int l) {     // 建立一个新节点，长度为 l
        sz++;
        ch[sz].assign(30, 0);
        len[sz] = l;
        fail[sz] = cnt[sz] = 0;
        return sz;
    }
```

```
    void clear() {   // 初始化
        sz = -1;
        last = 0;
        s[tot = 0] = '$';
        node(0);
        node(-1);
        fail[0] = 1;
    }
    int getfail(int x) {   // 找后缀回文
        while (s[tot - len[x] - 1] != s[tot]) x = fail[x];
        return x;
    }
    void insert(char c)    // 建树
    {
        s[++tot] = c;
        int now = getfail(last);
        if (!ch[now][c - 'a'])
        {
            int x = node(len[now] + 2);
            fail[x] = ch[getfail(fail[now])][c - 'a'];
            ch[now][c - 'a'] = x;
        }
        last = ch[now][c - 'a'];
        cnt[last]++;
    }
};
```

#

三、图论

dinic

```
const int V = 1010;
const int E = 101000;
using ll = long long;

template<typename T>
struct MaxFlow
{
    int s, t, vtot;
    int head[V], etot;
    int dis[V], cur[V];
    struct edge
    {
        int v, nxt;
        T f;
    }e[E * 2];
    void addedge(int u, int v, T f)
    {
        e[etot] = {v, head[u], f}; head[u] = etot++;
        e[etot] = {u, head[v], 0}; head[v] = etot++;
    }
    bool bfs()
```

```
    {
        for(int i = 1 ; i <= vtot ; i++ )
        {
            dis[i] = 0;
            cur[i] = head[i];
        }
        queue<int> q;
        q.push(s); dis[s] = 1;
        while(!q.empty())
        {
            int u = q.front(); q.pop();
            for(int i = head[u] ; ~i ; i = e[i].nxt)
            {
                if(e[i].f && !dis[e[i].v])
                {
                    int v = e[i].v;
                    dis[v] = dis[u] + 1;
                    if(v == t) return true;
                    q.push(v);
                }
            }
        }
        return false;
    }
    T dfs(int u, T m)
    {
        if(u == t) return m;
        T flow = 0;
        for(int i = cur[u]; ~i ; cur[u] = i = e[i].nxt)
        {
            if(e[i].f && dis[e[i].v] == dis[u] + 1)
            {
                T f = dfs(e[i].v, min(m, e[i].f));
                e[i].f -= f;
                e[i ^ 1].f += f;
                m -= f;
                flow += f;
                if(!m) break;
            }
        }
        if(!flow) dis[u] = -1;
        return flow;
    }
    T dinic()
    {
        T flow = 0;
        while(bfs()) flow += dfs(s, numeric_limits<T>::max());
        return flow;
    }
    void init(int s_, int t_, int vtot_ )
    {
        s = s_;
        t = t_;
        vtot = vtot_;
        etot = 0;
        for(int i = 1 ; i <= vtot ; i++ )
```

```
        {
            head[i] = -1;
        }
    }
};

MaxFlow<ll> g;
//***记得每次init,

```

费用流

```
const int V = 2010;
const int E = 20100;
// #define int double
using ll = long long;

template<typename T>
struct MaxFlow {
    int s, t, vtot;
    int head[V], etot, cur[V];
    int pre[V];
    bool vis[V];
    T dis[V], cost, flow;

    struct edge {
        int v, nxt;
        T f, c;
    }e[E * 2];

    void addedge(int u, int v, T f, T c, T f2 = 0)
    {
        e[etot] = {v, head[u], f, c}; head[u] = etot++;
        e[etot] = {u, head[v], f2, -c}; head[v] = etot++;
    }

    bool spfa() {
        T inf = numeric_limits<T>::max() / 2;
        for(int i = 1; i <= vtot; i++) {
            dis[i] = inf;
            vis[i] = false;
            pre[i] = -1;
            cur[i] = head[i];
        }
        dis[s] = 0;
        vis[s] = true;
        queue<int> q;
        q.push(s);
        while(!q.empty()) {
            int u = q.front();
            for(int i = head[u]; ~i; i = e[i].nxt) {
                int v = e[i].v;
                if(e[i].f && dis[v] > dis[u] + e[i].c) {
                    dis[v] = dis[u] + e[i].c;
                    pre[v] = i;
```

```cpp
                    if(!vis[v]) {
                        vis[v] = 1;
                        q.push(v);
                    }
                }
            }
            q.pop();
            vis[u] = false;
        }
        return dis[t] < inf;
    }

    void augment() {
        int u = t;
        T f = numeric_limits<T>::max();
        while(~pre[u]) {
            f = min(f, e[pre[u]].f);
            u = e[pre[u] ^ 1].v;
        }
        flow += f;
        cost += f * dis[t];
        u = t;
        while(~pre[u]) {
            e[pre[u]].f -= f;
            e[pre[u] ^ 1].f += f;
            u = e[pre[u] ^ 1].v;
        }
    }

    pair<T, T> sol() {
        flow = cost = 0;
        while(spfa()) {
            augment();
        }
        return {flow, cost};
    }

    void init(int s_, int t_, int vtot_ )
    {
        s = s_;
        t = t_;
        vtot = vtot_;
        etot = 0;
        for(int i = 1 ; i <= vtot ; i++ )
        {
            head[i] = -1;
        }
    }
};

//***记得每次init,
```

二分图最大匹配

```
int a[N];
int v[N], n1, n2;
int to[N], b[N];
int n;
vector<int> e[N];
//n1为左边点数量，n2为右边点数量，v为右边的点连向左边哪条边
bool find(int x)
{
    b[x] = true;
    for(auto y : e[x])
    {
        if(!v[y] || (!b[v[y]] && find(v[y])))
        {
            v[y] = x;
            return true;
        }
    }
    return false;
}

int match()
{
    int ans = 0;
    memset(v, 0 ,sizeof(v));
    for(int i = 1 ; i <= n1 ; i++ )
    {
        memset(b, 0, sizeof(b));
        if(find(i))
        {
            ++ans;
        }
    }
    return ans;
}
```

2—SAT—Tarjan

```
struct TwoSat {
    int n;
    std::vector<std::vector<int>> e;
    std::vector<bool> ans;
    TwoSat(int n) : n(n), e(2 * n), ans(n) {}
    void addClause(int u, bool f, int v, bool g) {
        e[2 * u + f].push_back(2 * v + g);
    }
    bool satisfiable() {
        std::vector<int> id(2 * n, -1), dfn(2 * n, -1), low(2 * n, -1);
        std::vector<int> stk;
        int now = 0, cnt = 0;
        std::function<void(int)> tarjan = [&](int u) {
            stk.push_back(u);
            dfn[u] = low[u] = now++;
```

```
            for (auto v : e[u]) {
                if (dfn[v] == -1) {
                    tarjan(v);
                    low[u] = std::min(low[u], low[v]);
                } else if (id[v] == -1) {
                    low[u] = std::min(low[u], dfn[v]);
                }
            }
            if (dfn[u] == low[u]) {
                int v;
                do {
                    v = stk.back();
                    stk.pop_back();
                    id[v] = cnt;
                } while (v != u);
                ++cnt;
            }
        };
        for (int i = 0; i < 2 * n; ++i) if (dfn[i] == -1) tarjan(i);
        for (int i = 0; i < n; ++i) {
            if (id[2 * i] == id[2 * i + 1]) return false;
            ans[i] = id[2 * i] > id[2 * i + 1];
        }
        return true;
    }
    std::vector<bool> answer() { return ans; }
};
```

SCC hosoraju

```
int vis[N], n, m;
vector<int> out, c, e[N], erev[N];
int sz[N];
int bel[N], cnt;
vector<vector<int> >scc;

void dfs1(int u)
{
    vis[u] = 1;
    for(auto v : e[u])
    {
        if(!vis[v]) dfs1(v);
    }
    out.push_back(u);
}

void dfs2(int u, int cnt)
{

    vis[u] = 1;
    for(auto v : erev[u])
    {
        if(!vis[v]) dfs2(v, cnt);
    }
    bel[u] = cnt;
```

```cpp
    sz[cnt]++;
    c.push_back(u);
}

int main()
{
    fastio
    //freopen("1.in","r",stdin);
    int n, m, x, y;
    cin >> n >> m;
    for(int i = 1 ; i <= m ; i++ )
    {
        cin >> x >> y;
        e[x].push_back(y);
        erev[y].push_back(x);
    }
    memset(vis, 0, sizeof(vis));
    for(int i = 1 ; i <= n ; i++ )
    {
        if(!vis[i])
        {
            dfs1(i);
        }
    }
    reverse(out.begin(), out.end());
    memset(vis, 0, sizeof(vis));
    for(auto u : out)
    {
        if(!vis[u])
        {
            c.clear();
            dfs2(u, ++cnt);
            sort(c.begin(), c.end());
            scc.push_back(c);
        }

    }
    sort(scc.begin(), scc.end());
    for(auto c : scc)
    {
        for(auto x : c)
        {
            cout << x << " ";
        }
        cout << "\n";
    }
    return 0;
}
```

SCC Tarjan

```cpp
struct SCC {
    int n;
    std::vector<std::vector<int>> adj;
    std::vector<int> stk;
```

```cpp
    std::vector<int> dfn, low, bel;
    int cur, cnt;

    SCC() {}
    SCC(int n) {
        init(n);
    }

    void init(int n) {
        this->n = n;
        adj.assign(n + 1, {});
        dfn.assign(n + 1, -1);
        low.resize(n + 1);
        bel.assign(n + 1, -1);
        stk.clear();
        cur = cnt = 0;
    }

    void addEdge(int u, int v) {
        adj[u].push_back(v);
    }

    void dfs(int x) {
        dfn[x] = low[x] = cur++;
        stk.push_back(x);

        for (auto y : adj[x]) {
            if (dfn[y] == -1) {
                dfs(y);
                low[x] = std::min(low[x], low[y]);
            } else if (bel[y] == -1) {
                low[x] = std::min(low[x], dfn[y]);
            }
        }

        if (dfn[x] == low[x]) {
            int y;
            ++cnt;
            do {
                y = stk.back();
                bel[y] = cnt;
                stk.pop_back();
            } while (y != x);
        }
    }

    std::vector<int> work() {
        for (int i = 1; i <= n; i++) {
            if (dfn[i] == -1) {
                dfs(i);
            }
        }
        return bel;
    }
};
```

边双连通分量

```
int head[N], e[N], nxt[N], idx = 1, n, m;
int dfn[M], low[M], cnt, b[N], bel[N], anscnt[M];
vector<vector<int> > dcc;
void add(int x, int y)
{
    nxt[++idx] = head[x];
    head[x] = idx;
    e[idx] = y;
}
void tarjan(int x, int e_in)
{
    dfn[x] = low[x] = ++cnt;
    for(int i = head[x] ; i ; i = nxt[i])
    {
        int y = e[i];
        if(!dfn[y])
        {
            tarjan(y, i);
            if(dfn[x] < low[y])
            {
                b[i] = b[i ^ 1] = 1;
            }
            low[x] = min(low[x], low[y]);
        }else if (i != (e_in ^ 1))
        {
            low[x] = min(low[x], dfn[y]);
        }
    }
}

vector<int> v;

void dfs(int x, int cnt)
{
    bel[x] = cnt;
    v.push_back(x);
    anscnt[cnt]++;
    for(int i = head[x] ; i ; i = nxt[i])
    {
        int y = e[i];
        if(bel[y] || b[i]) continue;
        dfs(y, cnt);
    }

}
signed main()
{
    fastio
    //freopen("1.in","r",stdin);
    cin >> n >> m;
    int x, y;
    for(int i = 1 ; i <= m ; i++ )
    {
```

```
        cin >> x >> y;
        if(x == y) continue;
        add(x, y);
        add(y, x);
    }
    for(int i = 1 ; i <= n ; i++ )
    {
        if(!dfn[i]) tarjan(i, 0);
    }
    int ans = 0;
    for(int i = 1 ; i <= n ; i++ )
    {
        if(!bel[i])
        {
            v.clear();
            dfs(i, ++ans);
            dcc.push_back(v);
        }

    }
    int sz = dcc.size();
    cout << dcc.size() << "\n";
    for(int i = 0 ; i < sz ; i++ )
    {
        auto v = dcc[i];
        cout << anscnt[i + 1] << " ";
        for(auto x : v)
        {
            cout << x << " ";
        }
        cout << "\n";
    }
    return 0;
}
```

割点

```
struct CutPoint {
    int n, m, idx;
    std::vector<int> dfn, low, vis, cut;
    std::vector<std::vector<int>> adj;
    CutPoint(int _n, int _m) : n(_n), m(_m), dfn(_n + 1),
    low(_n + 1), vis(_n + 1), cut(_n + 1), adj(_n + 1) {

    }

    void dfs(int x, int root) {
        vis[x] = 1;
        dfn[x] = ++idx;
        low[x] = idx;
        int child = 0;
        for (auto y : adj[x]) {
            if (!vis[y]) {
                dfs(y, root);
                low[x] = std::min(low[x], low[y]);
```

```
                if (low[y] >= dfn[x] && x != root) {
                    cut[x] = 1;
                }
                if (x == root) {
                    child++;
                }
            }
            low[x] = std::min(low[x], dfn[y]);
        }
        if (child >= 2 && x == root) {
            cut[x] = 1;
        }
    }

    std::vector<int> work() {
        std::vector<int> q;
        for (int i = 1; i <= n; i++) {
            if (!vis[i]) {
                dfs(i, i);
            }
        }
        for (int i = 1; i <= n; i++) {
            if (cut[i]) {
                q.push_back(i);
            }
        }
        return q;
    }

    void addEdge(int u, int v) {
        adj[u].push_back(v);
    }
};
```

割边

```
struct CutEdges {
    int n;
    int idx = 0;
    vector<int> low, dfn, fa;
    vector<int> head, nxt, to;
    vector<int> b;
    int iddx = 1;
    vector<pair<int,int>> bridge;
    CutEdges(int n, int m) : low(n + 1), dfn(n + 1), fa(n + 1),
    head(n + 1), to(2 * m + 4), nxt(2 * m + 4), b(2 * m + 4) {
        this->n = n;
    }
    void addEdge(int x, int y) {
        nxt[++iddx] = head[x];
        head[x] = iddx;
        to[iddx] = y;
    }
    vector<pair<int, int>> work() {
        for (int i = 1; i <= n; i++) {
```

```
            if (!dfn[i]) tarjan(i, 0);
        }
        return bridge;
    }
    void tarjan(int x, int e_in) {;
        dfn[x] = low[x] = ++idx;
        for(int i = head[x]; i; i = nxt[i]) {
            int y = to[i];
            if(!dfn[y]) {
                tarjan(y, i);
                if(dfn[x] < low[y]) {
                    bridge.push_back({x, y});
                    b[i] = b[i ^ 1] = 1;
                }
                low[x] = min(low[x], low[y]);
            } else if (i != (e_in ^ 1)) {
                low[x] = min(low[x], dfn[y]);
            }
        }
    }

};
CutEdges g(n, m);
```

无向图欧拉图

```
vector<pair<int ,int > > e[N];
int d[N], n, m;
int f[N], b[N], sz[N], ans[N], idxans;

void dfs(int x)
{
    //cout << "dfs = " << x << endl;
    for(; f[x] < sz[x] ; )
    {
        int y = e[x][f[x]].first, id = e[x][f[x]].second;
        if(!b[id])
        {
            b[id] = 1;
            f[x]++;
            dfs(y);
            ans[++idxans] = y;
        }else{
            f[x]++;
        }
    }
}

void Euler()
{
    memset(f, 0, sizeof(f));
    memset(b, 0 ,sizeof(b));
    int cnt = 0, x = 0;
```

```
    for(int i = 1 ; i <= n ; i++ )
    {
        if(d[i] & 1)
        {
            cnt++;
            x = i;
        }
    }
    if(!(cnt == 0 || cnt == 2))
    {
        cout << "No\n";
        return;
    }
    for(int i = 1 ; i <= n ; i++ )
    {
        sz[i] = e[i].size();
        if(!x)
            if(d[i])
            {
                x = i;
            }
    }
    dfs(x);
    ans[++idxans] = x;
    if(idxans == m + 1)
    {
        cout << "Yes\n";
    }else{
        cout << "No\n";
    }
}
int main()
{
    fastio
    //freopen("1.in","r",stdin);
    cin >> n >> m;
    int idx = 0;
    for(int i = 1 ; i <= m ; i++ )
    {
        int x, y;
        cin >> x >> y;
        ++idx;
        ++d[x];
        ++d[y];
        e[x].push_back({y, idx});
        e[y].push_back({x, idx});

    }
    Euler();
    return 0;
}
```

有向图欧拉图

```
int n;
```

```
vector<int> e[N];
int ind[N], outd[N], f[N], sz[N], ans[N], idx = 0;

void dfs(int x)
{
    for(; f[x] < sz[x] ;)
    {
        int y = e[x][f[x]];
        f[x]++;
        dfs(y);
        ans[++idx] = y;
    }
}
void Euler()
{
    memset(f, 0, sizeof(f));
    int cntdiff = 0;
    int cntin = 0;
    int x = 0;
    for(int i = 1 ; i <= n ; i++ )
    {
        if(ind[i] != outd[i])
        {
            cntdiff++;
        }
        if(ind[i] + 1 == outd[i])
        {
            cntin++;
            x = i;
        }
    }
    if(!(cntdiff == 2 && cntin == 1 || cntdiff == 0))
    {
        cout << "No\n";
        return;
    }
    for(int i = 1 ; i <= n ; i++ )
    {
        sz[i] = e[i].size();
        //cout << e[i].size();
        if(!x)
        {
            if(ind[i])
            {
                x = i;
            }
        }
    }
    dfs(x);
    ans[++idx]= x;
    if(idx == n + 1)
    {
        cout << "Yes\n";
    }else{
        cout << "No\n";
    }
```

```
    for(int i = idx ; i > 0 ; i--)
    {
        cout << ans[i] << " ";
    }
}
```

笛卡尔树

```
//每个父节点都小于其所有子节点

int a[N], n, l[N], r[N];
int root = 0;

void build()
{
    stack<int> st;
    for(int i = 1 ; i <= n ; i++ )
    {
        int last = 0;
        while(!st.empty() && a[st.top()] > a[i])
        {
            last = st.top();
            st.pop();
        }
        if(!st.empty())
        {
            r[st.top()] = i;
        }else{
            root = i;
        }
        l[i] = last;
        st.push(i);
    }
}
```

dfs序求lca

```
int main()
{
    int idx = 0;
    vector<int> dfn(n + 5);
    vector st(__lg(n) + 2, vector<int> (n + 5));//****不能改成23****
    function<int(int,int)> get = [&](int x, int y)
    {
        return dfn[x] < dfn[y] ? x : y;
    };
    function<void(int,int)> dfs = [&](int x, int fa)
    {
        st[0][dfn[x] = ++idx] = fa;
        for(int y : adj[x]) if(y != fa) dfs(y, x);
    };
    function<int(int,int)> lca = [&](int u, int v)
```

```
    {
        if(u == v) return u;
        if((u = dfn[u]) > (v = dfn[v])) swap(u, v);
        int d = __lg(v - u++);
        return get(st[d][u], st[d][v - (1 << d) + 1]);
    };
    dfs(s, 0);
    for(int i = 1 ; i <= __lg(n) ; i++ )//****不能改成23****
    {
        for(int j = 1 ; j + (1 << i - 1) <= n  ; j++ ) // ****注意边界****
        {
            st[i][j] = get(st[i - 1][j], st[i - 1][j + (1 << i - 1)]);
        }
    }
    /// lca(u, v);
}
```

HLD

```
using i64 = long long;
struct HLD {
    int n;
    std::vector<int> siz, top, dep, parent, in, out, seq;
    std::vector<std::vector<int>> adj;
    int cur;

    HLD() {}
    HLD(int n) {
        init(n);
    }
    void init(int n) {
        this->n = n;
        siz.resize(n + 1);
        top.resize(n + 1);
        dep.resize(n + 1);
        parent.resize(n + 1);
        in.resize(n + 1);
        out.resize(n + 1);
        seq.resize(n + 1);
        cur = 1;
        adj.assign(n + 1, {});
    }
    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    void work(int root = 1) {
        top[root] = root;
        dep[root] = 0;
        parent[root] = -1;
        dfs1(root);
        dfs2(root);
    }
    void dfs1(int u) {
        if (parent[u] != -1) {
```

```
            adj[u].erase(std::find(adj[u].begin(), adj[u].end(),
parent[u]));
        }

        siz[u] = 1;
        for (auto &v : adj[u]) {
            parent[v] = u;
            dep[v] = dep[u] + 1;
            dfs1(v);
            siz[u] += siz[v];
            if (siz[v] > siz[adj[u][0]]) {
                std::swap(v, adj[u][0]);
            }
        }
    }
    void dfs2(int u) {
        in[u] = cur++;
        seq[in[u]] = u;
        for (auto v : adj[u]) {
            top[v] = v == adj[u][0] ? top[u] : v;
            dfs2(v);
        }
        out[u] = cur;
    }
    int lca(int u, int v) {
        while (top[u] != top[v]) {
            if (dep[top[u]] > dep[top[v]]) {
                u = parent[top[u]];
            } else {
                v = parent[top[v]];
            }
        }
        return dep[u] < dep[v] ? u : v;
    }

    int dist(int u, int v) {
        return dep[u] + dep[v] - 2 * dep[lca(u, v)];
    }

    int jump(int u, int k) {
        if (dep[u] < k) {
            return -1;
        }

        int d = dep[u] - k;

        while (dep[top[u]] > d) {
            u = parent[top[u]];
        }

        return seq[in[u] - dep[u] + d];
    }

    bool isAncester(int u, int v) {
        return in[u] <= in[v] && in[v] < out[u];
    }
```

```cpp

    int rootedParent(int u, int v) {
        std::swap(u, v);
        if (u == v) {
            return u;
        }
        if (!isAncester(u, v)) {
            return parent[u];
        }
        auto it = std::upper_bound(adj[u].begin(), adj[u].end(), v, [&](int x, int y) {
            return in[x] < in[y];
        }) - 1;
        return *it;
    }

    int rootedSize(int u, int v) {
        if (u == v) {
            return n;
        }
        if (!isAncester(v, u)) {
            return siz[v];
        }
        return n - siz[rootedParent(u, v)];
    }

    int rootedLca(int a, int b, int c) {
        return lca(a, b) ^ lca(b, c) ^ lca(c, a);
    }
};
```

点分治

```cpp
signed main()
{
    fastio
    int n, k, ans = 0;
    cin >> n >> k;
    ans = n + 1;
    vector<vector<pair<int,int>>> adj(n + 1);
    vector<int> sz(n + 1, 0), maxsz(n + 1, 0), del(n + 1, 0);
    vector<int> mark(k + 1, 0), c(k + 1, 0);
    int T = 1;
    int u, v, w;
    for(int i = 1 ; i < n ; i++ )
    {
        cin >> u >> v >> w;
        u++;
        v++;
        adj[u].emplace_back(v, w);
        adj[v].emplace_back(u, w);
    }
    function<void(int, int)> solve = [&](int x, int s)
```

```cpp
    {
        T++;
        int mxs = s + 1, root = -1;
        function<void(int, int)> dfs1 = [&](int x, int fx)
        {
            sz[x] = 1;
            maxsz[x] = 0;
            for(auto [y, w] : adj[x])
            {
                if(del[y] || y == fx) continue;
                dfs1(y, x);
                sz[x] += sz[y];
                maxsz[x] = max(maxsz[x], sz[y]);
            }
            maxsz[x] = max(maxsz[x], s - sz[x]);
            if(maxsz[x] < mxs)
            {
                mxs = maxsz[x], root = x;
            }
        };
        dfs1(x, -1);
        /////////////////////////////////
        mark[0] = T;
        c[0] = 0;
        for(auto [y, w] : adj[root])
        {
            if(del[y]) continue;
            vector<pair<int, int>> self;
            function<void(int, int, int, int)> dfs2 = [&](int x, int fx, int dis, int dep)
            {
                self.emplace_back(dis, dep);
                for(auto [y, w] : adj[x])
                {
                    if(del[y] || y == fx) continue;
                    dfs2(y, x, dis + w, dep + 1);
                }
            };
            dfs2(y, root, w, 1);
            for(auto [dis, dep] : self)
            {
                if(k - dis >= 0 && mark[k - dis] == T)
                {
                    ans = min(ans, c[k - dis] + dep);
                }
            }
            for(auto [dis, dep] : self)
            {
                if(dis > k) continue;
                if(mark[dis] == T)
                {
                    c[dis] = min(c[dis], dep);
                }else{
                    c[dis] = dep;
                    mark[dis] = T;
                }
```

```
            }
        }
        /////////////////////////////////
        del[root] = 1;
        for(auto [y, w] : adj[root])
        {
            if(del[y]) continue;
            solve(y, sz[y]);
        }
    };
    solve(1, n);
    cout << (ans > n ? -1 : ans) << "\n";
    return 0;
}
```

四、数论

exgcd

```
int exgcd(int a, int b, int &x, int &y)
{
    if(b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }
    int d = exgcd(b, a % b, y, x);
    y -= (a / b) * x;
    return d;
}
```

整除分块

```
for(ll l = 1 ; l <= n ; l++ )
    {
        ll d = n / l, r = n / d;
        cout << l << " : " << r << " = " << d << endl;
        l = r;
    }
```

sieve

```
int tot;
int p[N], pr[N], pe[N];
// p[x]为x最小的质因子, pr[x]为第x个质数, pe[x]为x的最小质因子个数幂

```

```cpp
void sieve(int n) {
    for (int i = 2; i <= n; i++) {
        if (pe[i] == 0) p[i] = i, pe[i] = i, pr[++tot] = i;
        for (int j = 1; j <= tot && i * pr[j] <= n; j++) {
            p[i * pr[j]] = pr[j];
            if (p[i] == pr[j]) {
                pe[i * pr[j]] = pe[i] * pr[j];
                break;
            } else {
                pe[i * pr[j]] = pr[j];
            }
        }
    }
}

void compute(int f[], int n, std::function<int(int)> calcpe) {
    f[1] = 1;
    for (int i = 2; i <= n; i++) {
        if (i == pe[i]) f[i] = calcpe(i);
        else f[i] = f[pe[i]] * f[i / pe[i]];
    }
}
compute(d, n, [&](int x) {
    return d[x / p[x]] + 1;
}); // d

compute(sigma, n, [&](int x) {
    return sigma[x / p[x]] + x;
}); // sigma

compute(phi, [&](int x) {
    phi[x] = x - x / p[x]; // f[x] = x / p[x] * (p[x] - 1);
}); // phi

compute(mu, n, [&](int x) {
    mu[x] = (x == p[x] ? -1 : 0);
}); // mu
```

积性函数

$id(x) = x$

$I(x) = 1(x) = 1$

$e(x) = \begin{cases} 1, x = 1 \\ 0, x \neq 1 \end{cases}$

$d(n) = \text{因子个数}$

$\sigma(n) = \text{因子和}$

$\mu(n) = \begin{cases} 1 & , x = 1 \\ (-1)^k & , n\text{不含平方因子，}k\text{为质因子个数} \\ 0 & , n\text{含有平方因子} \end{cases}$

Dirchlet卷积

$h(n)=f*g=\sum\limits_{d|n}f(d)g(\frac{n}{d})=\sum\limits_{d_1d_2=n}f(d_1)g(d_2)$

$d=1*1$

$d(n)=\sum\limits_{d|n}I(d)=\sum\limits_{d|n}I(d)I(\frac{n}{d})$

$\sigma=1*id$

$\sigma(n)=\sum\limits_{d|n}I(d)id(\frac{n}{d})$

$f=f*e$

$f*g=g*f$

$f*(g*h)=(f*g)*h$

f,g是积性函数 $\rightarrow f*g$是积性函数

莫比乌斯反演

$f(n)=\sum\limits_{d|n}g(d)\Leftrightarrow g(n)=\sum\limits_{d|n}\mu(d)f(\frac{n}{d})=\sum\limits_{d|n}\mu(\frac{n}{d})f(d)$

$f=g*1\Leftrightarrow g=f*\mu$

$1*\mu=e$

$id*\mu=\phi$

$\phi*1=\mu$

$id=\mu*\sigma$

$n=\sum\limits_{d|n}\phi(d)$

ax-by=1的解

```
ll exgcd(ll a, ll b, ll &x, ll &y)
{
    if(b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }
    int d = exgcd(b, a % b, y, x);
    y -= (a / b) * x;
    return d;
}

void solve()
{
    ll a, b;
```

```cpp
    cin >> a >> b;
    ll x, y;
    ll d = exgcd(a, b, x, y);
    y = -y;
    while(x < 0 || y < 0)
    {
        x += b/d;
        y += a/d;
    }
    while(x >= b/d && y >= a/d)
    {
        x -= b/d;
        y -= a/d;
    }
    cout << x << " " << y << "\n";
}
```

pollard_rho

```cpp
using i64 = long long;
using i128 = __int128;
i64 power(i64 a, i64 b, i64 m) {
    i64 res = 1;
    for (; b; b >>= 1, a = i128(a) * a % m) {
        if (b & 1) {
            res = i128(res) * a % m;
        }
    }
    return res;
}

bool isprime(i64 p) {
    if (p < 2) {
        return 0;
    }
    i64 d = p - 1, r = 0;
    while (!(d & 1)) {
        r++;
        d >>= 1;
    }
    int prime[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
    for (auto a : prime) {
        if (p == a) {
            return true;
        }
        i64 x = power(a, d, p);
        if (x == 1 || x == p - 1) {
            continue;
        }
        for (int i = 0; i < r - 1; i++) {
            x = i128(x) * x % p;
            if (x == p - 1) {
                break;
            }
        }
```

```
        if (x != p - 1) {
            return false;
        }
    }
    return true;
}

mt19937 rng((unsigned int)
chrono::steady_clock::now().time_since_epoch().count());

i64 pollard_rho(i64 x) {
    i64 s = 0, t = 0;
    i64 c = i64(rng()) % (x - 1) + 1;
    i64 val = 1;
    for (int goal = 1; ; goal <<= 1, s = t, val = 1) {
        for (int step = 1; step <= goal; step++) {
            t = (i128(t) * t + c) % x;
            val = i128(val) * abs(t - s) % x;
            if (step % 127 == 0) {
                i64 g = gcd(val, x);
                if (g > 1) {
                    return g;
                }
            }
        }
        i64 g = gcd(val, x);
        if (g > 1) {
            return g;
        }
    }
}

unordered_map<i64, int> getprimes(i64 x) {
    unordered_map<i64, int> p;
    function<void(i64)> get = [&](i64 x) {
        if (x < 2) {
            return;
        }
        if (isprime(x)) {
            p[x]++;
            return;
        }
        i64 mx = pollard_rho(x);
        get(x / mx);
        get(mx);
    };
    get(x);
    return p;
}

```

fft

```
void fft(vector<complex<double>>&a){
    int n=a.size(),L=31-__builtin_clz(n);
    vector<complex<long double>>R(2,1);
    vector<complex<double>>rt(2,1);
    for(int k=2;k<n;k*=2){
        R.resize(n);
        rt.resize(n);
        auto x=polar(1.0L, acos(-1.0L)/k);
        for(int i=k;i<2*k;++i) rt[i]=R[i]=i&1?R[i/2]*x:R[i/2];
    }
    vector<int>rev(n);
    for(int i=0;i<n;++i) rev[i]=(rev[i/2]|(i&1)<<L)/2;
    for(int i=0;i<n;++i) if(i<rev[i]) swap(a[i],a[rev[i]]);
    for (int k=1;k<n;k*=2)
        for (int i=0;i<n;i+=2*k)
            for(int j=0;j<k;++j){
                complex<double>z=rt[j+k]*a[i+j+k];
                a[i+j+k]=a[i+j]-z;
                a[i+j]+=z;
            }
}

vector<double>mul(const vector<double>&a,const vector<double>&b){
    if(a.empty() || b.empty()) return {};
    vector<double>res(a.size()+b.size()-1);
    int L=32-__builtin_clz(res.size()),n=1<<L;
    vector<complex<double>>in(n),out(n);
    copy(a.begin(),a.end(),in.begin());
    for(int i=0;i<b.size();++i) in[i].imag(b[i]);
    fft(in);
    for(auto &x:in) x*=x;
    for(int i=0;i<n;++i) out[i]=in[-i&(n-1)]-conj(in[i]);
    fft(out);
    for(int i=0;i<res.size();++i) res[i]=imag(out[i])/(4 * n);
    return res;
}
```

ntt

```
int mod=998244353;

int qpow(int a,int b){
    int ans=1;
    for(;b;b>>=1){
        if(b&1) ans=ans*a%mod;
        a=a*a%mod;
    }
    return ans;
}

vector<int>roots{0,1};
vector<int>rev;

void dft(vector<int>&a){
    int n=a.size();
```

```cpp
    if(rev.size()!=n){
        rev.resize(n);
        int k=__builtin_ctzll(n)-1;
        for(int i=0;i<n;++i) rev[i]=rev[i>>1]>>1|(i&1)<<k;
    }
    for(int i=0;i<n;++i) if(i<rev[i]) swap(a[i],a[rev[i]]);
    if(roots.size()<n){
        int k=__builtin_ctzll(roots.size());
        roots.resize(n);
        while((1<<k)<n){
            int e=qpow(3,(mod-1)>>(k+1));
            for(int i=(1<<(k-1));i<(1<<k);++i){
                roots[2*i]=roots[i];
                roots[2*i+1]=roots[i]*e%mod;
            }
            k++;
        }
    }
    for(int k=1;k<n;k<<=1){
        for(int i=0;i<n;i+=2*k){
            for(int j=0;j<k;++j){
                int u=a[i+j],v=a[i+j+k]*roots[k+j]%mod;
                a[i+j]=(u+v)%mod;
                a[i+j+k]=(u-v+mod)%mod;
            }
        }
    }
}
void idft(vector<int>&a){
    reverse(a.begin()+1,a.end());
    dft(a);
    int n=a.size(),inv=(1-mod)/n+mod;
    for(int i=0;i<n;++i) a[i]=a[i]*inv%mod;
}

struct Poly{
    vector<int>a;
    friend Poly operator*(Poly a,Poly b){
        int sz=1,tot=a.a.size()+b.a.size()-1;
        while(sz<tot) sz<<=1;
        a.a.resize(sz);b.a.resize(sz);
        dft(a.a);dft(b.a);
        for(int i=0;i<sz;++i) a.a[i]=a.a[i]*b.a[i]%mod;
        idft(a.a);
        a.a.resize(tot);
        return a;
    }
};
```

拉格朗日单点求值

```cpp
int Lagrange(vector<int>x,vector<int>y,int n,int k){
    for(int i=0;i<n;++i) if(x[i]==k) return y[i];
    vector<int>inv(n,1ll);
    for(int i=0;i<n;++i){
```

```
        for(int j=i+1;j<n;++j){
            inv[i]=inv[i]*(x[i]-x[j]+MOD)%MOD;
            inv[j]=inv[j]*(x[j]-x[i]+MOD)%MOD;
        }
    }
    int sum=1,ans=0;
    for(int i=0;i<n;++i) sum=sum*(k-x[i]+MOD)%MOD;
    for(int i=0;i<n;++i){
        int tmp=inv[i]*(k-x[i]+MOD)%MOD;
        ans=(ans+y[i]*sum%MOD*ksm(tmp,MOD-2)%MOD)%MOD;
    }
    return ans;
}
```

拉格朗日多点插值

```
vector<Poly>Q;

Poly MulT(Poly a,Poly b){
    int n=a.size(),m=b.size();
    reverse(b.a.begin(),b.a.end());
    b=a*b;
    for(int i=0;i<n;++i) a[i]=b[i+m-1];
    return a;
}

void MPinit(Poly &a,int u,int cl,int cr){
    if(cl==cr){
        Q[u].resize(2);
        Q[u][0]=1,Q[u][1]=mod-a[cl];
        return;
    }
    int mid=cl+cr>>1;
    MPinit(a,u<<1,cl,mid);MPinit(a,u<<1|1,mid+1,cr);
    Q[u]=Q[u<<1]*Q[u<<1|1];
}

void MPcal(int u,int cl,int cr,Poly f,Poly &g){
    f.resize(cr-cl+1);
    if(cl==cr){
        g[cl]=f[0];
        return;
    }
    int mid=cl+cr>>1;
    MPcal(u<<1,cl,mid,MulT(f,Q[u<<1|1]),g);
    MPcal(u<<1|1,mid+1,cr,MulT(f,Q[u<<1]),g);
}

Poly Multipoints(Poly f,Poly a,int n){                 //n为f和a的最大长度
    f.resize(n+1),a.resize(n);
    Poly v(n);
    Q.resize(n<<2);
    MPinit(a,1,0,n-1);
    MPcal(1,0,n-1,MulT(f,Q[1].inv(n+1)),v);
    return v;
```

```
}
```

五、数据结构

ST表

```
for(int i = 1 ; i <= n ; i++ )
{
    a[i] = read();
    f[0][i] = a[i];
}
for(int i = 1 ; i <= 22 ; i++ )
{
    for(int j = 1 ; j + (1 << i) - 1 <= n ; j++ )
    {
        f[i][j] = max(f[i-1][j], f[i-1][j + (1 << i - 1)]);
    }
}
for(int i = 1 ; i <= m ; i++ )
{
    int l = read(), r = read();
    int len = __lg(r - l + 1);
    printf("%d\n", max(f[len][l], f[len][r - (1 << len) + 1]));
}
```

树状数组

```
template<class T>
struct BIT {
    int size = 1;
    std::vector<T> c;
    BIT (int x) {
        size = x + 5;
        c.resize(x + 5);
    }
    void init(int x) {
        size = x + 5;
        c.resize(x + 5);
    }
    void clear() {
        for(int i = 0; i < size; i++) {
            c[i] = 0;
        }
    }
    void change(int x, T y) {
        for(; x < size ; x += x & (-x)) {
            c[x] += y;
        }
    }
    T query(int x) {
        T s = 0;
```

```
        for(;x ;x -= x & (-x)) {
            s += c[x];
        }
        return s;
    }
    T query(int l, int r) {
        if (l == 0) return query(r);
        return query(r) - query(l - 1);
    }
    int kth(int k) {
        int sum = 0, x = 0;
        for (int i = log2(size); ~i; --i) {
            x += 1 << i;
            if (x >= size || sum + c[x] >= k)
                x -= 1 << i;
            else
                sum += c[x];
        }
        return x + 1;
    }
};
```

并查集

```
struct DSU {
    std::vector<int> f, siz;
    DSU(int n) : f(n), siz(n, 1) { std::iota(f.begin(), f.end(), 0); }
    int leader(int x) {
        while (x != f[x]) x = f[x] = f[f[x]];
        return x;
    }
    bool same(int x, int y) { return leader(x) == leader(y); }
    bool merge(int x, int y) {
        x = leader(x);
        y = leader(y);
        if (x == y) return false;
        siz[x] += siz[y];
        f[y] = x;
        return true;
    }
    int size(int x) { return siz[leader(x)]; }
};

struct DSU {
    std::vector<int> parent, siz;
    std::vector<std::array<int, 5> > stk;
    DSU(int n) : parent(n + 1), siz(n + 1, 1) {
        std::iota(parent.begin(), parent.end(), 0);
    }

    int leader(int x) {
        while (x != parent[x]) {
            x = parent[x];
        }
        return x;
```

```cpp
    }

    bool merge(int x, int y, int t) {
        x = leader(x), y = leader(y);
        if (x == y) return false;
        if (siz[x] < siz[y]) {
            std::swap(x, y);
        }

        stk.push_back({t, x, siz[x], y, siz[y]});
        siz[x] += siz[y];
        parent[y] = x;
        return true;
    }

    void undo(int t) {
        while (stk.size() && stk.back()[0] > t) {
            auto &[_, x, sx, y, sy] = stk.back();
            siz[x] = sx;
            parent[x] = x;
            siz[y] = sy;
            parent[y] = y;
            stk.pop_back();
        }
    }

};
```

二维树状数组维护区间查询，修改

```cpp
ll c1[N][N], c2[N][N], c3[N][N], c4[N][N];

int n, m, k, q;

int lowbit(int x)
{
    return x & (-x);
}

void add(ll x, ll y, ll d)
{
    for(int i = x ; i <= n ; i += lowbit(i))
    {
        for(int j = y ; j <= m ; j += lowbit(j))
        {
            //cout << "test" << endl;
            c1[i][j] += d;
            c2[i][j] += d * x;
            c3[i][j] += d * y;
            c4[i][j] += d * x * y;
        }
    }
}

void modify(int x1, int y1, int x2, int y2, int d)
```

```
{
    add(x1, y1, d);
    add(x1, y2 + 1, -d);
    add(x2 + 1, y1, -d);
    add(x2 + 1, y2 + 1, d);
}

ll sum(ll x, ll y)
{
    ll ans = 0;
    for(int i = x ; i ; i -= lowbit(i))
    {
        for(int j = y ; j ; j -= lowbit(j))
        {
            ans += (x + 1) * (y + 1) * c1[i][j];
            ans -= (y + 1) * c2[i][j];
            ans -= (x + 1) * c3[i][j];
            ans += c4[i][j];
        }
    }
    return ans;
}
ll query(int x1, int y1, int x2, int y2)
{
    return (sum(x2, y2) - sum(x1 - 1, y2) - sum(x2, y1 - 1) + sum(x1 - 1, y1 - 1));
}
int h[100005];
int main()
{
    fastio
    //freopen("1.in","r",stdin);
    cin >> n >> m >> k >> q;
    for(int i = 1 ; i <= k ; i++ )
    {
        cin >> h[i];
    }
    for(int i = 1 ; i <= q ; i++ )
    {
        int op;
        cin >> op;
        if(op == 1)
        {
            int a, b, c, d, id;
            cin >> a >> b >> c >> d >> id;
            modify(a, b, c, d, h[id]);
        }else{
            int a, b, c, d;
            cin >> a >> b >> c >> d;
            cout << query(a, b, c, d) << "\n";
        }
    }
    return 0;
}

```

SegmentTree

```
struct Info {

};

Info operator+(const Info &a, const Info &b){

}

template<class Info>
struct SegmentTree{
    int n;
    vector<Info> info;

    SegmentTree() {}

    SegmentTree(int n, Info _init = Info()){
        init(vector<Info>(n, _init));
    }

    SegmentTree(const vector<Info> &_init){
        init(_init);
    }

    void init(const vector<Info> &_init){
        n = (int)_init.size();
        info.assign((n << 2) + 1, Info());
        function<void(int, int, int)> build = [&](int p, int l, int r){
            if (l == r){
                info[p] = _init[l - 1];
                return;
            }
            int m = (l + r) / 2;
            build(2 * p, l, m);
            build(2 * p + 1, m + 1, r);
            pull(p);
        };
        build(1, 1, n);
    }

    void pull(int p){
        info[p] = info[2 * p] + info[2 * p + 1];
    }

    void modify(int p, int l, int r, int x, Info v){
        if (l == r){
            info[p] = v;
            return;
        }
        int m = (l + r) / 2;
        if (x <= m){
            modify(2 * p, l, m, x, v);
        }
        else{
```

```
            modify(2 * p + 1, m + 1, r, x, v);
        }
        pull(p);
    }

    void modify(int p, Info v){
        modify(1, 1, n, p, v);
    }

    Info query(int p, int l, int r, int x, int y){
        if (l > y || r < x){
            return Info();
        }
        if (l >= x && r <= y){
            return info[p];
        }
        int m = (l + r) / 2;
        return query(2 * p, l, m, x, y) + query(2 * p + 1, m + 1, r, x, y);
    }

    Info query(int l, int r){
        return query(1, 1, n, l, r);
    }
};
```

LazySegmentTree

```
struct Info {
    ll sum = 0, len = 0;
};

struct Tag {
    ll add = 0;
};

Info operator+(const Info &a, const Info &b){
    return {a.sum + b.sum, a.len + b.len};
}

void apply(Info &x, Tag &a, Tag f){
    x.sum += x.len * f.add;
    a.add += f.add;
}

template<class Info, class Tag>
struct LazySegmentTree{
    int n;
    vector<Info> info;
    vector<Tag> tag;

    LazySegmentTree() {}

    LazySegmentTree(int n, Info _init = Info()){
```

```
        init(vector<Info>(n, _init));
    }

    LazySegmentTree(const vector<Info> &_init){
        init(_init);
    }

    void init(const vector<Info> &_init){
        n = (int)_init.size() - 1;
        info.assign((n << 2) + 1, Info());
        tag.assign((n << 2) + 1, Tag());
        function<void(int, int, int)> build = [&](int p, int l, int r){
            if (l == r){
                info[p] = _init[l];
                return;
            }
            int m = (l + r) / 2;
            build(2 * p, l, m);
            build(2 * p + 1, m + 1, r);
            pull(p);
        };
        build(1, 1, n);
    }

    void pull(int p){
        info[p] = info[2 * p] + info[2 * p + 1];
    }

    void apply(int p, const Tag &v){
        ::apply(info[p], tag[p], v);
    }

    void push(int p){
        apply(2 * p, tag[p]);
        apply(2 * p + 1, tag[p]);
        tag[p] = Tag();
    }

    void modify(int p, int l, int r, int x, const Info &v){
        if (l == r){
            info[p] = v;
            return;
        }
        int m = (l + r) / 2;
        push(p);
        if (x <= m){
            modify(2 * p, l, m, x, v);
        }
        else{
            modify(2 * p + 1, m + 1, r, x, v);
        }
        pull(p);
    }

    void modify(int p, const Info &v){
        modify(1, 1, n, p, v);
```

```
    }

    Info query(int p, int l, int r, int x, int y){
        if (l > y || r < x){
            return Info();
        }
        if (l >= x && r <= y){
            return info[p];
        }
        int m = (l + r) / 2;
        push(p);
        return query(2 * p, l, m, x, y) + query(2 * p + 1, m + 1, r, x, y);
    }

    Info query(int l, int r){
        return query(1, 1, n, l, r);
    }

    void modify(int p, int l, int r, int x, int y, const Tag &v){
        if (l > y || r < x){
            return;
        }
        if (l >= x && r <= y){
            apply(p, v);
            return;
        }
        int m = (l + r) / 2;
        push(p);
        modify(2 * p, l, m, x, y, v);
        modify(2 * p + 1, m + 1, r, x, y, v);
        pull(p);
    }

    void modify(int l, int r, const Tag &v){
        return modify(1, 1, n, l, r, v);
    }
};

```

DynamicSegmentTree

```
class SegTree {
private:
    struct Node {
        Node () : left_(nullptr), right_(nullptr), val_(0), lazy_(0) {}
        int val_;
        int lazy_;
        Node* left_;
        Node* right_;
    };

public:
    Node* root_;
```

```cpp
    SegTree() { root_ = new Node(); }
    ~SegTree() {}

    // 更新区间值
    void upDate(Node* curNode, int curLeft, int curRight, int upDateLeft, int upDateRight, int addVal) {
        if (upDateLeft <= curLeft && upDateRight >= curRight) {
            // 如果需要更新的区间[upDateLeft, upDateRight] 包含了 当前这个区间[curLeft, curRight]
            // 那么暂存一下更新的值
            // 等到什么时候用到孩子结点了，再把更新的值发放给孩子
            curNode->val_ += addVal * (curRight - curLeft + 1);
            curNode->lazy_ += addVal;
            return;
        }

        // 到这里说明要用到左右孩子了
        // 因此，要用pushDown函数把懒标签的值传递下去
        int mid = (curLeft + curRight) / 2;
        pushDown(curNode, mid - curLeft + 1, curRight - mid);

        // 说明在[curLeft, curRight]中，
        if (upDateLeft <= mid) {
            upDate(curNode->left_, curLeft, mid, upDateLeft, upDateRight, addVal);
        }
        if (upDateRight > mid) {
            upDate(curNode->right_, mid + 1, curRight, upDateLeft, upDateRight, addVal);
        }

        // 更新了子节点还需要更新现在的结点
        pushUp(curNode);
    }


    // 把结点curNode的懒标记分发给左右孩子 然后自己的懒标记清零
    void pushDown(Node* curNode, int leftChildNum, int rightChildNum) {
        if (curNode->left_ == nullptr) curNode->left_ = new Node;
        if (curNode->right_ == nullptr) curNode->right_ = new Node;

        if (curNode->lazy_ == 0) return;

        curNode->left_->val_ += curNode->lazy_ * leftChildNum;
        curNode->left_->lazy_ += curNode->lazy_;

        curNode->right_->val_ += curNode->lazy_ * rightChildNum;
        curNode->right_->lazy_ += curNode->lazy_;

        curNode->lazy_ = 0;

        // 注意不需要递归再继续下推懒标签
        // 每次只需要推一层即可
    }

    // 一般是子节点因为要被用到了，所以需要更新值 因此也要同时更新父节点的值
```

```
    void pushUp(Node* curNode) {
        curNode->val_ = curNode->left_->val_ + curNode->right_->val_;
    }

    // 查询
    int query(Node* curNode, int curLeft, int curRight, int queryLeft, int queryRight) {
        if (queryLeft <= curLeft && queryRight >= curRight) {
            return curNode->val_;
        }
        // 用到左右结点力 先下推!
        int mid = (curLeft + curRight) / 2;
        pushDown(curNode, mid - curLeft + 1, curRight - mid);

        int curSum = 0;
        if (queryLeft <= mid) curSum += query(curNode->left_, curLeft, mid, queryLeft, queryRight);
        if (queryRight > mid) curSum += query(curNode->right_, mid + 1, curRight, queryLeft, queryRight);

        return curSum;
    }
};
```

PersistentSegmentTree

```
struct Info {
    int sum = 0;
};

Info operator+(const Info &a, const Info &b) {
    return {a.sum + b.sum};
}

struct PersistentSegmentTree {
    vector<Info> tr;
    vector<Info> a;
    vector<int> ls, rs;
    int n, idx = 1;
    PersistentSegmentTree(int _n) {
        this->n = _n;
        this->a = a;
        ls.resize(_n << 5);
        rs.resize(_n << 5);
        tr.resize(_n << 5);
        a.assign(_n + 1, {0});
        build(1, 1, _n);
    }
    void build(int u, int L, int R) {
        // test(u, L, R);
        if (L == R) {
            tr[u] = a[L];
            return;
        }
        int mid = L + R >> 1;
```

```
        if (!ls[u]) {
            ls[u] = ++idx;
        }
        if (!rs[u]) {
            rs[u] = ++idx;
        }
        build(ls[u], L, mid);
        build(rs[u], mid + 1, R);
    }
    int modify(int u, int L, int R, int p, int x) {
        if (L == R && p == L) {
            tr[++idx] = {tr[u].sum + 1};
            return idx;
        }
        int mid = L + R >> 1;
        if (p <= mid) {
            int id = modify(ls[u], L, mid, p, x);
            ls[++idx] = id;
            rs[idx] = rs[u];
            tr[idx] = tr[ls[idx]] + tr[rs[idx]];
            return idx;
        } else {
            int id = modify(rs[u], mid + 1, R, p, x);
            rs[++idx] = id;
            ls[idx] = ls[u];
            tr[idx] = tr[ls[idx]] + tr[rs[idx]];
            return idx;
        }
    }
    int query(int u, int v, int L, int R, int k) {
        if (L == R) return L;
        int x = tr[ls[v]].sum - tr[ls[u]].sum;
        int mid = L + R >> 1;
        if (x >= k) {
            return query(ls[u], ls[v], L, mid, k);
        } else {
            return query(rs[u], rs[v], mid + 1, R, k - x);
        }
    }
};
```

pbds

```
#include<ext/pb_ds/tree_policy.hpp>
#include<ext/pb_ds/assoc_container.hpp>

using namespace __gnu_pbds;
__gnu_pbds::tree<pair<ll,ll>, null_type, less<pair<ll,ll>>, rb_tree_tag, tree_order_statistics_node_update> T;

if(op == 1)
{
    T.insert({x, i});
```

```
}else if (op == 2)
{
    T.erase(T.lower_bound({x, 0}));
}else if (op == 3)
{
    cout << T.order_of_key({x, 0}) + 1 << "\n";
}else if (op == 4)
{
    cout << T.find_by_order(x - 1)->first << "\n";
}else if (op == 5)
{
    cout << prev(T.lower_bound({x, 0}))->first << "\n";
}else if (op == 6)
{
    cout << T.lower_bound({x + 1, 0})->first << "\n";
}


```

bitset

```
#include<tr2/dynamic_bitset>
using std::tr2::dynamic_bitset;
dynamic_bitset< >bt(n+1);
_Find_fisrt就是找到从低位到高位第一个1的位置
_Find_next就是找到当前位置的下一个1的位置
all/any/none checks if all, any or none of the bits are set to true
count   returns the number of bits set to true
set 1
reset 0
filp
```

六、简单计算几何

点

```
using i64 = long long;

using T = double;
struct Point {
    T x;
    T y;
    Point(T x = 0, T y = 0) : x(x), y(y) {}

    Point &operator+=(const Point &p) {
        x += p.x, y += p.y;
        return *this;
    }
    Point &operator-=(const Point &p) {
        x -= p.x, y -= p.y;
        return *this;
```

```
    }
    Point &operator*=(const T &v) {
        x *= v, y *= v;
        return *this;
    }
    friend Point operator-(const Point &p) {
        return Point(-p.x, -p.y);
    }
    friend Point operator+(Point lhs, const Point &rhs) {
        return lhs += rhs;
    }
    friend Point operator-(Point lhs, const Point &rhs) {
        return lhs -= rhs;
    }
    friend Point operator*(Point lhs, const T &rhs) {
        return lhs *= rhs;
    }
};

T dot(const Point &a, const Point &b) {
    return a.x * b.x + a.y * b.y;
}

T cross(const Point &a, const Point &b) {
    return a.x * b.y - a.y * b.x;
}
```

七、杂项

矩阵快速幂

```
struct Matrix{
    int n , m ;
    vector<vector<ll>> s;

    Matrix(int n , int m):n(n) ,m(m) , s(n , vector<ll>(m ,0)){}

    friend Matrix operator * (Matrix a , Matrix b){
        assert(a.m == b.n);
        Matrix res(a.n , b.m);
        for(int k = 0 ; k < a.m ; k ++ )
            for(int i = 0 ; i < a.n ; i ++ )
                for(int j = 0 ; j < b.m ; j ++ )
                    res.s[i][j] = (res.s[i][j] + a.s[i][k] * b.s[k][j] %
mod) % mod;
        return res;
    }

    Matrix qmi(ll b){
        assert(n == m);
        Matrix res(n , n);
        for(int i = 0 ; i < n ; i ++ )
            res.s[i][i] = 1;
        while(b){
```

```
            if(b & 1)res = ((*this) * res );
            b >>= 1;
            *this = (*this) * (*this);
        }
        return (*this) = res;
    };

};
```

组合数

```
ll fact[N] = {1}, inv[N] = {1};
ll C(ll x, ll y)
{
    return(((fact[x] * inv[y])% MOD * inv[x-y]) % MOD);
}

ll P(ll x, ll y)
{
    return fact[x] * inv[x - y] % MOD;
}

ll ksm(ll x, ll y)
{
    ll ans = 1;
    x %= MOD;
    while(y)
    {
        if(y&1)
        {
            ans = ans * x % MOD;
        }
        x = x * x % MOD;
        y /= 2;
    }
    return ans;
}

void build()
{
    for(int i = 1 ; i < N ; i++ )
    {
        fact[i] = fact[i-1] * i % MOD;
    }
    for(int i = 1 ; i < N ; i++ )
    {
        inv[i] = inv[i-1] * ksm(i, MOD-2) % MOD;
    }
}
```

八、python

```
'''
```

```
def main():
    Do somthing
if __name__ == '__main__':
    t = int(input())
    for i in range(t):
        main()
'''
for T in range(0,int(input())): #T组数据
    N=int(input())
    n,m=map(int,input().split())
    s=input()
    s=[int(x) for x in input().split()] #一行输入的数组
    a[1:]=[int(x) for x in input().split()] #从下标1开始读入一行
    for i in range(0,len(s)):
        a,b=map(int,input().split())

while True: #未知多组数据
    try:
        #n,m=map(int,input().split())
        #print(n+m,end="\n")
    except EOFError: #捕获到异常
        break
//////////////////////
'''多行输入，指定行数'''

n, m = map(int, input().strip().split())#获取第一行，获取第二行可以再写一句同样的语句
#需要矩阵承接数据时
data = []
for i in range(n):
    tmp = list(map(int, input().split()))
    data.append(tmp)

'''多行输入，不指定行数'''
try:
    data = []
    while True:
        line = input().strip() #strip去除左右两边的空白符
        if line == ' ':
            break
        tmp = list(map(int, line.split())) #split按空白符拆开
        data.append(tmp)
expect:
    pass

```

一些基本数据结构

python中的栈和队列可以使用列表来模拟，或者`import deque`

匿名函数使用lambda关键字来定义`lambda 参数: 表达式`

```
#使用中括号[]定义一个列表
# l=[23,'wtf',3.14]
list.append(obj)#将obj添加到list末尾,O(1)
list.insert(index,obj)#将obj插入列表index位置,O(n)
```

```
list.pop([index=-1])#移除元素并返回该元素
list.sort(key=None,reverse=False)#默认升序排序,O(nlogn)
list.reverse()#反转列表元素
list.clear()
len(list)#列表元素个数,O(1)
max(list)#返回列表元素最大值,O(n)
del list[2]#删除list中第三个元素

#用小括号定义一个元组,可以当作不能修改的list
# t=(23,'wtf',3.14)

#用花括号{}定义一个字典
d={key1:value1,key2:value2}#通过key访问value
print(d[key1])#输出value1
if key in dict : #key不存在会报错,要先询问
    do somthing #或者使用
d.get(key)
for key in d: #遍历字典d
    print(key,':',d.get(key))
dMerge=dict(d1,**d2)#将d1和d2合并为dMerge

#调用set()方法创建集合
s=set([1,2,3])#定义
s.add(4)#添加
s.remove(4)#删除
```

math库

```
import math
math.e #常量e,2.718281828459045
math.pi #常量pi,3.141592653589793
math.factorial(x) #x的阶乘
math.gcd(x,y) #x,y的gcd
math.sqrt(x) #x的平方根
x=math.log(n,a) #以a为底n的对数x,a^x=n,默认底数为e
math.log(32,2) #5.0
math.degrees(math.pi/4) #将Π/4转为角度
math.radians(45) #将45度转为弧度
math.cos(math.pi/4) #参数都为弧度
```

快速幂

```
def qmod(a,b,mod):
    a=a%mod
    ans=1
    while b!=0:
        if b&1:
            ans=(ans*a)%mod
        b>>=1
        a=(a*a)%mod
    return ans
```

并查集

```
N,m=map(int,input().split())
fa=[int(i) for i in range(N+1)]
siz=[1]*(N+1)
def findfa(x):
    if fa[x]!=x:
        fa[x]=findfa(fa[x])
    return fa[x]
def Merge(x,y):
    xx,yy=findfa(x),findfa(y)
    if xx == yy:
        return False
    if siz[xx] > siz[yy]: #按秩合并
        fa[yy]=xx
        siz[xx]+=siz[yy]
    else:
        fa[xx]=yy
        siz[yy]+=siz[xx]
    return True
for i in range(m):
    z,x,y=map(int,input().split())
    if z==1:
        Merge(x,y)
    else:
        print('Y' if findfa(x)==findfa(y)else 'N')
```

线段树区间加区间和

```
class SegTreeNode(): #python3中所有类默认都是新式类
    def __init__(self): #类似构造函数,类方法必须包含参数self
        self.value=0
        self.lazytag=0

Data=[0 for i in range(0,100010)]

class SegTree():
    def __init__(self):
        self.SegTree=[SegTreeNode() for i in range(0,400010)]

    def Build_SegTree(self,Root,L,R):
        if L==R:
            self.SegTree[Root].value=Data[L]
            return
        mid=(L+R)>>1
        self.Build_SegTree(Root<<1,L,mid)
        self.Build_SegTree(Root<<1|1,mid+1,R)
 self.SegTree[Root].value=self.SegTree[Root<<1].value+self.SegTree[Root<<1|1
].value
        return

    def Push_Down(self,Root,L,R):
        if self.SegTree[Root].lazytag==0:
```

```
            return
        Add=self.SegTree[Root].lazytag
        self.SegTree[Root].lazytag=0
        mid=(L+R)>>1
        self.SegTree[Root<<1].value+=(mid-L+1)*Add
        self.SegTree[Root<<1|1].value+=(R-mid)*Add
        self.SegTree[Root<<1].lazytag+=Add
        self.SegTree[Root<<1|1].lazytag+=Add
        return

    def Update(self,Root,L,R,QL,QR,Add):
        if R<QL or QR<L:
            return
        if QL<=L and R<=QR:
            self.SegTree[Root].value+=(R-L+1)*Add
            self.SegTree[Root].lazytag+=Add
            return
        mid=(L+R)>>1
        self.Push_Down(Root,L,R)
        self.Update(Root<<1,L,mid,QL,QR,Add)
        self.Update(Root<<1|1,mid+1,R,QL,QR,Add)
        self.SegTree[Root].value=self.SegTree[Root<<1].value+self.SegTree[Root<<1|1].value
        return

    def Query(self,Root,L,R,QL,QR):
        if R<QL or QR<L:
            return 0
        if QL<=L and R<=QR:
            return self.SegTree[Root].value
        mid=(L+R)>>1
        self.Push_Down(Root,L,R)
        return self.Query(Root<<1,L,mid,QL,QR)+self.Query(Root<<1|1,mid+1,R,QL,QR)

Tree=SegTree()
N,M=map(int,input().split())
a=input().split() #初始值

for i in range(1,N+1):
    Data[i]=int(a[i-1])

Tree.Build_SegTree(1,1,N)

while M:
    opt,L,R=map(int,input().split())
    if opt==1:
        Tree.Update(1,1,N,L,R,int(a[3]))
    else:
        print(str(Tree.Query(1,1,N,L,R)))
    M-=1
```

字符串

```
ord('a')# 返回单个字符的 unicode:
chr(100)# 返回'd'

#strip和split
'   spacious   '.strip()#strip()移除 string 前后的字符串，默认来移除空格
'1,2,3'.split(',') #['1', '2', '3'],按照某个字符串来切分，返回一个 list,
'1,2,3'.split(',', maxsplit=1)#['1', '2,3'],传入一个参数maxsplit来限定分离数

#将字符串和列表相互转换
字符串转换成列表，注意交换字符需要先转换成列表
#1.list
str1 = '12345'
list1 = list(str1)
print(list1) #['1', '2', '3', '4', '5']
#2.str.split()通过指定分隔符对字符串进行切片
str3 = 'this is string example'
list3 = str3.split('i', 1)
print(list3) #['th', 's is string example']

列表转换成字符串，join里面的可以是list、set
#1.split.join(str),split是指定的分隔符，str是要转换的字符串
list1 = ['1', '2', '3', '4', '5']
str1 = "".join(list1)#12345

list3 = ['www', 'baidu', 'com']
str3 = ".".join(list3)#www.baidu.com

#是元音
def isVowel(ch:str) -> bool:
            return ch in "aeiouAEIOU"
isVowel(s[i])

```

二维列表

```
ls = [] #二维列表新建可以直接建一个一维列表，后面直接append列表数据就可以了
ls_T = list(map(list, zip(*ls)))# 转置，用于取列元素
if 元素 in ls_T[0]: #判断是不是在0列里面
    j = ls_T[0].index(元素) #第0列中该元素的位置，即多少行
```

list

```
#初始化
l = [0] * len(array)
l=[]

#从后往前访问
l[-1]表示最后一个数
for i in range(0, -10, -1)     #0, -1, -2, -3, -4, -5, -6, -7, -8, -9
for j in reversed(range(len(nums)-1)) #加一个reverse可以直接颠倒

#enumerate 枚举
l = ["a", "b", "c"]
for i, v in enumerate(l):
```

```
    print(i, v)
#0 a
#1 b
#2 c

#map
#可以将参数一一映射来计算， 比如
date = "2019-8-15"
Y, M, D = map(int, date.split('-'))    #Y = 2019, M = 8, D = 15
#map返回的是迭代对象而不是一个列表，要转成列表要加list


#sort
1.调用sort()排序，不会产生新的列表。lst.sort()升序排序
降序排序lst.sort(reverse=True)  升序排序lst.sort()
2.使用内置函数sorted()排序，会产生新的列表对象
lst1=sorted(lst)升序排序   lst2=sorted(lst,reverse=True)降序排序
l1 = [(1,2), (0,1), (3,10) ]
l2 = sorted(l1, key=lambda x: x[0])#按照 tuple 的第一个元素进行排序key允许传入一个自定义参数
# l2 = [(0, 1), (1, 2), (3, 10)]
#排序默认从小到大。可以用reverse=True倒序

#列表生成式
lst = [i*j for i in range(1,10)]
#ZIP
x = [1, 2, 3]
y = [4, 5, 6]
zipped = zip(x, y)
list(zipped)#[(1, 4), (2, 5), (3, 6)]
```keys(), values(), items()
这三个方法可以分别获得key, value, {key: value}的数组。

#max可以代替if来更新更大的数
maxnums=max(maxnums,tmp)

#多维数组
res = [[], []]
res[0].append()

#extend一次性添加多个元素
lst1.extend(lst2)
#insert在i位置添加x
lst.insert(i, x)

```
```

常用函数

```
round(x)：四舍五入
abs(x)/max()/min()：绝对值/最大值/最小值
range(start=0, stop, step=1])：返回一个可迭代对象，常用于for循环
pow(x, y, [z])：求幂函数x^y，运算完毕可以顺带对z取模
sorted(iterable, key, reverse)：采用Timsort的稳定排序算法，默认升序
int(x, base=10))/float()/str()：转整数(可自定义进制)/转浮点数/转字符串
bin()/oct()/hex()：10进制转二进制(返回0b开头的字符串)/10进制转八进制(返回0开头的字符串)/10进制转十六进制(返回0x开头的字符串)
ord()/chr()：字符转ASCII或ASCII转字符
math.gcd(x,y)：返回x和y的最大公约数

if ……elif……else注意不要用else if
```

验证数据

最大流(dinic)

```
signed main()
{
    mt19937 rand(0);
    for (int i = 1; i <= 20; i++) {
        int n = rand() % 100, m = rand() % (n * n);
        int s = n + 1, t = n + 2;
        g.init(s, t, t);
        for (int i = 1; i <= m; i++) {
            int u, v, w;
            u = rand() % (n + 2) + 1;
            v = rand() % (n + 2) + 1;
            w = rand() % n;
            g.addedge(u, v, w);//u -> v单向边
        }
        cout << g.dinic() << " ";
    }
}
```

```
722 12 377 500 1240 412 460 550 95 2039 523 0 40 1655 877 221 3562 100 0 2528
```

最小费用最大流

```
mt19937 rand(0);
for (int i = 1; i <= 20; i++) {
    int n = rand() % 100 + 2;
    int m = rand() % (n * n) + 1;
    int s = n - 1, t = n;
    g.init(s, t, t);
    for (int i = 1; i <= m; i++) {
        int u, v, w, c;
        u = rand() % n + 1;
        v = rand() % n + 1;
```

```
        w = rand() % 1000;
        c = rand() % 1000;
        g.addedge(u, v, w, c);
    }
    auto [ans1, ans2] = g.sol();
    cout << ans1 << " " << ans2 << "\n";

}
```

```
15932 14704703
2209 2617444
21746 22827734
13113 14083600
21734 21946209
28796 27196768
3776 4579568
28384 29502294
23196 24861190
0 0
1288 1898029
1025 1127660
2807 1738067
36782 38352187
624 1922442
9168 10702007
10849 10835609
3154 4430069
8088 8840656
32961 31591050
```

Splay

```
mt19937 rand(0);
int n = 1000;
insert(1e18);
insert(-1e18);
int ans1 = 0, ans2 = 0;
while(n--) {
    int x = rand() % 6 + 1;
    int y = rand() % 10000;
    if(x == 1) {
        insert(y);
    }
    if(x == 2) {
        del(y);
    }
    if(x == 3) {
        int tmp = get_rank(y);
        ans1 += tmp;
        ans2 ^= tmp;
    }
    if(x == 4) {
        int tmp = get_val(y);
        ans1 += tmp;
```

```
            ans2 ^= tmp;
        }
        if(x == 5) {
            int tmp = tr[get_pre(y)].v;
            ans1 += tmp;
            ans2 ^= tmp;
        }
        if(x == 6) {
            int tmp = tr[get_suc(y)].v;
            ans1 += tmp;
            ans2 ^= tmp;
        }
    }
    cout << ans1 << " " << ans2 << "\n";
```

```
1000000000002329961 1000000000000001667
```

fft

```
signed main {
    mt19937 rand(0);
    for (int i = 1; i <= 20; i++) {
        int n = rand() % 100 + 1, m = rand() % 100 + 1;
        vector<double> a(n), b(m);
        for (int i = 0 ; i < n; i++) {
            a[i] = rand() % 100 + 1;
        }
        for (int i = 0; i < m; i++) {
            b[i] = rand() % 100 + 1;
        }
        auto ans = mul(a, b);
        double sum = 0;
        for (auto x : ans) {
            sum += x;
        }
        cout << fixed << setprecision(0) << sum << " ";
    }
}
```

```
5507964 1764445 1323685 8355072 22732435 4250782 3275356 1602420 4754812
6657250 4982142 2173858 109809 2031180 12458752 7225646 11931738 15165549
595010 47796
```

ntt

```
signed main() {
    mt19937 rand(0);
    for (int i = 1; i <= 20; i++) {
        int n = rand() % 100 + 1, m = rand() % 100 + 1;
        Poly a, b;
```

```cpp
        a.a.resize(n);
        b.a.resize(m);
        for (int i = 0 ; i < n; i++) {
            a.a[i] = rand() % 100 + 1;
        }
        for (int i = 0; i < m; i++) {
            b.a[i] = rand() % 100 + 1;
        }
        Poly ans = a * b;
        int sum = 0;
        for (auto x : ans.a) {
            sum += x;
        }
        cout << sum << " ";
    }
}
```

```
5507964 1764445 1323685 8355072 22732435 4250782 3275356 1602420 4754812 6657250 4982142 2173858 109809 2031180 12458752 7225646 11931738 15165549 595010 47796
```

Prime

```
2 2 3 3 5 5 7 7
11 11 11 13 17 19 23 29 41 79 83 83 89 97
101 103 107 109 113 331 547 587 797 977 983 991 997
1009 1013 1019 1021 1031 2693 8039 8467 9547 9941 9949 9967 9973
10007 10009 10037 10039 10061 46381 57077 62851 98213 99961 99971 99989 99991
100003 100019 100043 100049 100057 107183 234383 573509 984007 999959 999961 999979 999983
1000003 1000033 1000037 1000039 1000081 1016927 1055189 6900961 7922111 9999943 9999971 9999973 9999991
10000019 10000079 10000103 10000121 10000139 10917271 68353301 75707057 88814903 99999941 99999959 99999971 99999989
100000007 100000037 100000039 100000049 100000073 166082023 574322789 654228647 676053907 999999883 999999893 999999929 999999937
1000000007 1000000009 1000000021 1000000033 1000000087 3397148761 5440806487 7154354923 9380086069 9999999881 9999999929 9999999943 9999999967
10000000019 10000000033 10000000061 10000000069 10000000097 12482387257 37315188863 50976305209 54383534603 99999999907 99999999943 99999999947 99999999977
100000000003 100000000019 100000000057 100000000063 100000000069 286708136027 452216566141 733218692447 772825281731 999999999937 999999999959 999999999961 999999999989
1000000000039 1000000000061 1000000000063 1000000000091 1000000000121 3032586593209 3836572107527 5416485186193 7173322551641 9999999999763 9999999999799 9999999999863 9999999999971
10000000000037 10000000000051 10000000000099 10000000000129 10000000000183 36885590072783 37541934267829 48213030604087 50498419100719 99999999999931 99999999999959 99999999999971 99999999999973
```

```
100000000000031 100000000000067 100000000000097 100000000000099 100000000000133 403474996735879 421689567204827 779233068792001 896525624400839 999999999999877 999999999999883 999999999999947 999999999999989
1000000000000037 1000000000000091 1000000000000159 1000000000000187 1000000000000223 4520969764685597 4892903438650489 5545765365226421 6511840361739199 9999999999999851 9999999999999887 9999999999999917 9999999999999937
10000000000000061 10000000000000069 10000000000000079 10000000000000099 10000000000000453 15275914421071933 20593827648809693 58624671566507821 71885865416282849 99999999999999943 99999999999999961 99999999999999977 99999999999999997
100000000000000003 100000000000000013 100000000000000019 100000000000000021 100000000000000049 407170308628310287 798161809347356543 912597791950842211 971601792743437627 999999999999999863 999999999999999877 999999999999999967 999999999999999989
1000000000000000003 1000000000000000009 1000000000000000031 1000000000000000079 1000000000000000177 1123531975834070419 2245710371805018757 3108977463994466341 4717942754514214051
```

题号	读	会	过	知识点	注意点
A					
B					
C					
D					
E					
F					
G					
H					
I					
J					
K					
L					
M					

Contents

积分表

Integrals of Rational Functions

$$\int \frac{1}{1+x^2}dx = \tan^{-1} x \tag{1}$$

$$\int \frac{1}{a^2+x^2}dx = \frac{1}{a}\tan^{-1}\frac{x}{a} \tag{2}$$

$$\int \frac{x}{a^2+x^2}dx = \frac{1}{2}\ln|a^2+x^2| \tag{3}$$

$$\int \frac{x^2}{a^2+x^2}dx = x - a\tan^{-1}\frac{x}{a} \tag{4}$$

$$\int \frac{x^3}{a^2+x^2}dx = \frac{1}{2}x^2 - \frac{1}{2}a^2\ln|a^2+x^2| \tag{5}$$

$$\int \frac{1}{ax^2+bx+c}dx = \frac{2}{\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}} \tag{6}$$

$$\int \frac{1}{(x+a)(x+b)}dx = \frac{1}{b-a}\ln\frac{a+x}{b+x},\ a\neq b \tag{7}$$

$$\int \frac{x}{(x+a)^2}dx = \frac{a}{a+x} + \ln|a+x| \tag{8}$$

$$\int \frac{x}{ax^2+bx+c}dx = \frac{1}{2a}\ln|ax^2+bx+c| - \frac{b}{a\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}} \tag{9}$$

Integrals with Roots

$$\int \frac{x}{\sqrt{x\pm a}}dx = \frac{2}{3}(x\mp 2a)\sqrt{x\pm a} \tag{10}$$

$$\int \sqrt{\frac{x}{a-x}}dx = -\sqrt{x(a-x)} - a\tan^{-1}\frac{\sqrt{x(a-x)}}{x-a} \tag{11}$$

$$\int \sqrt{\frac{x}{a+x}}dx = \sqrt{x(a+x)} - a\ln\left[\sqrt{x}+\sqrt{x+a}\right] \tag{12}$$

$$\int x\sqrt{ax+b}dx = \frac{2}{15a^2}(-2b^2+abx+3a^2x^2)\sqrt{ax+b} \tag{13}$$

$$\int \sqrt{x(ax+b)}dx = \frac{1}{4a^{3/2}}\left[(2ax+b)\sqrt{ax(ax+b)} - b^2\ln\left|a\sqrt{x}+\sqrt{a(ax+b)}\right|\right] \tag{14}$$

$$\int \sqrt{x^3(ax+b)}dx = \left[\frac{b}{12a} - \frac{b^2}{8a^2x} + \frac{x}{3}\right]\sqrt{x^3(ax+b)} + \frac{b^3}{8a^{5/2}}\ln\left|a\sqrt{x}+\sqrt{a(ax+b)}\right| \tag{15}$$

$$\int \sqrt{x^2\pm a^2}dx = \frac{1}{2}x\sqrt{x^2\pm a^2} \pm \frac{1}{2}a^2\ln\left|x+\sqrt{x^2\pm a^2}\right| \tag{16}$$

$$\int \sqrt{a^2-x^2}dx = \frac{1}{2}x\sqrt{a^2-x^2} + \frac{1}{2}a^2\tan^{-1}\frac{x}{\sqrt{a^2-x^2}} \tag{17}$$

$$\int x\sqrt{x^2\pm a^2}dx = \frac{1}{3}\left(x^2\pm a^2\right)^{3/2} \tag{18}$$

$$\int \frac{1}{\sqrt{x^2\pm a^2}}dx = \ln\left|x+\sqrt{x^2\pm a^2}\right| \tag{19}$$

$$\int \frac{1}{\sqrt{a^2-x^2}}dx = \sin^{-1}\frac{x}{a} \tag{20}$$

$$\int \frac{x}{\sqrt{x^2\pm a^2}}dx = \sqrt{x^2\pm a^2} \tag{21}$$

$$\int \frac{x}{\sqrt{a^2-x^2}}dx = -\sqrt{a^2-x^2} \tag{22}$$

$$\int \frac{x^2}{\sqrt{x^2\pm a^2}}dx = \frac{1}{2}x\sqrt{x^2\pm a^2} \mp \frac{1}{2}a^2\ln\left|x+\sqrt{x^2\pm a^2}\right| \tag{23}$$

$$\int \sqrt{ax^2+bx+c}dx = \frac{b+2ax}{4a}\sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx^+c)}\right| \tag{24}$$

$$\int x\sqrt{ax^2+bx+c} = \frac{1}{48a^{5/2}}\left(2\sqrt{a}\sqrt{ax^2+bx+c} \times \left(-3b^2+2abx+8a(c+ax^2)\right) +3(b^3-4abc)\ln\left|b+2ax+2\sqrt{a}\sqrt{ax^2+bx+c}\right|\right) \tag{25}$$

$$\int \frac{1}{\sqrt{ax^2+bx+c}}dx = \frac{1}{\sqrt{a}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right| \tag{26}$$

$$\int \frac{x}{\sqrt{ax^2+bx+c}}dx = \frac{1}{a}\sqrt{ax^2+bx+c} - \frac{b}{2a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right| \tag{27}$$

$$\int \frac{dx}{(a^2+x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2+x^2}} \tag{28}$$

Integrals with Logarithms

$$\int \frac{\ln ax}{x} dx = \frac{1}{2} (\ln ax)^2 \tag{29}$$

$$\int \ln(ax+b)dx = \left(x + \frac{b}{a}\right) \ln(ax+b) - x, a \neq 0 \tag{30}$$

$$\int \ln(x^2 + a^2)\,\mathrm{dx} = x\ln(x^2+a^2) + 2a\tan^{-1}\frac{x}{a} - 2x \tag{31}$$

$$\int \ln(x^2 - a^2)\,\mathrm{dx} = x\ln(x^2-a^2) + a\ln\frac{x+a}{x-a} - 2x \tag{32}$$

$$\int \ln\left(ax^2+bx+c\right)dx = \frac{1}{a}\sqrt{4ac-b^2}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}} - 2x + \left(\frac{b}{2a}+x\right)\ln\left(ax^2+bx+c\right) \tag{33}$$

$$\int x\ln(ax+b)dx = \frac{bx}{2a} - \frac{1}{4}x^2 + \frac{1}{2}\left(x^2 - \frac{b^2}{a^2}\right)\ln(ax+b) \tag{34}$$

$$\int x\ln\left(a^2-b^2x^2\right)dx = -\frac{1}{2}x^2 + \frac{1}{2}\left(x^2-\frac{a^2}{b^2}\right)\ln\left(a^2-b^2x^2\right) \tag{35}$$

Integrals with Exponentials

$$\int x^n e^{ax}\,\mathrm{dx} = \frac{x^n e^{ax}}{a} - \frac{n}{a}\int x^{n-1}e^{ax}\,\mathrm{dx} \tag{36}$$

$$\int xe^{-ax^2}\,\mathrm{dx} = -\frac{1}{2a}e^{-ax^2} \tag{37}$$

Integrals with Trigonometric Functions

$$\int \sin^3 ax dx = -\frac{3\cos ax}{4a} + \frac{\cos 3ax}{12a} \tag{38}$$

$$\int \cos^2 ax dx = \frac{x}{2} + \frac{\sin 2ax}{4a} \tag{39}$$

$$\int \cos^3 ax dx = \frac{3\sin ax}{4a} + \frac{\sin 3ax}{12a} \tag{40}$$

$$\int \cos ax \sin bx dx = \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)}, a \neq b \tag{41}$$

$$\int \sin^2 ax \cos bx dx = -\frac{\sin[(2a-b)x]}{4(2a-b)} + \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)} \tag{42}$$

$$\int \sin^2 x \cos x dx = \frac{1}{3}\sin^3 x \tag{43}$$

$$\int \cos^2 ax \sin bx dx = \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} - \frac{\cos[(2a+b)x]}{4(2a+b)} \tag{44}$$

$$\int \cos^2 ax \sin ax dx = -\frac{1}{3a}\cos^3 ax \tag{45}$$

$$\int \sin^2 ax \cos^2 bx dx = \frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} + \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)} \tag{46}$$

$$\int \sin^2 ax \cos^2 ax dx = \frac{x}{8} - \frac{\sin 4ax}{32a} \tag{47}$$

$$\int \tan ax dx = -\frac{1}{a}\ln\cos ax \tag{48}$$

$$\int \tan^2 ax dx = -x + \frac{1}{a}\tan ax \tag{49}$$

$$\int \tan^3 ax dx = \frac{1}{a}\ln\cos ax + \frac{1}{2a}\sec^2 ax \tag{50}$$

$$\int \sec x dx = \ln|\sec x + \tan x| = 2\tanh^{-1}\left(\tan\frac{x}{2}\right) \tag{51}$$

$$\int \sec^2 ax dx = \frac{1}{a}\tan ax \tag{52}$$

$$\int \sec^3 x\,\mathrm{dx} = \frac{1}{2}\sec x\tan x + \frac{1}{2}\ln|\sec x + \tan x| \tag{53}$$

$$\int \sec x \tan x dx = \sec x \tag{54}$$

$$\int \sec^2 x \tan x dx = \frac{1}{2}\sec^2 x \tag{55}$$

$$\int \sec^n x \tan x dx = \frac{1}{n}\sec^n x, n \neq 0 \tag{56}$$

$$\int \csc x dx = \ln\left|\tan\frac{x}{2}\right| = \ln|\csc x - \cot x| + C \tag{57}$$

$$\int \csc^2 ax dx = -\frac{1}{a}\cot ax \tag{58}$$

$$\int \csc^3 x dx = -\frac{1}{2}\cot x\csc x + \frac{1}{2}\ln|\csc x - \cot x| \tag{59}$$

$$\int \csc^n x \cot x dx = -\frac{1}{n}\csc^n x, n \neq 0 \tag{60}$$

$$\int \sec x \csc x dx = \ln|\tan x| \tag{61}$$

Products of Trigonometric Functions and Monomials

$$\int x\cos x dx = \cos x + x\sin x \tag{62}$$

$$\int x\cos ax dx = \frac{1}{a^2}\cos ax + \frac{x}{a}\sin ax \tag{63}$$

$$\int x^2\cos x dx = 2x\cos x + \left(x^2-2\right)\sin x \tag{64}$$

$$\int x^2\cos ax dx = \frac{2x\cos ax}{a^2} + \frac{a^2x^2-2}{a^3}\sin ax \tag{65}$$

$$\int x\sin x dx = -x\cos x + \sin x \tag{66}$$

$$\int x\sin ax dx = -\frac{x\cos ax}{a} + \frac{\sin ax}{a^2} \tag{67}$$

$$\int x^2\sin x dx = \left(2-x^2\right)\cos x + 2x\sin x \tag{68}$$

$$\int x^2\sin ax dx = \frac{2-a^2x^2}{a^3}\cos ax + \frac{2x\sin ax}{a^2} \tag{69}$$

Products of Trigonometric Functions and Exponentials

$$\int e^x\sin x dx = \frac{1}{2}e^x(\sin x - \cos x) \tag{70}$$

$$\int e^{bx}\sin ax dx = \frac{1}{a^2+b^2}e^{bx}(b\sin ax - a\cos ax) \tag{71}$$

$$\int e^x\cos x dx = \frac{1}{2}e^x(\sin x + \cos x) \tag{72}$$

$$\int e^{bx}\cos ax dx = \frac{1}{a^2+b^2}e^{bx}(a\sin ax + b\cos ax) \tag{73}$$

$$\int xe^x\sin x dx = \frac{1}{2}e^x(\cos x - x\cos x + x\sin x) \tag{74}$$

$$\int xe^x\cos x dx = \frac{1}{2}e^x(x\cos x - \sin x + x\sin x) \tag{75}$$

Dynamic Hull

```
bool __slp__x__;
template<typename T>
struct HULL{
  struct node{
    T slp,x,y;
    inline node(T _slp=0,T _x=0,T _y=0){slp=_slp;x=_x;y=_y;}
    inline bool operator<(const node&a)const{return __slp__x__?slp>a.slp:x<a.x;}
    inline T operator-(const node&a)const{return(y-a.y)/(x-a.x);}
  };
  set<node>Q;
  inline void add(T x,T y){
    __slp__x__=0;
    node t(0,x,y);
    typename set<node>::iterator it=Q.lower_bound(node(0,x,0));
    if(it!=Q.end()){
      if((it->x==x&&it->y>=y)||(it->x!=x&&it->slp<=*it-t))return;
      if(it->x==x)Q.erase(it);
    }it=Q.insert(t).c0;
    typename set<node>::iterator it3=it;it3--;
    while(it!=Q.begin()){
      typename set<node>::iterator it2=it3;
      if(it2!=Q.begin()&&t-*it2>=*it2-*--it3)Q.erase(it2);
      else break;
    }it3=it;it3++;
    while(it3!=Q.end()){
      typename set<node>::iterator it2=it3;
      if(++it3!=Q.end()&&*it2-*it3>=*it2-t)Q.erase(it2);
      else break;
    }if(it==Q.begin())const_cast<T&>(it->slp)=1e9;
    else{
      typename set<node>::iterator it2=it;it2--;
      const_cast<T&>(it->slp)=t-*it2;
    }typename set<node>::iterator it2=it;it2++;
    if(it2!=Q.end())const_cast<T&>(it2->slp)=t-*it2;
  }inline pair<T,T>get(T a,T b){
    //min(ax+by)
    if(Q.empty())return mp(0,0);
    __slp__x__=1;
    typename set<node>::iterator it=Q.lower_bound(node(-a/b,0,0));
    if(it!=Q.begin())it--;
    return mp(it->x,it->y);
  }
};
```

lyndon

```
namespace lyndon{
  vector<int> work(char *s,int n){
    int i=1;vector<int> res;res.clear();
    while(i<=n){
      int j=i;
      int k=i+1;
      while(k<=n&&s[j]<=s[k]){
        if(s[j]<s[k])j=i;
        else j++;
        k++;
      }
      while(i<=j){
        res.push_back(i);
        i+=k-j;
      }
    }
    return res;
  }
};
```

SAM

```
namespace sam{
  const int N=1010000;int go[N][26],len[N],fail[N],tot,last;
  void initsam(){rep(i,1,tot){len[i]=fail[i]=0;rep(j,0,25)go[i][j]=0;}tot=last=1;}
  // Rev(S) 的后缀自动机建出来就是 S 的后缀树
  // 对于结点 x,fail[x] 到 x 的边是 right[x]-len[fail[x]]..right[x]-len[x]+1 ( 倒着的 )
  // 这个板子没有求 right, 需要的话自己写个拓扑排序求
  void add(int c,int pos){
    int p=last;int np=++tot;len[np]=len[p]+1;last=np;
    for(;p&&(!go[p][c]);p=fail[p])go[p][c]=np;
    if(!p){
      fail[np]=1;//lct::newson(1,np,pos);
      return;
    }
    int gt=go[p][c];
    if(len[p]+1==len[gt]){
      fail[np]=gt;//lct::newson(gt,np,pos);
      return;
    }
    int nt=++tot;len[nt]=len[p]+1;fail[nt]=fail[gt];fail[gt]=nt;
    //lct::cut(fail[nt],gt,nt);
    fail[np]=nt;
    //lct::newson(nt,np,pos);
```

```
    rep(i,0,25)go[nt][i]=go[gt][i];for(;p&&go[p][c]==gt;p=fail[p])go[p][c]=nt;
  }
};
```

exkmp

```
void ex_kmp(char s[], int next[], int n) {
  //s[1..next[i]]=s[i..i+next[i]-1]
    int i, a = 0, l = 0, p = 0;
    for (i = 2, next[1] = n; i <= n; ++i) {
        l = max(min(next[i - a + 1], p - i + 1), 0);
        for (; i + l <= n && s[1 + l] == s[i + l]; ++l);
        next[i] = l;
        if (i + l - 1 > p) a = i, p = i + l - 1;
    }
}
```

Manacher

```
namespace manacher{
  const int N=110000;
  char ch[N<<1],s[N];
  int f[N<<1],id,mx,n,len;
  // 以 i 为中心对应 f[i*2], 以 (i,i+1) 为中心对应 f[i*2+1]
  // f[i]-1 为回文串长度
  void init(char *s){
      n=strlen(s); ch[0]='$'; ch[1]='#';
      for (int i=1;i<=n;i++){
          ch[i*2]=s[i-1]; ch[i*2+1]='#';
      }
      id=0; mx=0; ch[n*2+2]='#';
      for (int i=0;i<=2*n+10;i++) f[i]=0;
      for (int i=1;i<=2*n+2;i++){
          if (i>mx) f[i]=1; else f[i]=min(f[id*2-i],mx-i);
          while (ch[i-f[i]]==ch[i+f[i]]) f[i]++;
          if (i+f[i]>mx){mx=i+f[i]; id=i;}
      }
  }
}
```

Maximum Express

```
// 找字典序最大的循环表示 要求 s 下标从 0 开始，并扩展到 2 倍
int lex_find(char s[], int n, bool rev) {
    int a = 0, b = 1, l;
    while (a < n && b < n) {
        for (l = 0; l < n; ++l)
            if (s[a + l] != s[b + l]) break;
        if (l < n) {
            if (s[a + l] > s[b + l]) b = b + l + 1;
            else a = a + l + 1;
            if (a == b) ++b;
        } else {
            if (a > b) swap(a, b);
            if (rev) return n - (b - a) + a;
            else return a;
        }
    }
    return min(a, b);
}
```

Suffix Array

```
namespace SA{
  const int N=110000;
  int n,m,p,x[N],y[N],c[N],sa[N],rank[N],height[N];
  char ch[N];
  void make(){
    for (int i=1;i<=m;i++) c[i]=0;
    for (int i=1;i<=n;i++) c[x[i]]++;
    for (int i=1;i<=m;i++) c[i]+=c[i-1];
    for (int i=1;i<=n;i++){
      sa[c[x[i]]]=i; c[x[i]]--;
    }
    int k=1;
    while (k<n){
      p=0;
      for (int i=n-k+1;i<=n;i++) y[++p]=i;
      for (int i=1;i<=n;i++) if (sa[i]>k) y[++p]=sa[i]-k;
      for (int i=1;i<=m;i++) c[i]=0;
      for (int i=1;i<=n;i++) c[x[i]]++;
      for (int i=1;i<=m;i++) c[i]+=c[i-1];
      for (int i=n;i;i--) sa[c[x[y[i]]]--]=y[i];
      for (int i=1;i<=n;i++) y[i]=x[i];
      p=1; x[sa[1]]=1;
      for (int i=2;i<=n;i++){
        if (y[sa[i]]!=y[sa[i-1]]||y[sa[i]+k]!=y[sa[i-1]+k]) p++; x[sa[i]]=p;
      }
      if (p==n) break; m=p; k<<=1;
    }
    for (int i=1;i<=n;i++) rank[sa[i]]=i;
    k=0;
    for (int i=1;i<=n;i++){
```

```
      if (rank[i]==1) continue;
      if (k) k--;
      while (ch[i+k]==ch[sa[rank[i]-1]+k]) k++;
      height[rank[i]]=k;
    }
  }
  void init(char *s){
    n=strlen(s);
    for (int i=0;i<=n+10;i++) height[i]=0;
    for (int i=1;i<=n;i++) x[i]=s[i-1],ch[i]=x[i]; x[n+1]=-1; ch[n+1]=-1;
    m=200; make();
  }
  // init 的时候把字符串传进去就可以了
  // sa 和 height 同定义
}
```

Ukk

```
namespace UKK{
const int maxN=1010000,inf=1e9;
struct tree{
    int l,r,go[26],father,link,d,id,e;
}t[maxN<<1];
struct state{
    int where,rem,d;
}now;
int len,S[maxN],N,en,Q[maxN],Ql,Qr;
long long ans;
// ans 表示所有边的长度和 , 统计子串个数时要减去无穷边的数量 ( 叶子节点是无穷边 )
void newnode(){
  len++; t[len].l=t[len].r=t[len].father=t[len].link=t[len].d=t[len].id=t[len].e=0;
  memset(t[len].go,0x00,sizeof t[len].go);
}
state follow(state now,int way){
    if (now.rem==0){
        if (t[now.where].go[way]==0) return (state){0,0,0};
        else {
            int k1=t[now.where].go[way]; return (state){k1,t[k1].r-t[k1].l-1,now.d+1};
        }
    } else if (S[t[now.where].r-now.rem]==way) return (state){now.where,now.rem-1,now.d+1};
    else return (state){0,0,0};
}
state go(int now,int l,int r){
    while (l<r){
        int k1=t[now].go[S[l]];
        if (t[k1].r-t[k1].l>=r-l) return (state){k1,t[k1].r-t[k1].l-(r-l),t[now].d+(r-l)};
        now=k1; l+=t[k1].r-t[k1].l;
    }
    return (state){now,0,t[now].d};
}
void change(int l,int r,int pre,int where){
    t[where].father=pre; t[where].l=l; t[where].r=r;
    t[where].d=t[pre].d+r-l; t[pre].e+=(t[pre].go[S[l]]==0);
    t[pre].go[S[l]]=where;
    if (r==inf) t[where].id=inf-t[where].d,t[pre].id=max(t[pre].id,t[where].id);
}
int splite(state now){
    if (now.rem==0) return now.where;
    if (now.rem==t[now.where].r-t[now.where].l) return t[now.where].father;
    newnode();
    change(t[now.where].l,t[now.where].r-now.rem,t[now.where].father,len);
    int k1=S[t[len].r]; t[len].go[k1]=now.where; t[now.where].father=len;
    t[now.where].l=t[len].r; t[len].id=t[now.where].id; t[len].e=1;
    return len;
}
int getlink(int k){
    if (t[k].link) return t[k].link;
    t[k].link=splite(go(getlink(t[k].father),t[k].l+(t[k].father==1),t[k].r));
    return t[k].link;
}
void insert(int k){ // push back
    S[++N]=k;
    while (1){
        state ne=follow(now,k);
        if (ne.where){
            now=ne; return;
        }
        int mid=splite(now);
        newnode(); int leaf=len;
        change(N,inf,mid,leaf); int newnod=getlink(mid);
        now.where=newnod; now.rem=0; now.d=t[newnod].d;
        Q[++Qr]=leaf; ans+=t[leaf].r-t[leaf].l; en++;
        if (mid==1) return;
    }
}
long long getsubstring(){
    return ans-1ll*(inf-N-1)*en;
}
```

```
void del(){ // pop front
    Ql++;
    int where=Q[Ql];
    ans-=t[where].r-t[where].l; if (t[where].r==inf) en--;
    if (where==now.where){
        now=go(getlink(t[where].father),t[where].l+(t[where].father==1),t[where].r-now.rem);
        int prel=t[where].l;
        t[where].l=Qr+t[t[where].father].d+1; t[where].d=t[t[where].father].d+t[where].r-t[where].l;
        t[where].id=Qr+1; if (where==now.where) now.rem+=prel-t[where].l;
        ans+=t[where].r-t[where].l; Q[++Qr]=where; en+=(t[where].r==inf);
        return;
    }
    t[t[where].father].go[S[t[where].l]]=0; t[t[where].father].e--;
    if (t[t[where].father].e==1&&t[where].father!=1){
        int newson=0,r=t[where].father,rr=t[r].father;
        for (int i=0;i<26;i++) if (t[r].go[i]) newson=t[t[where].father].go[i];
        int pre=t[newson].r-t[newson].l;
        int newl=t[newson].r-t[newson].l+t[r].r-t[r].l;
        if (t[newson].e)
            change(t[newson].id+t[rr].d,t[newson].id+t[rr].d+newl,rr,newson);
        else change(t[newson].id+t[rr].d,inf,rr,newson);
        if (now.where==r) now.where=newson,now.rem+=pre;
    }
}
void init(){
  len=0; newnode(); now=(state){1,0,0}; t[1].link=1; en=0; Ql=Qr=0; N=0; ans=0;
}
}
```

回文树

```
const int M=26;int fail[N];
int go[N][M],len[N],diff[N],anc[N],lst;
int n;char str[N];int p;int s[N];int f[N],g[N];
void addChar(int c,int ww){
  int x=lst;while(s[ww]!=s[ww-len[x]-1])x=fail[x];// ww 是位置 , 下标从 1 开始
  if(!go[x][c]){
    len[p]=len[x]+2;int k=fail[x];while(s[ww]!=s[ww-len[k]-1])k=fail[k];fail[p]=go[k][c];
    go[x][c]=p;diff[p]=len[p]-len[fail[p]];
    if(diff[p]==diff[fail[p]])anc[p]=anc[fail[p]]; // anc[x] 表示祖先中 , 第一个和 x 不在一个等差数列里的回文串
    else anc[p]=fail[p];p++;
  }
  lst=go[x][c]; // 求长度 , 直接倒着 for 一遍即可
}
void init(){
  rep(i,1,p){anc[i]=diff[i]=len[i]=fail[i]=0;rep(j,0,M-1)go[i][j]=0;}
  p=2;len[0]=0;len[1]=-1;fail[0]=1; //node 1: 所有奇数长度的串的祖先
  fail[1]=0;f[0]=1;lst=1;
}
void work(){
  s[0]=-1;init(); //s[0] 要设成字符集之外的数
  rep(i,1,n){
    addChar(s[i],i);
    for(int x=lst;x;x=anc[x]){
      g[x]=f[i-(len[anc[x]]+diff[x])]; //g[x] 记录包含 x 的等差数列链的信息 (x 一定是链底 )
      if(anc[x]!=fail[x])g[x]=(g[x]+g[fail[x]])%P;
      if(i%2==0)f[i]=(f[i]+g[x])%P;
      /*  当新加入一个字符 , 扩展整个链时 .g[x] 被扩展到时 ,fail[x] 上一次被扩展到一定是在 i-d 时
        假设这一段等差数列里的长度是 l[1]..l[m],l[m] 是 g[x] 代表的点 , 则 g[fail[x]]=sum(j=1..m-1)f[i-d-l[j]]
        而 g[x]=sum(j=1..m)f[i-l[j]]=f[i-l[1]]+sum(j=2..m)f[i-l[j-1]-d]=f[i-l[1]]+g[fail[x]]=f[i-(len[anc[x]]+diff[x])]+g[fail[x]]
        可以认为 ,g[x] 是在 g[fail[x]] 的基础上 , 添加了 i-l[1] 这个左边界
        注意 , 如果是维护其他的信息 , 注意把 g[x] 以前的贡献去除掉 */
    }
  }
}
```

KM

```
namespace KM{
typedef long long i64;
const int maxN = 401;
const int oo = 0x3f3f3f3f;
int vx[maxN],vy[maxN],lx[maxN],ly[maxN],slack[maxN];
int w[maxN][maxN]; // 以上为权值类型
int pre[maxN],left[maxN],right[maxN],NL,NR,N;
void match(int& u) {
  for(;u; std::swap(u, right[pre[u]]))
    left[u] = pre[u];
}
void bfs(int u) {
  static int q[maxN], front, rear;
  front = 0; vx[q[rear = 1] = u] = true;
  while(true) {
    while(front < rear) {
```

```
      int u = q[++front];
      for(int v = 1;v <= N; ++v) {
        int tmp;
        if(vy[v] || (tmp = lx[u] + ly[v] - w[u][v]) > slack[v])
          continue;
        pre[v] = u;
        if(!tmp) {
          if(!left[v]) return match(v);
          vy[v] = vx[q[++rear] = left[v]] = true;
        } else slack[v] = tmp;
      }
    }

    int a = oo;
    for(int i = 1;i <= N; ++i)
      if(!vy[i] && a > slack[i]) a = slack[u = i];
    for(int i = 1;i <= N; ++i) {
      if(vx[i]) lx[i] -= a;
      if(vy[i]) ly[i] += a;
      else slack[i] -= a;
    }
    if(!left[u]) return match(u);
    vy[u] = vx[q[++rear] = left[u]] = true;
  }
}
void exec() {
  for(int i = 1;i <= N; ++i) {
    for(int j = 1;j <= N; ++j) {
      slack[j] = oo;
      vx[j] = vy[j] = false;
    }
    bfs(i);
  }
}
i64 work(int nl,int nr){// NL , NR 为左右点数 , 返回最大权匹配的权值和
  NL=nl;NR=nr;
  N=std::max(NL,NR);
  for(int u = 1;u <= N; ++u)
    for(int v = 1;v <= N; ++v){
      lx[u] = std::max(lx[u], w[u][v]);
    }

  exec();

  i64 ans = 0;
  for(int i = 1;i <= N; ++i)
    ans += lx[i] + ly[i];
  return ans;
}
void output(){ // 输出左边点与右边哪个点匹配 , 没有匹配输出 0
  for(int i = 1;i <= NL; ++i)
    printf("%d ",(w[i][right[i]] ? right[i] : 0));
  printf("\n");
}
}
```

带花树

```
namespace KM{
typedef long long i64;
const int maxN = 401;
const int oo = 0x3f3f3f3f;
int vx[maxN],vy[maxN],lx[maxN],ly[maxN],slack[maxN];
int w[maxN][maxN]; // 以上为权值类型
int pre[maxN],left[maxN],right[maxN],NL,NR,N;
void match(int& u) {
  for(;u; std::swap(u, right[pre[u]]))
    left[u] = pre[u];
}
void bfs(int u) {
  static int q[maxN], front, rear;
  front = 0; vx[q[rear = 1] = u] = true;
  while(true) {
    while(front < rear) {
      int u = q[++front];
      for(int v = 1;v <= N; ++v) {
        int tmp;
        if(vy[v] || (tmp = lx[u] + ly[v] - w[u][v]) > slack[v])
          continue;
        pre[v] = u;
        if(!tmp) {
          if(!left[v]) return match(v);
          vy[v] = vx[q[++rear] = left[v]] = true;
        } else slack[v] = tmp;
      }
    }

    int a = oo;
    for(int i = 1;i <= N; ++i)
      if(!vy[i] && a > slack[i]) a = slack[u = i];
    for(int i = 1;i <= N; ++i) {
```

```
      if(vx[i]) lx[i] -= a;
      if(vy[i]) ly[i] += a;
      else slack[i] -= a;
    }
    if(!left[u]) return match(u);
    vy[u] = vx[q[++rear] = left[u]] = true;
  }
}
void exec() {
  for(int i = 1;i <= N; ++i) {
    for(int j = 1;j <= N; ++j) {
      slack[j] = oo;
      vx[j] = vy[j] = false;
    }
    bfs(i);
  }
}
i64 work(int nl,int nr){// NL , NR 为左右点数 , 返回最大权匹配的权值和
  NL=nl;NR=nr;
  N=std::max(NL,NR);
  for(int u = 1;u <= N; ++u)
    for(int v = 1;v <= N; ++v){
      lx[u] = std::max(lx[u], w[u][v]);
    }

  exec();

  i64 ans = 0;
  for(int i = 1;i <= N; ++i)
    ans += lx[i] + ly[i];
  return ans;
}
void output(){ // 输出左边点与右边哪个点匹配 , 没有匹配输出 0
  for(int i = 1;i <= NL; ++i)
    printf("%d ",(w[i][right[i]] ? right[i] : 0));
  printf("\n");
}
}
```

2-sat

```
namespace Twosat{
  const int M=4000010,N=1000010;
  struct bian{
    int next,point;
  }b[M];
  int n,len,p[N<<1],dfs[N<<1],low[N<<1],where[N<<1],now,sign;
  int s[N<<1],head,in[N<<1],ans[N],ou[N<<1],sign2;
  void add(int k1,int k2){
    b[++len]=(bian){p[k1],k2}; p[k1]=len;
  }
  // n 表示限制个数 ,i 为取 ,i+n 为不取
  // 连边一定要对称
  void init(int _n){
    n=_n;
    for (int i=0;i<=n*2+1;i++){
      p[i]=where[i]=dfs[i]=low[i]=s[i]=in[i]=ou[i]=0;
    }
    len=now=sign=head=sign2=0;
  }
  void tarjan(int k1){
    s[++head]=k1; in[k1]=1; dfs[k1]=++sign; low[k1]=sign;
    for (int i=p[k1];i;i=b[i].next){
      int j=b[i].point;
      if (dfs[j]==0){
        tarjan(j); low[k1]=min(low[k1],low[j]);
      } else if (in[j]) low[k1]=min(low[k1],dfs[j]);
    }
    if (dfs[k1]==low[k1]){
      now++;
      while (1){
        where[s[head]]=now; in[s[head]]=0;
        ou[s[head]]=++sign2; head--;
        if (s[head+1]==k1) break;
      }
    }
  }
  // ans[i]=0 表示取 i, 否则表示取 i+n
  int solve(){
    for (int i=1;i<=n*2;i++) if (dfs[i]==0) tarjan(i);
    for (int i=1;i<=n;i++) if (where[i]==where[i+n]) return 0;
    for (int i=1;i<=n;i++) if (ou[i]<ou[i+n]) ans[i]=0; else ans[i]=1;
    return 1;
  }
}
```

Cactus

```
namespace Cactus{
  const int NN=51000,M=101000;
  struct bian{
    int next,point;
```

```
}b[M<<1];
int p[NN],len,n,m,pd[M<<1],father[NN],d[NN];
int N,u[M],v[M],A[NN+M];
vector<int>go[NN+M];
// A 每一个环最上方的节点
// 将仙人掌建成圆方树 , 编号 >n 的节点为环 ,d 为深度
// 把边 add 进去之后调用 buildtree, 树存在 go 中
// 如果不一定是仙人掌则要事先判断 , 不然复杂度会爆炸
void init(int _n){
  n=_n; len=-1;
  memset(p,0xff,sizeof p);
  memset(pd,0x00,sizeof pd);
  memset(d,0x00,sizeof d);
  memset(father,0x00,sizeof father);
  for (int i=1;i<=n+m;i++) go[i].clear();
}
void ade(int k1,int k2){
  b[++len]=(bian){p[k1],k2}; p[k1]=len;
}
void Add(int k1,int k2){
  ade(k1,k2); ade(k2,k1);
}
void add(int k1,int k2){
  m++; u[m]=k1; v[m]=k2; Add(k1,k2);
}
void dfs(int k1,int k2){
  father[k1]=k2; d[k1]=d[k2]+1;
  for (int i=p[k1];i!=-1;i=b[i].next){
    int j=b[i].point;
    if (d[j]==0){
      pd[(i>>1)+1]=1; dfs(j,k1);
    }
  }
}
int compare(int k1,int k2){
  return d[k1]<d[k2];
}
void buildtree(){
  dfs(1,0); N=n;
  for (int i=1;i<=m;i++)
    if (pd[i]==0){
      int k1=u[i],k2=v[i]; N++;
      if (d[k1]<d[k2]) swap(k1,k2); A[N]=k2;
      while (k1!=k2){
        // 不是仙人掌的话在这儿判一下每一条边只被覆盖一次
          int pre=father[k1]; father[k1]=N; k1=pre;
        }
        go[k2].push_back(N);
      }
    for (int i=2;i<=n;i++){
      go[father[i]].push_back(i);
    }
  }
}
```

点双

```
namespace CC{
// 先调用 init() 点双数 num
// 点双树在 go 中 , 每一个点双向割点连边 , pd>1 为割点 , pd=0 为孤立点
const int N=110000;
struct bian{
  int next,point;
}b[N<<1];
vector<int>E[N<<1],V[N<<1],go[N<<1];
int num,s[N<<1],head,pd[N],dfs[N],low[N],sign,loc[N];
int p[N],len,n;
void ade(int k1,int k2){b[++len]=(bian){p[k1],k2}; p[k1]=len;}
void add(int k1,int k2){ade(k1,k2); ade(k2,k1);}
void solve(int k1){
  dfs[k1]=++sign; low[k1]=dfs[k1];
  for (int i=p[k1];i!=-1;i=b[i].next){
    int j=b[i].point;
    if (dfs[j]==0){
      s[++head]=i; solve(j); low[k1]=min(low[k1],low[j]);
      if (low[j]==dfs[k1]){
        num++;
        while (1){
          E[num].push_back(s[head]); head--;
          if (s[head+1]==i) break;
        }
      }
    } else if (dfs[j]<dfs[k1])
      s[++head]=i,low[k1]=min(low[k1],dfs[j]);
  }
}
void work(){
  for (int i=1;i<=n;i++) if (dfs[i]==0) solve(i);
  sign=0;
  for (int i=1;i<=num;i++){
    sign++;
```

```
    for (int j=0;j<E[i].size();j++){
      int k1=b[E[i][j]].point;
      if (pd[k1]!=sign){
        pd[k1]=sign; V[i].push_back(k1);
      }
    }
  }
  for (int i=1;i<=num;i++)
    for (int j=0;j<V[i].size();j++) pd[V[i][j]]++;
  for (int i=1;i<=num;i++)
    for (int j=0;j<V[i].size();j++)
      if (pd[V[i][j]]>1){
        go[V[i][j]].push_back(i+n);
        go[i+n].push_back(V[i][j]);
      } else loc[V[i][j]]=i;
}
void init(int _n){
  n=_n; sign=head=0; len=-1; num=0;
  for (int i=1;i<=n;i++) pd[i]=dfs[i]=low[i]=loc[i]=0;
  for (int i=1;i<=n*2;i++) go[i].clear(),V[i].clear(),E[i].clear();
}
}
```

zkw

```
namespace Flow{
  const int M=100010,N=1010,inf=1e9;
  struct bian{
    int next,point,f,w;
  }b[M];
  int totpoint,p[N],len,n,m,D[N],pd[N],sign,flow,cost,bo[N];
  // D 为顶标 , 对残量网络满足最短路那个不等式
  void ade(int k1,int k2,int k3,int k4){
    b[++len]=(bian){p[k1],k2,k3,k4}; p[k1]=len;
  }
  void add(int k1,int k2,int k3,int k4){
    n=max(n,k1); n=max(n,k2);
    ade(k1,k2,k3,k4); ade(k2,k1,0,-k4);
  }
  void init(int _totpoint){
    memset(p,0xff,sizeof p); len=-1; flow=0; cost=0;
    memset(D,0x00,sizeof D);
    totpoint=_totpoint; n=totpoint;
  }
  int dfs(int k1,int k2){
    pd[k1]=sign;
    if (k1==totpoint||k2==0) return k2;
    for (int i=p[k1];i!=-1;i=b[i].next){
      int j=b[i].point;
      if (b[i].f&&D[j]==D[k1]+b[i].w&&pd[j]!=sign){
        int k=dfs(j,min(k2,b[i].f));
        if (k==0) continue;
        b[i].f-=k; b[i^1].f+=k;
        cost+=k*b[i].w;
        return k;
      }
    }
    return 0;
  }
  int newD(){
    if (pd[totpoint]==sign) return 1;
    int w=inf;
    for (int now=0;now<=n;now++)
      if (pd[now]==sign)
        for (int i=p[now];i!=-1;i=b[i].next){
          int j=b[i].point;
          if (b[i].f&&pd[j]!=sign) w=min(w,D[now]-D[j]+b[i].w);
        }
    if (w==inf) return 0;
    for (int i=0;i<=n;i++) if (pd[i]==sign) D[i]-=w;
    return 1;
  }
  void get(){
    do{
      sign++; flow+=dfs(0,inf);
    }while(newD());
  }
}
```

最小树形图

```
namespace ZL{
  // a 尽量开大 , 之后的边都塞在这个里面
  const int N=100010,M=100010,inf=1e9;
  struct bian{
      int u,v,w,use,id;
  }b[M],a[2000100];
  int n,m,ans,pre[N],id[N],vis[N],root,In[N],h[N],len,way[M];
  // 从 root 出发能到达每一个点的最小支撑树
  // 先调用 init 然后把边 add 进去 , 需要方案就 getway,way[i] 为 1 表示使用
  void init(int _n,int _root){
    n=_n; m=0; b[0].w=1e9; root=_root;
```

```
  }
  void add(int u,int v,int w){
    m++; b[m]=(bian){u,v,w,0,m}; a[m]=b[m];
  }
  int work(){
    len=m;
      for (;;){
          for (int i=1;i<=n;i++){pre[i]=0; In[i]=inf; id[i]=0; vis[i]=0; h[i]=0;}
          for (int i=1;i<=m;i++) if (b[i].u!=b[i].v&&b[i].w<In[b[i].v]){
              pre[b[i].v]=b[i].u; In[b[i].v]=b[i].w; h[b[i].v]=b[i].id;
          }
          for (int i=1;i<=n;i++) if (pre[i]==0&&i!=root) return 0;
          int cnt=0; In[root]=0;
          for (int i=1;i<=n;i++){
              if (i!=root) a[h[i]].use++; int now=i; ans+=In[i];
              while (vis[now]==0&&now!=root){ vis[now]=i; now=pre[now]; }
              if (now!=root&&vis[now]==i){
                  cnt++; int kk=now;
                  while (1){
                      id[now]=cnt; now=pre[now];
                      if (now==kk) break;
                  }
              }
          }
          if (cnt==0) return 1; for (int i=1;i<=n;i++) if (id[i]==0) id[i]=++cnt;
          // 缩环 , 每一条接入的边都会茶包原来接入的那条边 , 所以要调整边权
          // 新加的边是 u, 茶包的边是 v
          for (int i=1;i<=m;i++){
              int k1=In[b[i].v]; int k2=b[i].v; b[i].u=id[b[i].u]; b[i].v=id[b[i].v];
              if (b[i].u!=b[i].v){
                  b[i].w-=k1; a[++len].u=b[i].id; a[len].v=h[k2]; b[i].id=len;
              }
          }
          n=cnt; root=id[root];
      }
      return 1;
  }
  void getway(){
    for (int i=1;i<=m;i++) way[i]=0;
    for (int i=len;i>m;i--){ a[a[i].u].use+=a[i].use; a[a[i].v].use-=a[i].use; }
    for (int i=1;i<=m;i++) way[i]=a[i].use;
  }
}
```

Diameter Tree

```
//Floyd First
for(i=-1;++i!=n;) {
  for(j=-1;++j!=n;r[j][0]=dis[i][r[j][1]]);
  qsort(r,n,8,cmp);
  for(j=i;++j!=n;)
    if(map[i][j]!=0x3F3F3F3F) {
      for(d=x=0;++x!=n;)
        if(dis[j][r[x][1]]>dis[j][r[d][1]])
          ans=min(ans,map[i][j]+dis[i][r[x][1]]+dis[j][r[d][1]]),d=x;
    if(!d) ans=min(ans,min(dis[j][r[0][1]],r[0][0])<<1);
  }
}
```

Dominator Tree

```
namespace dominator{
  // DAG 的 dominator tree 可以直接 LCA 做
  // 最开始先 init() 传入点数 , 通过 add 加边 , 出发点编号为 1 可能需要重标号
  // dominator tree 的结构存在 go 中 , 可能存在点无法到达即不在树中
  // go 中的下标是根据 dfs 序重标号过的
  // semi i 的祖先 x, 不经过 i 到 x 之间树上的点能到达 i 的最高祖先
  const int N=110000,M=1010000;
  struct bian{
      int next,point;
  }b[M];
  int dfs[N],x[N],p[N],len,pre[N];
  int idom[N],best[N],semi[N],f[N];
  vector<int>go[N];
  void ade(int k1,int k2){
      b[++len]=(bian){p[k1],k2}; p[k1]=len;
  }
  void add(int k1,int k2){
      ade(k1,k2); ade(k2,k1);
  }
  // 先通过一次 dfs 给所有点标号 , 如果已经给出了标号这一步可以省略
  void solve(int k){
      dfs[k]=++len; x[len]=k;
      for (int i=p[k];i!=-1;i=b[i].next){
          int j=b[i].point; if (i&1) continue;
          if (dfs[j]==0) {solve(j); pre[dfs[j]]=dfs[k];}
      }
  }
  int get(int k){
      if (k==f[k]) return k;
      int k1=get(f[k]);
```

```
      if (semi[best[k]]>semi[best[f[k]]]) best[k]=best[f[k]];
      f[k]=k1; return f[k];
  }
  void tarjan(){
      for (int now=len;now>=2;now--){
          int k1=x[now];
          for (int i=p[k1];i!=-1;i=b[i].next){
              if ((i&1)==0) continue;
              int j=dfs[b[i].point];
              if (j==0) continue; get(j);
              if (semi[best[j]]<semi[now]) semi[now]=semi[best[j]];
          }
          go[semi[now]].push_back(now);
          int k2=pre[now]; f[now]=pre[now];
          for (int i=0;i<go[k2].size();i++){
              int j=go[k2][i];
              get(j);
              if (semi[best[j]]<k2) idom[j]=best[j]; else idom[j]=k2;
          }
          go[k2].clear();
      }
      for (int i=2;i<=len;i++){
          if (semi[i]!=idom[i]) idom[i]=idom[idom[i]];
          go[idom[i]].push_back(i);
      }
  }
  void init(int n){
    len=-1;
    for (int i=1;i<=n;i++){
      p[i]=-1,f[i]=best[i]=semi[i]=i,go[i].clear();
      idom[i]=0,pre[i]=x[i]=dfs[i]=0;
    }
  }
  void getdominator(){
    len=0; solve(1); tarjan();
  }
}
```

SS-algorithm

```
const int N=55;
int n,m;
struct perm{
    int p[N];
    inline perm(int ise=0){rep(i,1,n)p[i]=i*ise;}
    inline perm inv(){perm res;rep(i,1,n)res.p[p[i]]=i;return res;}
};
vector<perm> T[N];perm R[N][N];
inline int sz(int x){int ans=0;rep(i,1,n)ans+=R[x][i].p[1]>0;return ans;}
inline perm operator *(const perm &a,const perm &b){
    perm c;rep(i,1,n)c.p[i]=a.p[b.p[i]];return c;
}
//------------------permutation----------------------------------
bool check(perm x,int k){
    //check if x in <S>
    return (!k)||(R[k][x.p[k]].p[1]&&check(R[k][x.p[k]]*x,k-1));
}
void dfs(perm x,int k);
void insert(perm x,int k){//insert(x,n)
    if(check(x,k))return;
    T[k].push_back(x);
    rep(i,1,n)if(R[k][i].p[1])dfs(x*R[k][i].inv(),k);
}
void dfs(perm x,int k){
    if(R[k][x.p[k]].p[1])insert(R[k][x.p[k]]*x,k-1);
    else{
        R[k][x.p[k]]=x.inv();
        rep(i,0,T[k].size()-1)dfs(T[k][i]*x,k);
    }
}
void init(){
    rep(i,1,n)rep(j,1,n)R[i][j]=perm(i==j);rep(i,1,n)T[i].clear();
}
```

洲阁筛

```
namespace Sieve{
  const int N=100000;//sqrt(N)
  const int S=100000;
  int inv[55];
  int vf(int x){return inv[2];}//V(p)
  int vg(int x,int c){return inv[c+1];}//V(p^c) (c>1)
  int Val(int p,int c){if(c==1)return vf(p);else return vg(p,c);}//V(p^c) (c>=1)
  bool notp[N+10];int
   ↪ pr[N+10],prtot,w[N+10],m,pos[N+10],n,pre[N+10],small[N+10],f[N+10],g[N+10];
  int fd(int x){//find the largest prime that is no more than x
    int l=1;int r=prtot;int ret=0;
    while(l<r){int mid=(l+r)>>1;if(pr[mid]<=x)ret=mid,l=mid+1;else r=mid;}
    if(pr[l]<=x)ret=l;return ret;
  }
  int preSV(int p){return p*1ll*inv[2]%P;}//sum(x=1..p)V(pr[x])
  int sumF(int l,int r){return (r-l+1);}//sum(x=1..p)F(x)
```

```
int sumV(int l,int r){if(l>r)return 0;return (preSV(fd(r))+P-preSV(fd(l-1)))%P;}//sum(x=l..r&x is prime)V(x)
int sumV2(int l,int r){if(l>r)return 0;return (preSV(r)+P-preSV(l-1))%P;}//sum(x=l..r)V(pr[x])
int sumV3(int l,int r){if(l>r)return 0;return (r-l+1)%P;}//sum(x=l..r)F(pr[x])
int vfg(int x){return 1;}//F(x)
int getPos(int x){if(x<=S)return pos[x];else return m+1-pos[n/x];}
int getVal(int x,int t){return (g[x]+P-sumV3(pre[x]+1,min(t-1,small[x])))%P;}
void Main(int _n){
  rep(i,1,50)inv[i]=Pow(i,P-2);n=_n;
  for(int i=2;i<=N;++i){
    if(!notp[i])pr[++prtot]=i;
    for(int j=1;j<=prtot&&pr[j]*1ll*i<=N;++j){
      notp[i*pr[j]]=1;if(i%pr[j]==0)break;
    }
  }
  for(int i=1;i<=n;i=n/(n/i)+1)w[++m]=n/i;
  sort(w+1,w+1+m);rep(i,1,m)if(w[i]<=S)pos[w[i]]=i;
  f[getPos(n)]=1;int up=1;int ans=0;
  rep(i,1,m){small[i]=small[i-1];while(small[i]<prtot&&pr[small[i]+1]<=w[i])++small[i];}
  rep(i,1,prtot){
    int nup=up;
    rep(j,up,m){
      if(pr[i]>w[j]){
        nup=max(nup,j+1);continue;
      }
      if(pr[i]*pr[i]>w[j]){
        nup=max(nup,j+1);int res=f[j];res=res*1ll*sumV(pr[i],w[j])%P;ans=(ans+res)%P;continue;
      }
      for(int v=w[j]/pr[i],c=1;v;v/=pr[i],c++){
        int y=getPos(v);f[y]=(f[y]+f[j]*1ll*Val(pr[i],c))%P;
        if(pr[i]*pr[i]>w[y]){
          int s=f[j]*1ll*Val(pr[i],c)%P;s=s*1ll*sumV2(i+1,small[y])%P;ans=(ans+s)%P;
        }
      }
    }
    up=nup;
  }
  //G must meet G(ab)=G(a)G(b)
  rep(i,1,m)g[i]=sumF(1,w[i]);up=1;
  rep(i,1,prtot){
    int nup=up;
    per(j,m,up){
      if(pr[i]>w[j]){
        nup=max(nup,j+1);g[j]=1;pre[j]=i;continue;
      }
      g[j]=(g[j]+P-(vfg(pr[i])*1ll*getVal(getPos(w[j]/pr[i]),i)%P))%P;
      if(pr[i]*pr[i]>w[j]){nup=max(nup,j+1);pre[j]=i;continue;}
      pre[j]=i;
    }
    up=nup;
  }
  rep(i,1,m)g[i]=getVal(i,prtot+1);
  rep(i,1,m)ans=(ans+f[i]*1ll*(1+(g[i]+P-1)*1ll*inv[2]%P))%P;//need modify
  printf("%d\n",ans);
}
};
```

fft

```
#define upmo(a,b) (((a)=((a)+(b))%mo)<0?(a)+=mo:(a))
const db pi=3.14159265358979323846264338327L;const int FFT_MAXN=262144;int mo=2;
struct cp{
  db a,b;
  cp operator +(const cp&y)const{return (cp){a+y.a,b+y.b};}
  cp operator -(const cp&y)const{return (cp){a-y.a,b-y.b};}
  cp operator *(const cp&y)const{return (cp){a*y.a-b*y.b,a*y.b+b*y.a};}
  cp operator !()const{return cp{a,-b};};
}nw[FFT_MAXN+1];
int bitrev[FFT_MAXN];
void dft(cp*a,int n,int flag=1){
  int d=0;while((1<<d)*n!=FFT_MAXN)d++;
  rep(i,0,n-1)if(i<(bitrev[i]>>d))swap(a[i],a[bitrev[i]>>d]);
  for(int l=2;l<=n;l<<=1){
    int del=FFT_MAXN/l*flag;
    for(int i=0;i<n;i+=l){
      cp *le=a+i;cp *ri=a+i+(l>>1);
      cp *w=flag==1?nw:nw+FFT_MAXN;
      rep(k,0,(l>>1)-1){
        cp ne=*ri**w;*ri=*le-ne,*le=*le+ne;le++,ri++,w+=del;
      }
    }
  }
  if(flag!=1)rep(i,0,n-1)a[i].a/=n,a[i].b/=n;
}
void fft_init(){
  int L=0;while((1<<L)!=FFT_MAXN)L++;
  bitrev[0]=0;rep(i,1,FFT_MAXN-1)bitrev[i]=bitrev[i>>1]>>1|((i&1)<<(L-1));
```

```cpp
  rep(i,0,FFT_MAXN)nw[i]=(cp){(db)cosl(2*pi/FFT_MAXN*i),(db)sinl(2*pi/FFT_MAXN*i)};
}
void convoP(int *a,int n,int *b,int m,int *c){ // 任意模数 fft, 需要提前设定 mo
  rep(i,0,n+m)c[i]=0;
  static cp f[FFT_MAXN],g[FFT_MAXN],t[FFT_MAXN];int N=2;while(N<=n+m)N<<=1;
  rep(i,0,N-1){
    int aa=i<=n?a[i]:0;int bb=i<=m?b[i]:0;
    upmo(aa,0);upmo(bb,0);
    f[i]=(cp){db(aa>>15),db(aa&32767)};
    g[i]=(cp){db(bb>>15),db(bb&32767)};
  }
  dft(f,N);dft(g,N);
  rep(i,0,N-1){int j=i?N-i:0;t[i]=((f[i]+!f[j])*(!g[j]-g[i])+(!f[j]-
   ↪ f[i])*(g[i]+!g[j]))*(cp){0,0.25};}
  dft(t,N,-1);
  rep(i,0,n+m)upmo(c[i],(ll(t[i].a+0.5))%mo<<15);
  rep(i,0,N-1){int j=i?N-i:0;t[i]=(!f[j]-f[i])*(!g[j]-g[i])*(cp){-
   ↪ 0.25,0}+(cp){0,0.25}*(f[i]+!f[j])*(g[i]+!g[j]);}
  dft(t,N,-1);
  rep(i,0,n+m)upmo(c[i],ll(t[i].a+0.5)+(ll(t[i].b+0.5)%mo<<30));
}
void convoF(int *a,int n,int *b,int m,int *c,int P){ // 快速的 fft
  static cp f[FFT_MAXN>>1],g[FFT_MAXN>>1],t[FFT_MAXN>>1];
  int N=2; while (N<=n+m) N<<=1;
  rep(i,0,N-1){
    if (i&1){
      f[i>>1].b=(i<=n)?a[i]:0.0;g[i>>1].b=(i<=m)?b[i]:0.0;
    } else {
      f[i>>1].a=(i<=n)?a[i]:0.0;g[i>>1].a=(i<=m)?b[i]:0.0;
    }
  }
  dft(f,N>>1); dft(g,N>>1);int del=FFT_MAXN/(N>>1);
  cp qua=(cp){0,0.25},one=(cp){1,0},four=(cp){4,0},*w=nw;
  rep(i,0,(N>>1)-1){
    int j=i?(N>>1)-i:0;
    t[i]=(four*!(f[j]*g[j])-(!f[j]-f[i])*(!g[j]-g[i])*(one+*w))*qua;
    w+=del;
  }
  dft(t,N>>1,-1);
  rep(i,0,n+m) c[i]=((long long)(((i&1)?t[i>>1].a:t[i>>1].b)+0.5))%P;
}
```

ntt

```cpp
const int P=998244353;
const int G=3;const int N=(1<<22)+5;
int rev[N],w[2][N];
inline void init(int n){
    rep(i,0,n-1){
        int x=0;int y=i;for(int k=1;k<n;k<<=1,y>>=1)(x<<=1)|=(y&1);rev[i]=x;
    }
    w[0][0]=w[1][0]=1;int cha=Pow(G,(P-1)/n);int cha2=Pow(cha,P-2);
    rep(i,1,n-1){
        w[0][i]=w[0][i-1]*1ll*cha%P;
        w[1][i]=w[1][i-1]*1ll*cha2%P;
    }
}
inline void NTT(int *A,int N,bool ms){
    for(int i=0;i<N;i++)if(i<rev[i]){
        int tmp=A[i];A[i]=A[rev[i]];A[rev[i]]=tmp;
    }
    for(int i=1;i<N;i<<=1){
        for(int j=0;j<N;j+=(i<<1)){
            for(int k=0,l=0;k<i;k++,l+=N/(i<<1)){
                int x,y;y=A[j+k];x=A[j+k+i]*1ll*w[ms][l]%P;
                A[j+k]=(x+y)%P;A[j+k+i]=(y-x+P)%P;
            }
        }
    }
    if(ms){
        int v=Pow(N,P-2);rep(i,0,N-1)A[i]=A[i]*1ll*v%P;
    }
}
```

BM

```cpp
namespace BM{
  const int mo=1e9+7,L=31000; const long long N=5ll*mo*mo;
  int x[L],y[L],len,prelen,prep,A[L],n,z[L],prew;
  // 依次加入 A[i], 找到长度为 len 的递推式, 其中 sum A[j-len+i]*x[i]=0
  // 时间复杂度 O(n^2), 插入直接 addin(), 输出 x 数组即可
  // 求行列式可以随机两个向量乘成数列 , 然后利用这个把特征多项式求出来
  int check(int n){
    long long w=0;
    for (int i=0;i<=len;i++){
      w=(w+1ll*A[n-len+i]*x[i]); if (w>N) w-=N;
    }
    return w%mo;
  }
  int quick(int k1,int k2){
    int k3=1;
    while (k2){
```

```
      if (k2&1) k3=1ll*k3*k1%mo; k2>>=1; k1=1ll*k1*k1%mo;
    }
    return k3;
  }
  void addin(int k1){
    A[++n]=k1; int num=check(n); if (num==0) return;
    int last=prep-prelen,now=n-len,kk=1ll*prew*num%mo;
    if (now<=last){
      for (int i=last-now;i<=prelen+last-now;i++){
        x[i]=(x[i]-1ll*y[i-last+now]*kk)%mo; if (x[i]<0) x[i]+=mo;
      }
      return;
    }
    for (int i=0;i<=len;i++) z[i]=x[i];
    int shi=now-last;
    for (int i=len;i>=0;i--) x[i+shi]=x[i];
    for (int i=0;i<shi;i++) x[i]=0;
    for (int i=0;i<=prelen;i++){
      x[i]=(x[i]-1ll*y[i]*kk)%mo; if (x[i]<0) x[i]+=mo;
    }
    prelen=len; prep=n; prew=quick(num,mo-2); for (int i=0;i<=len;i++) y[i]=z[i];
    len+=shi;
  }
  void init(){
    memset(x,0x00,sizeof x); memset(y,0x00,sizeof y);
    memset(z,0x00,sizeof z); memset(A,0x00,sizeof A);
    prelen=0; y[0]=1; prep=0; len=0; x[0]=1; n=0; prew=0;
  }
};
```

Pollard Rho

```
namespace Pollard_Rho {
typedef long long ll;
inline ll gcd(ll a, ll b) {ll c; while (b) c=a%b, a=b, b=c; return a;}
inline ll mulmod(ll x, ll y, const ll z) {return (x*y-(ll)(((long
  ↪ double)x*y+0.5)/(long double)z)*z+z)%z;}
inline ll powmod(ll a, ll b, const ll mo) {
  ll s = 1;
  for (; b; b>>=1, a = mulmod(a, a, mo)) if(b&1) s = mulmod(s, a, mo);
  return s;
}
bool isPrime(ll p) { // Miller-Rabin
  const int lena = 10, a[lena] = {2,3,5,7,11,13,17,19,23,29};
  if (p == 2) return true;
  if (p == 1 || !(p&1)) return false;
  ll D = p - 1;while (!(D&1)) D >>= 1;
  for (int i = 0; i < lena && a[i] < p; i++) {
    ll d = D, t = powmod(a[i], d, p); if (t == 1) continue;
    for (; d != p - 1 && t != p - 1; d <<= 1) t = mulmod(t, t, p);
    if (d == p - 1) return false;
  }
  return true;
}
void reportFactor(ll n){ // 得到一个素因子
  ans=min(ans,n);
}
ll ran(){return rand();} // 随机数
void getFactor(ll n) { // Pollard-Rho
  if (n == 1) return;
  if (isPrime(n)) { reportFactor(n); return; }
  while (true) {
    ll c = ran() % n, i = 1, x = ran() % n, y = x, k = 2;
    do {
      ll d = gcd(n + y - x, n);
      if(d != 1 && d != n) { getFactor(d); getFactor(n / d); return; }
      if (++i == k) y = x, k <<= 1;
      x = (mulmod(x, x, n) + c) % n;
    } while (y != x);
  }
}
}
```

Simplex

```
namespace Simplex{
  // where,w,way 至少要开两倍 默认有变量 >=0 的限制
  double A[30][30];
  const double eps=1e-10;
  int n,m,where[70],M,flag,ifun;
  double ans,w[70],way[70];
  void init(int _n){
    memset(A,0x00,sizeof A); memset(where,0x00,sizeof where);
    memset(w,0x00,sizeof w); memset(way,0x00,sizeof way);
    n=m=M=flag=ifun=ans=0; n=_n;
  }
  void turn(int e,int l){
    swap(where[e],where[l+n]);
    for (int i=0;i<=M;i++)
      if (i!=l){
        double t=A[i][e]/A[l][e];
        for (int j=0;j<=n;j++)
```

```
        if (j!=e) A[i][j]-=t*A[l][j]; else A[i][e]=-t;
      }
    double pre=A[l][e]; A[l][e]=1;
    for (int i=0;i<=n;i++) A[l][i]/=pre;
  }
  double solve(){
    while (1){
      int e=0,l=0;
      for (int i=1;i<=n;i++) if (A[0][i]>eps) {
        if (e==0||where[i]<where[e]) e=i;
      }
      if (e==0){return -A[0][0];}
      for (int i=1;i<=m;i++)
        if (A[i][e]>eps){
          if (l==0||A[i][0]*A[l][e]<A[i][e]*A[l][0]-eps||(A[i][0]*A[l][e]<A[i][e]*A[l][0]+eps&&where[i+n]<where[l+n])) l=i;
        }
      if (l==0){ifun=1; return 0;}
      turn(e,l);
    }
  }
  int getans(){ // 0 表示无解 ,1 表示无穷大 ,2 表示存在最大值
    n++; int l=1;
    for (int i=1;i<=m;i++) A[i][n]=-1; A[0][n]=-1;
    for (int i=1;i<=n+M;i++) where[i]=i;
    for (int i=2;i<=m;i++) if (A[i][0]<A[l][0]) swap(l,i);
    if (A[l][0]<0) turn(n,l);
    if (solve()<-eps) return 0;
    m++; for (int i=0;i<=n;i++) swap(A[0][i],A[m][i]),A[m][i]=-A[m][i];
    ans=solve(); if (ifun) return 1;
    for (int i=1;i<=n-1;i++) w[i]=0;
    for (int i=1;i<=m;i++) w[i+n]=A[i][0];
    for (int i=1;i<=n-1;i++)
      for (int j=1;j<=n+m;j++) if (where[j]==i) way[i]=w[j];
    return 2;
  }
  void setcondition(double *x,double lim){ // x 为系数 ,lim 为小于等于多少
    m++; for (int i=1;i<=n;i++) A[m][i]=x[i]; A[m][0]=lim;
  }
  void setmaximal(double *x){ // x 为系数 , 要最大化多少 , 要在限制加完后在加
    for (int i=1;i<=n;i++) A[m+1][i]=x[i]; M=m+1;
  }
};
```

Int Simplex

```
namespace simplex{ // 默认有变量 >=0 的限制
typedef int db;
const int N=1000+5,M=10000+5,inf=1e9;
db a[M][N],b[M];
int idn[N],idm[M],nxt[N],n,m;
void init(int _n){ // nxt 数组不需要初始化
  n=_n;
  memset(a,0,sizeof(a)); memset(b,0,sizeof(b));
  memset(idn,0,sizeof(idn)); memset(idm,0,sizeof(idm));
}
void pivot(int x,int y){
  swap(idm[x],idn[y]);
    db k=a[x][y];b[x]/=k;a[x][y]=1/k;
    rep(j,1,n)a[x][j]/=k;  int t=n+1;
  for(int i=1;i<=n;i++)  if(a[x][i]){nxt[t]=i;t=i;nxt[t]=-1;}
  rep(i,0,m)if(i!=x){
    db k=a[i][y];  if(!k)continue;
    b[i]-=k*b[x],a[i][y]=0;
    for(int j=nxt[n+1];j!=-1;j=nxt[j])a[i][j]-=a[x][j]*k;
  }
}
void simplex(){
  idn[0]=inf;
  while(1){
    int y=0;
    rep(j,1,n)if(a[0][j]>0&&idn[j]<idn[y])y=j;
    if(!y)break;int x=0;
    rep(i,1,m)if(a[i][y]>0)
      if(!x) x=i;else{
        int t=b[i]/a[i][y]-b[x]/a[x][y];
        if(t<0||(t==0&&idm[i]<idm[x]))x=i;
      }
    if(!x){puts("Unbounded");  exit(0);}
    pivot(x,y);
  }
}
void init_solution(){
  rep(j,1,n)idn[j]=j;  rep(i,1,m)idm[i]=n+i;
  idm[0]=inf;idn[0]=inf;
  // 寻找初始解 , 如果全为 0 是一个合法的解那么以下过程不需要进行
  while(1){
    int x=0;rep(i,1,m)if(b[i]<0&&idm[i]<idm[x])x=i;
    if(!x)break;  int y=0;
    rep(j,1,n)if(a[x][j]<0&&idn[j]<idn[y])y=j;
```

```
    if(!y){puts("Infeasible");  exit(0);}  pivot(x,y);
  }
}
void output(){ // 输出方案
  rep(j,1,n){
    bool f=1;
    rep(i,1,m)if(idm[i]==j){printf("%d ",b[i]);f=1;break;}
    if(!f)printf("0 ");
  }
  puts("");
}
void setcondition(db *x,db lim){ // x 为系数 , lim 为小于等于多少
  m++; for (int i=1;i<=n;i++) a[m][i]=x[i]; b[m]=lim;
}
void setmaximal(db *x){  // x 为系数 , 要最大化多少 , 可以在限制加完前加
  for (int i=1;i<=n;i++) a[0][i]=x[i];
}
db solve(){
  init_solution();  simplex(); return -b[0];
}
}
```

Geometry2D

```
#define mp make_pair
#define fi first
#define se second
#define pb push_back
typedef double db;
const db eps=1e-6;
const db pi=acos(-1);
int sign(db k){
    if (k>eps) return 1; else if (k<-eps) return -1; return 0;
}
int cmp(db k1,db k2){return sign(k1-k2);}
int inmid(db k1,db k2,db k3){return sign(k1-k3)*sign(k2-k3)<=0;}// k3 在 [k1,k2] 内
struct point{
    db x,y;
    point operator + (const point &k1) const{return (point){k1.x+x,k1.y+y};}
    point operator - (const point &k1) const{return (point){x-k1.x,y-k1.y};}
    point operator * (db k1) const{return (point){x*k1,y*k1};}
    point operator / (db k1) const{return (point){x/k1,y/k1};}
    int operator == (const point &k1) const{return cmp(x,k1.x)==0&&cmp(y,k1.y)==0;}
    // 逆时针旋转
    point turn(db k1){return (point){x*cos(k1)-y*sin(k1),x*sin(k1)+y*cos(k1)};}
    point turn90(){return (point){-y,x};}
    bool operator < (const point k1) const{
        int a=cmp(x,k1.x);
        if (a==-1) return 1; else if (a==1) return 0; else return cmp(y,k1.y)==-1;
    }
    db abs(){return sqrt(x*x+y*y);}
    db abs2(){return x*x+y*y;}
    db dis(point k1){return ((*this)-k1).abs();}
    point unit(){db w=abs(); return (point){x/w,y/w};}
    void scan(){double k1,k2; scanf("%lf%lf",&k1,&k2); x=k1; y=k2;}
    void print(){printf("%.11lf %.11lf\n",x,y);}
    db getw(){return atan2(y,x);}
    point getdel(){if (sign(x)==-1||(sign(x)==0&&sign(y)==-1)) return (*this)*(-1); else return (*this);}
  int getP() const{return sign(y)==1||(sign(y)==0&&sign(x)==-1);}
};
int inmid(point k1,point k2,point k3){return inmid(k1.x,k2.x,k3.x)&&inmid(k1.y,k2.y,k3.y);}
db cross(point k1,point k2){return k1.x*k2.y-k1.y*k2.x;}
db dot(point k1,point k2){return k1.x*k2.x+k1.y*k2.y;}
db rad(point k1,point k2){return atan2(cross(k1,k2),dot(k1,k2));}
// -pi -> pi
int compareangle (point k1,point k2){
    return k1.getP()<k2.getP()||(k1.getP()==k2.getP()&&sign(cross(k1,k2))>0);
}
point proj(point k1,point k2,point q){ // q 到直线 k1,k2 的投影
    point k=k2-k1; return k1+k*(dot(q-k1,k)/k.abs2());
}
point reflect(point k1,point k2,point q){return proj(k1,k2,q)*2-q;}
int clockwise(point k1,point k2,point k3){// k1 k2 k3 逆时针 1 顺时针 -1 否则 0
    return sign(cross(k2-k1,k3-k1));
}
int checkLL(point k1,point k2,point k3,point k4){// 求直线 (L) 线段 (S)k1,k2 和 k3,k4 的交点
    return cmp(cross(k3-k1,k4-k1),cross(k3-k2,k4-k2))!=0;
}
point getLL(point k1,point k2,point k3,point k4){
    db w1=cross(k1-k3,k4-k3),w2=cross(k4-k3,k2-k3); return (k1*w2+k2*w1)/(w1+w2);
}
int intersect(db l1,db r1,db l2,db r2){
    if (l1>r1) swap(l1,r1); if (l2>r2) swap(l2,r2); return cmp(r1,l2)!=-1&&cmp(r2,l1)!=-1;
}
int checkSS(point k1,point k2,point k3,point k4){
    return intersect(k1.x,k2.x,k3.x,k4.x)&&intersect(k1.y,k2.y,k3.y,k4.y)&&
    sign(cross(k3-k1,k4-k1))*sign(cross(k3-k2,k4-k2))<=0&&
```

```cpp
    sign(cross(k1-k3,k2-k3))*sign(cross(k1-k4,k2-k4))<=0;
}
db disSP(point k1,point k2,point q){
    point k3=proj(k1,k2,q);
    if (inmid(k1,k2,k3)) return q.dis(k3); else return min(q.dis(k1),q.dis(k2));
}
db disSS(point k1,point k2,point k3,point k4){
    if (checkSS(k1,k2,k3,k4)) return 0;
    else return min(min(disSP(k1,k2,k3),disSP(k1,k2,k4)),min(disSP(k3,k4,k1),disSP(k3,k4,k2)));
}
int onS(point k1,point k2,point q){return inmid(k1,k2,q)&&sign(cross(k1-q,k2-k1))==0;}
struct circle{
    point o; db r;
    void scan(){o.scan(); scanf("%lf",&r);}
    int inside(point k){return cmp(r,o.dis(k));}
};
struct line{
    // p[0]->p[1]
    point p[2];
    line(point k1,point k2){p[0]=k1; p[1]=k2;}
    point& operator [] (int k){return p[k];}
    int include(point k){return sign(cross(p[1]-p[0],k-p[0]))>0;}
    point dir(){return p[1]-p[0];}
    line push(){ // 向外 ( 左手边 ) 平移 eps
        const db eps = 1e-6;
        point delta=(p[1]-p[0]).turn90().unit()*eps;
        return {p[0]-delta,p[1]-delta};
    }
};
point getLL(line k1,line k2){return getLL(k1[0],k1[1],k2[0],k2[1]);}
int parallel(line k1,line k2){return sign(cross(k1.dir(),k2.dir()))==0;}
int sameDir(line k1,line k2){return parallel(k1,k2)&&sign(dot(k1.dir(),k2.dir()))==1;}
int operator < (line k1,line k2){
    if (sameDir(k1,k2)) return k2.include(k1[0]);
    return compareangle(k1.dir(),k2.dir());
}
int checkpos(line k1,line k2,line k3){return k3.include(getLL(k1,k2));}
vector<line> getHL(vector<line> &L){ // 求半平面交 , 半平面是逆时针方向 , 输出按照逆时针
    sort(L.begin(),L.end()); deque<line> q;
    for (int i=0;i<(int)L.size();i++){
        if (i&&sameDir(L[i],L[i-1])) continue;
        while (q.size()>1&&!checkpos(q[q.size()-2],q[q.size()-1],L[i])) q.pop_back();
        while (q.size()>1&&!checkpos(q[1],q[0],L[i])) q.pop_front();
        q.push_back(L[i]);
    }
    while (q.size()>2&&!checkpos(q[q.size()-2],q[q.size()-1],q[0])) q.pop_back();
    while (q.size()>2&&!checkpos(q[1],q[0],q[q.size()-1])) q.pop_front();
    vector<line>ans; for (int i=0;i<q.size();i++) ans.push_back(q[i]);
    return ans;
}
db closepoint(vector<point>&A,int l,int r){ // 最近点对 , 先要按照 x 坐标排序
    if (r-l<=5){
        db ans=1e20;
        for (int i=l;i<=r;i++) for (int j=i+1;j<=r;j++) ans=min(ans,A[i].dis(A[j]));
        return ans;
    }
    int mid=l+r>>1; db ans=min(closepoint(A,l,mid),closepoint(A,mid+1,r));
    vector<point>B; for (int i=l;i<=r;i++) if (abs(A[i].x-A[mid].x)<=ans) B.push_back(A[i]);
    sort(B.begin(),B.end(),[](point k1,point k2){return k1.y<k2.y;});
    for (int i=0;i<B.size();i++) for (int j=i+1;j<B.size()&&B[j].y-B[i].y<ans;j++) ans=min(ans,B[i].dis(B[j]));
    return ans;
}
int checkposCC(circle k1,circle k2){// 返回两个圆的公切线数量
    if (cmp(k1.r,k2.r)==-1) swap(k1,k2);
    db dis=k1.o.dis(k2.o);  int w1=cmp(dis,k1.r+k2.r),w2=cmp(dis,k1.r-k2.r);
    if (w1>0) return 4; else if (w1==0) return 3; else if (w2>0) return 2;
    else if (w2==0) return 1; else return 0;
}
vector<point> getCL(circle k1,point k2,point k3){ // 沿着 k2->k3 方向给出 , 相切给出两个
    point k=proj(k2,k3,k1.o); db d=k1.r*k1.r-(k-k1.o).abs2();
    if (sign(d)==-1) return {};
    point del=(k3-k2).unit()*sqrt(max((db)0.0,d)); return {k-del,k+del};
}
vector<point> getCC(circle k1,circle k2){// 沿圆 k1 逆时针给出 , 相切给出两个
    int pd=checkposCC(k1,k2); if (pd==0||pd==4) return {};
    db a=(k2.o-k1.o).abs2(),cosA=(k1.r*k1.r+a-k2.r*k2.r)/(2*k1.r*sqrt(max(a,(db)0.0)));
    db b=k1.r*cosA,c=sqrt(max((db)0.0,k1.r*k1.r-b*b));
    point k=(k2.o-k1.o).unit(),m=k1.o+k*b,del=k.turn90()*c;
    return {m-del,m+del};
}
vector<point> TangentCP(circle k1,point k2){// 沿圆 k1 逆时针给出
```

```cpp
    db a=(k2-k1.o).abs(),b=k1.r*k1.r/a,c=sqrt(max((db)0.0,k1.r*k1.r-b*b));
    point k=(k2-k1.o).unit(),m=k1.o+k*b,del=k.turn90()*c;
    return {m-del,m+del};
}
vector<line> TangentoutCC(circle k1,circle k2){
    int pd=checkposCC(k1,k2); if (pd==0) return {};
    if (pd==1){point k=getCC(k1,k2)[0]; return {(line){k,k}};}
    if (cmp(k1.r,k2.r)==0){
        point del=(k2.o-k1.o).unit().turn90().getdel();
        return
        ↪ {(line){k1.o-del*k1.r,k2.o-del*k2.r},(line){k1.o+del*k1.r,k2.o+del*k2.r}};
    } else {
        point p=(k2.o*k1.r-k1.o*k2.r)/(k1.r-k2.r);
        vector<point>A=TangentCP(k1,p),B=TangentCP(k2,p);
        vector<line>ans; for (int i=0;i<A.size();i++)
        ↪ ans.push_back((line){A[i],B[i]});
        return ans;
    }
}
vector<line> TangentinCC(circle k1,circle k2){
    int pd=checkposCC(k1,k2); if (pd<=2) return {};
    if (pd==3){point k=getCC(k1,k2)[0]; return {(line){k,k}};}
    point p=(k2.o*k1.r+k1.o*k2.r)/(k1.r+k2.r);
    vector<point>A=TangentCP(k1,p),B=TangentCP(k2,p);
    vector<line>ans; for (int i=0;i<A.size();i++) ans.push_back((line){A[i],B[i]});
    return ans;
}
vector<line> TangentCC(circle k1,circle k2){
    int flag=0; if (k1.r<k2.r) swap(k1,k2),flag=1;
    vector<line>A=TangentoutCC(k1,k2),B=TangentinCC(k1,k2);
    for (line k:B) A.push_back(k);
    if (flag) for (line &k:A) swap(k[0],k[1]);
    return A;
}
db getarea(circle k1,point k2,point k3){
    // 圆 k1 与三角形 k2 k3 k1.o 的有向面积交
    point k=k1.o; k1.o=k1.o-k; k2=k2-k; k3=k3-k;
    int pd1=k1.inside(k2),pd2=k1.inside(k3);
    vector<point>A=getCL(k1,k2,k3);
    if (pd1>=0){
        if (pd2>=0) return cross(k2,k3)/2;
        return k1.r*k1.r*rad(A[1],k3)/2+cross(k2,A[1])/2;
    } else if (pd2>=0){
        return k1.r*k1.r*rad(k2,A[0])/2+cross(A[0],k3)/2;
    }else {
        int pd=cmp(k1.r,disSP(k2,k3,k1.o));
        if (pd<=0) return k1.r*k1.r*rad(k2,k3)/2;
        return cross(A[0],A[1])/2+k1.r*k1.r*(rad(k2,A[0])+rad(A[1],k3))/2;
    }
}
circle getcircle(point k1,point k2,point k3){
    db a1=k2.x-k1.x,b1=k2.y-k1.y,c1=(a1*a1+b1*b1)/2;
    db a2=k3.x-k1.x,b2=k3.y-k1.y,c2=(a2*a2+b2*b2)/2;
    db d=a1*b2-a2*b1;
    point o=(point){k1.x+(c1*b2-c2*b1)/d,k1.y+(a1*c2-a2*c1)/d};
    return (circle){o,k1.dis(o)};
}
circle getScircle(vector<point> A){
    random_shuffle(A.begin(),A.end());
    circle ans=(circle){A[0],0};
    for (int i=1;i<A.size();i++)
        if (ans.inside(A[i])==-1){
            ans=(circle){A[i],0};
            for (int j=0;j<i;j++)
                if (ans.inside(A[j])==-1){
                    ans.o=(A[i]+A[j])/2; ans.r=ans.o.dis(A[i]);
                    for (int k=0;k<j;k++)
                        if (ans.inside(A[k])==-1)
                            ans=getcircle(A[i],A[j],A[k]);
                }
        }
    return ans;
}
db area(vector<point> A){ // 多边形用 vector<point> 表示 , 逆时针
    db ans=0;
    for (int i=0;i<A.size();i++) ans+=cross(A[i],A[(i+1)%A.size()]);
    return ans/2;
}
int checkconvex(vector<point>A){
    int n=A.size(); A.push_back(A[0]); A.push_back(A[1]);
    for (int i=0;i<n;i++) if (sign(cross(A[i+1]-A[i],A[i+2]-A[i]))==-1) return 0;
    return 1;
}
int contain(vector<point>A,point q){ // 2 内部 1 边界 0 外部
    int pd=0; A.push_back(A[0]);
    for (int i=1;i<A.size();i++){
        point u=A[i-1],v=A[i];
        if (onS(u,v,q)) return 1; if (cmp(u.y,v.y)>0) swap(u,v);
        if (cmp(u.y,q.y)>=0||cmp(v.y,q.y)<0) continue;
        if (sign(cross(u-v,q-v))<0) pd^=1;
```

```cpp
    }
    return pd<<1;
}
vector<point> ConvexHull(vector<point>A,int flag=1){ // flag=0 不严格 flag=1 严格
    int n=A.size(); vector<point>ans(n*2);
    sort(A.begin(),A.end()); int now=-1;
    for (int i=0;i<A.size();i++){
        while (now>0&&sign(cross(ans[now]-ans[now-1],A[i]-ans[now-1]))<flag) now--;
        ans[++now]=A[i];
    } int pre=now;
    for (int i=n-2;i>=0;i--){
        while (now>pre&&sign(cross(ans[now]-ans[now-1],A[i]-ans[now-1]))<flag) now--;
        ans[++now]=A[i];
    } ans.resize(now); return ans;
}
db convexDiameter(vector<point>A){
    int now=0,n=A.size(); db ans=0;
    for (int i=0;i<A.size();i++){
        now=max(now,i);
        while (1){
            db k1=A[i].dis(A[now%n]),k2=A[i].dis(A[(now+1)%n]);
            ans=max(ans,max(k1,k2)); if (k2>k1) now++; else break;
        }
    }
    return ans;
}
vector<point> convexcut(vector<point>A,point k1,point k2){
    // 保留 k1,k2,p 逆时针的所有点
    int n=A.size(); A.push_back(A[0]); vector<point>ans;
    for (int i=0;i<n;i++){
        int w1=clockwise(k1,k2,A[i]),w2=clockwise(k1,k2,A[i+1]);
        if (w1>=0) ans.push_back(A[i]);
        if (w1*w2<0) ans.push_back(getLL(k1,k2,A[i],A[i+1]));
    }
    return ans;
}
int checkPoS(vector<point>A,point k1,point k2){
    // 多边形 A 和直线 ( 线段 )k1->k2 严格相交 , 注释部分为线段
    struct ins{
        point m,u,v;
        int operator < (const ins& k) const {return m<k.m;}
    }; vector<ins>B;
    //if (contain(A,k1)==2||contain(A,k2)==2) return 1;
    vector<point>poly=A; A.push_back(A[0]);
    for (int i=1;i<A.size();i++) if (checkLL(A[i-1],A[i],k1,k2)){
        point m=getLL(A[i-1],A[i],k1,k2);
        if (inmid(A[i-1],A[i],m)/*&&inmid(k1,k2,m)*/) B.push_back((ins){m,A[i-1],A[i]});
    }
    if (B.size()==0) return 0; sort(B.begin(),B.end());
    int now=1; while (now<B.size()&&B[now].m==B[0].m) now++;
    if (now==B.size()) return 0;
    int flag=contain(poly,(B[0].m+B[now].m)/2);
    if (flag==2) return 1;
    point d=B[now].m-B[0].m;
    for (int i=now;i<B.size();i++){
        if (!(B[i].m==B[i-1].m)&&flag==2) return 1;
        int tag=sign(cross(B[i].v-B[i].u,B[i].m+d-B[i].u));
        if (B[i].m==B[i].u||B[i].m==B[i].v) flag+=tag; else flag+=tag*2;
    }
    //return 0;
    return flag==2;
}
int checkinp(point r,point l,point m){
  if (compareangle(l,r)){return compareangle(l,m)&&compareangle(m,r);}
  return compareangle(l,m)||compareangle(m,r);
}
int checkPosFast(vector<point>A,point k1,point k2){ // 快速检查线段是否和多边形严格相交
  if (contain(A,k1)==2||contain(A,k2)==2) return 1; if (k1==k2) return 0;
  A.push_back(A[0]); A.push_back(A[1]);
  for (int i=1;i+1<A.size();i++)
    if (checkLL(A[i-1],A[i],k1,k2)){
      point now=getLL(A[i-1],A[i],k1,k2);
      if (inmid(A[i-1],A[i],now)==0||inmid(k1,k2,now)==0) continue;
      if (now==A[i]){
        if (A[i]==k2) continue;
        point pre=A[i-1],ne=A[i+1];
        if (checkinp(pre-now,ne-now,k2-now)) return 1;
      } else if (now==k1){
        if (k1==A[i-1]||k1==A[i]) continue;
        if (checkinp(A[i-1]-k1,A[i]-k1,k2-k1)) return 1;
      } else if (now==k2||now==A[i-1]) continue;
      else return 1;
    }
  return 0;
}
// 拆分凸包成上下凸壳 凸包尽量都随机旋转一个角度来避免出现相同横坐标
// 尽量特判只有一个点的情况 凸包逆时针
void getUDP(vector<point>A,vector<point>&U,vector<point>&D){
```

```
    db l=1e100,r=-1e100;
    for (int i=0;i<A.size();i++) l=min(l,A[i].x),r=max(r,A[i].x);
    int wherel,wherer;
    for (int i=0;i<A.size();i++) if (cmp(A[i].x,l)==0) wherel=i;
    for (int i=A.size();i;i--) if (cmp(A[i-1].x,r)==0) wherer=i-1;
    U.clear(); D.clear(); int now=wherel;
    while (1){D.push_back(A[now]); if (now==wherer) break; now++; if (now>=A.size()) now=0;}
    now=wherel;
    while (1){U.push_back(A[now]); if (now==wherer) break; now--; if (now<0) now=A.size()-1;}
}
// 需要保证凸包点数大于等于 3,2 内部 ,1 边界 ,0 外部
int containCoP(const vector<point>&U,const vector<point>&D,point k){
    db lx=U[0].x,rx=U[U.size()-1].x;
    if (k==U[0]||k==U[U.size()-1]) return 1;
    if (cmp(k.x,lx)==-1||cmp(k.x,rx)==1) return 0;
    int where1=lower_bound(U.begin(),U.end(),(point){k.x,-1e100})-U.begin();
    int where2=lower_bound(D.begin(),D.end(),(point){k.x,-1e100})-D.begin();
    int w1=clockwise(U[where1-1],U[where1],k),w2=clockwise(D[where2-1],D[where2],k);
    if (w1==1||w2==-1) return 0; else if (w1==0||w2==0) return 1; return 2;
}
// d 是方向 , 输出上方切点和下方切点
pair<point,point> getTangentCow(const vector<point> &U,const vector<point> &D,point d){
    if (sign(d.x)<0||(sign(d.x)==0&&sign(d.y)<0)) d=d*(-1);
    point whereU,whereD;
    if (sign(d.x)==0) return mp(U[0],U[U.size()-1]);
    int l=0,r=U.size()-1,ans=0;
    while (l<r){int mid=l+r>>1; if (sign(cross(U[mid+1]-U[mid],d))<=0) l=mid+1,ans=mid+1; else r=mid;}
    whereU=U[ans]; l=0,r=D.size()-1,ans=0;
    while (l<r){int mid=l+r>>1; if (sign(cross(D[mid+1]-D[mid],d))>=0) l=mid+1,ans=mid+1; else r=mid;}
    whereD=D[ans]; return mp(whereU,whereD);
}
// 先检查 contain, 逆时针给出
pair<point,point> getTangentCoP(const vector<point>&U,const vector<point>&D,point k){
    db lx=U[0].x,rx=U[U.size()-1].x;
    if (k.x<lx){
        int l=0,r=U.size()-1,ans=U.size()-1;
        while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid+1])==1) l=mid+1; else ans=mid,r=mid;}
        point w1=U[ans]; l=0,r=D.size()-1,ans=D.size()-1;
        while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid+1])==-1) l=mid+1; else ans=mid,r=mid;}
        point w2=D[ans]; return mp(w1,w2);
    } else if (k.x>rx){
        int l=1,r=U.size(),ans=0;
        while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid-1])==-1) r=mid; else ans=mid,l=mid+1;}
        point w1=U[ans]; l=1,r=D.size(),ans=0;
        while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid-1])==1) r=mid; else ans=mid,l=mid+1;}
        point w2=D[ans]; return mp(w2,w1);
    } else {
        int where1=lower_bound(U.begin(),U.end(),(point){k.x,-1e100})-U.begin();
        int where2=lower_bound(D.begin(),D.end(),(point){k.x,-1e100})-D.begin();
        if ((k.x==lx&&k.y>U[0].y)||(where1&&clockwise(U[where1-1],U[where1],k)==1)){
            int l=1,r=where1+1,ans=0;
            while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid-1])==1) ans=mid,l=mid+1; else r=mid;}
            point w1=U[ans]; l=where1,r=U.size()-1,ans=U.size()-1;
            while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid+1])==1) l=mid+1; else ans=mid,r=mid;}
            point w2=U[ans]; return mp(w2,w1);
        } else {
            int l=1,r=where2+1,ans=0;
            while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid-1])==-1) ans=mid,l=mid+1; else r=mid;}
            point w1=D[ans]; l=where2,r=D.size()-1,ans=D.size()-1;
            while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid+1])==-1) l=mid+1; else ans=mid,r=mid;}
            point w2=D[ans]; return mp(w1,w2);
        }
    }
}
struct P3{
    db x,y,z;
    P3 operator + (P3 k1){return (P3){x+k1.x,y+k1.y,z+k1.z};}
    P3 operator - (P3 k1){return (P3){x-k1.x,y-k1.y,z-k1.z};}
    P3 operator * (db k1){return (P3){x*k1,y*k1,z*k1};}
    P3 operator / (db k1){return (P3){x/k1,y/k1,z/k1};}
    db abs2(){return x*x+y*y+z*z;}
    db abs(){return sqrt(x*x+y*y+z*z);}
    P3 unit(){return (*this)/abs();}
    int operator < (const P3 k1) const{
        if (cmp(x,k1.x)!=0) return x<k1.x;
        if (cmp(y,k1.y)!=0) return y<k1.y;
```

```cpp
        return cmp(z,k1.z)==-1;
    }
    int operator == (const P3 k1){
        return cmp(x,k1.x)==0&&cmp(y,k1.y)==0&&cmp(z,k1.z)==0;
    }
    void scan(){
        double k1,k2,k3; scanf("%lf%lf%lf",&k1,&k2,&k3);
        x=k1; y=k2; z=k3;
    }
};
P3 cross(P3 k1,P3 k2){return (P3){k1.y*k2.z-k1.z*k2.y,k1.z*k2.x-k1.x*k2.z,k1.x*k2.y-k1.y*k2.x};}
db dot(P3 k1,P3 k2){return k1.x*k2.x+k1.y*k2.y+k1.z*k2.z;}
//p=(3,4,5),l=(13,19,21),theta=85 ans=(2.83,4.62,1.77)
P3 turn3D(db k1,P3 l,P3 p){
    l=l.unit(); P3 ans; db c=cos(k1),s=sin(k1);
    ans.x=p.x*(l.x*l.x*(1-c)+c)+p.y*(l.x*l.y*(1-c)-l.z*s)+p.z*(l.x*l.z*(1-c)+l.y*s);
    ans.y=p.x*(l.x*l.y*(1-c)+l.z*s)+p.y*(l.y*l.y*(1-c)+c)+p.z*(l.y*l.z*(1-c)-l.x*s);
    ans.z=p.x*(l.x*l.z*(1-c)-l.y*s)+p.y*(l.y*l.z*(1-c)+l.x*s)+p.z*(l.x*l.x*(1-c)+c);
    return ans;
}
typedef vector<P3> VP;
typedef vector<VP> VVP;
db Acos(db x){return acos(max(-(db)1,min(x,(db)1)));}
// 球面距离 , 圆心原点 , 半径 1
db Odist(P3 a,P3 b){db r=Acos(dot(a,b)); return r;}
db r; P3 rnd;
vector<db> solve(db a,db b,db c){
    db r=sqrt(a*a+b*b),th=atan2(b,a);
    if (cmp(c,-r)==-1) return {0};
    else if (cmp(r,c)<=0) return {1};
    else {
        db tr=pi-Acos(c/r); return {th+pi-tr,th+pi+tr};
    }
}
vector<db> jiao(P3 a,P3 b){
    // dot(rd+x*cos(t)+y*sin(t),b) >= cos(r)
    if (cmp(Odist(a,b),2*r)>0) return {0};
    P3 rd=a*cos(r),z=a.unit(),y=cross(z,rnd).unit(),x=cross(y,z).unit();
    vector<db> ret = solve(-(dot(x,b)*sin(r)),-(dot(y,b)*sin(r)),-(cos(r)-dot(rd,b)));
    return ret;
}
db norm(db x,db l=0,db r=2*pi){ // change x into [l,r)
    while (cmp(x,l)==-1) x+=(r-l); while (cmp(x,r)>=0) x-=(r-l);
    return x;
}
db disLP(P3 k1,P3 k2,P3 q){
    return (cross(k2-k1,q-k1)).abs()/(k2-k1).abs();
}
db disLL(P3 k1,P3 k2,P3 k3,P3 k4){
    P3 dir=cross(k2-k1,k4-k3); if (sign(dir.abs())==0) return disLP(k1,k2,k3);
    return fabs(dot(dir.unit(),k1-k2));
}
VP getFL(P3 p,P3 dir,P3 k1,P3 k2){
    db a=dot(k2-p,dir),b=dot(k1-p,dir),d=a-b;
    if (sign(fabs(d))==0) return {};
    return {(k1*a-k2*b)/d};
}
VP getFF(P3 p1,P3 dir1,P3 p2,P3 dir2){// 返回一条线
    P3 e=cross(dir1,dir2),v=cross(dir1,e);
    db d=dot(dir2,v); if (sign(abs(d))==0) return {};
    P3 q=p1+v*dot(dir2,p2-p1)/d; return {q,q+e};
}
// 3D Covex Hull Template
db getV(P3 k1,P3 k2,P3 k3,P3 k4){ // get the Volume
    return dot(cross(k2-k1,k3-k1),k4-k1);
}
db rand_db(){return 1.0*rand()/RAND_MAX;}
VP convexHull2D(VP A,P3 dir){
    P3 x={(db)rand(),(db)rand(),(db)rand()}; x=x.unit();
    x=cross(x,dir).unit(); P3 y=cross(x,dir).unit();
    P3 vec=dir.unit()*dot(A[0],dir);
    vector<point>B;
    for (int i=0;i<A.size();i++) B.push_back((point){dot(A[i],x),dot(A[i],y)});
    B=ConvexHull(B); A.clear();
    for (int i=0;i<B.size();i++) A.push_back(x*B[i].x+y*B[i].y+vec);
    return A;
}
namespace CH3{
    VVP ret; set<pair<int,int> >e;
    int n; VP p,q;
    void wrap(int a,int b){
        if (e.find({a,b})==e.end()){
            int c=-1;
            for (int i=0;i<n;i++) if (i!=a&&i!=b){
                if (c==-1||sign(getV(q[c],q[a],q[b],q[i]))>0) c=i;
            }
            if (c!=-1){
                ret.push_back({p[a],p[b],p[c]});
```

```
                e.insert({a,b}); e.insert({b,c}); e.insert({c,a});
                wrap(c,b); wrap(a,c);
            }
        }
    }
    VVP ConvexHull3D(VP _p){
        p=q=_p; n=p.size();
        ret.clear(); e.clear();
        for (auto &i:q) i=i+(P3){rand_db()*1e-4,rand_db()*1e-4,rand_db()*1e-4};
        for (int i=1;i<n;i++) if (q[i].x<q[0].x) swap(p[0],p[i]),swap(q[0],q[i]);
        for (int i=2;i<n;i++) if
  ↪ ((q[i].x-q[0].x)*(q[1].y-q[0].y)>(q[i].y-q[0].y)*(q[1].x-q[0].x))
  ↪ swap(q[1],q[i]),swap(p[1],p[i]);
        wrap(0,1);
        return ret;
    }
}
VVP reduceCH(VVP A){
    VVP ret; map<P3,VP> M;
    for (VP nowF:A){
        P3 dir=cross(nowF[1]-nowF[0],nowF[2]-nowF[0]).unit();
        for (P3 k1:nowF) M[dir].pb(k1);
    }
    for (pair<P3,VP> nowF:M) ret.pb(convexHull2D(nowF.se,nowF.fi));
    return ret;
}
//  把一个面变成 ( 点 , 法向量 ) 的形式
pair<P3,P3> getF(VP F){
    return mp(F[0],cross(F[1]-F[0],F[2]-F[0]).unit());
}
// 3D Cut 保留 dot(dir,x-p)>=0 的部分
VVP ConvexCut3D(VVP A,P3 p,P3 dir){
    VVP ret; VP sec;
    for (VP nowF: A){
        int n=nowF.size(); VP ans; int dif=0;
        for (int i=0;i<n;i++){
            int d1=sign(dot(dir,nowF[i]-p));
            int d2=sign(dot(dir,nowF[(i+1)%n]-p));
            if (d1>=0) ans.pb(nowF[i]);
            if (d1*d2<0){
                P3 q=getFL(p,dir,nowF[i],nowF[(i+1)%n])[0];
                ans.push_back(q); sec.push_back(q);
            }
            if (d1==0) sec.push_back(nowF[i]); else dif=1;

  ↪ dif|=(sign(dot(dir,cross(nowF[(i+1)%n]-nowF[i],nowF[(i+1)%n]-nowF[i])))==-1);
        }
        if (ans.size()>0&&dif) ret.push_back(ans);
    }
    if (sec.size()>0) ret.push_back(convexHull2D(sec,dir));
    return ret;
}
db vol(VVP A){
    if (A.size()==0) return 0; P3 p=A[0][0]; db ans=0;
    for (VP nowF:A)
        for (int i=2;i<nowF.size();i++)
            ans+=abs(getV(p,nowF[0],nowF[i-1],nowF[i]));
    return ans/6;
}
VVP init(db INF) {
    VVP pss(6,VP(4));
    pss[0][0] = pss[1][0] = pss[2][0] = {-INF, -INF, -INF};
    pss[0][3] = pss[1][1] = pss[5][2] = {-INF, -INF, INF};
    pss[0][1] = pss[2][3] = pss[4][2] = {-INF, INF, -INF};
    pss[0][2] = pss[5][3] = pss[4][1] = {-INF, INF, INF};
    pss[1][3] = pss[2][1] = pss[3][2] = {INF, -INF, -INF};
    pss[1][2] = pss[5][1] = pss[3][3] = {INF, -INF, INF};
    pss[2][2] = pss[4][3] = pss[3][1] = {INF, INF, -INF};
    pss[5][0] = pss[4][0] = pss[3][0] = {INF, INF, INF};
    return pss;
}
```

弦图相关

1. 团数 $\leq$ 色数，弦图团数 $=$ 色数

2. 设 $next(v)$ 表示 $N(v)$ 中最前的点．令 w* 表示所有满足 $A \in B$ 的 w 中最后的一个点，判断 $v \cup N(v)$ 是否为极大团，只需判断是否存在一个 w, 满足 $Next(w) = v$ 且 $|N(v)| + 1 \leq |N(w)|$ 即可．

3. 最小染色：完美消除序列从后往前依次给每个点染色，给每个点染上可以染的最小的颜色

4. 最大独立集：完美消除序列从前往后能选就选

5. 弦图最大独立集数 $=$ 最小团覆盖数，最小团覆盖：设最大独立集为 $\{p_1, p_2, \ldots, p_t\}$，则 $\{p_1 \cup N(p_1), \ldots, p_t \cup N(p_t)\}$ 为最小团覆盖

综合

二分图　定理 1: 最小覆盖数 $=$ 最大匹配数

定理 2: 最大独立集 S 与 最小覆盖集 T 互补

算法:

1. 做最大匹配，没有匹配的空闲点 $\in S$
2. 如果 $u \in S$ 那么 u 的临点必然属于 T
3. 如果一对匹配的点中有一个属于 T 那么另外一个属于 S
4. 还不能确定的，把左子图的放入 S，右子图放入 T

算法结束

上下界流　上下界无源汇可行流 ： 不用添 $T->S$， 判断是否流量平衡
上下界有源汇可行流 ： 添 $T\to S$（ 下界 0，上界 ∞）， 判断是否流量平衡
上下界最小流 ： 不添 $T\to S$ 先流一遍 ， 再添 $T\to S$（ 下界 0，上界 ∞）在残图上流一遍 ， 答案为 $S\to T$ 的流量值
上下界最大流 ： 添 $T\to S$（ 下界 0，上界 ∞）流一遍，再在残图上流一遍S到T的最大流，答案为前者的 $S\to T$ 的值 ＋ 残图中 $S\to T$ 的最大流 （ 不删那条边的话，最后的最大流就是答案 ）

最大流对偶　考虑最大费用循环流的标准线性规划建模：

Maximize: $\sum_{i\in E} cost_i\cdot f_i$

- 对每条弧i有 $0\le f_i\le cap_i$ ，cap_i 表示这条弧的容量，$f_i\ge 0$。
- 对于每个点x有流量平衡： $\sum_{u_i=x} f_i-\sum_{v_i=x} f_i=0$

共有$|V|+|E|$个限制，对偶后，设前$|V|$个限制对应的变量为a_i，后$|E|$个限制对应的变量为d_i：

Minimize: $\sum_{i\in E} cap_i\cdot d_i$

- 对每条弧i有 $a_{v_i}-a_{u_i}+d_i\ge cost_i$。
- a_x无限制，$d_i\ge 0$。
  - $min\ge -> max\le$

所以，比如有很多变量然后给定一些差分后的不等式然后可以花费代价让一个不等式“放宽”，目标总代价最小的模型，都是最大费用流的对偶。

类欧几里得

- $f(a,b,c,n)=\sum_{i=0}^{n}\lfloor\frac{ai+b}{c}\rfloor$
- $m=\lfloor\frac{an+b}{c}\rfloor, f(a,b,c,n)=nm-f(c,c-b-1,a,m-1)$

拟阵　1、求最小权基，贪心；
2、求两个拟阵(M_1,I_1)和(M_2,I_2)的最小权拟阵交，从空集开始每次增加一个元素，
假设当前集合为A，建图：
如果x不属于A，$A+\{x\}\in I_1$，连边S->x，边权为x的权值；
如果x不属于A，$A+\{x\}\in I_2$，连边x->T，边权为0；
如果x不属于A， y属于A，$A-\{y\}+\{x\}\in I_2$，连边x->y,边权为y的权值的相反数；
如果x不属于A， y属于A，$A-\{y\}+\{x\}\in I_1$，连边y->x，边权为x的权值；
找出S->T的最短路，把路径上每个点的是否在集合里取反。
3、把S分解为最少的拟阵的并：
最小值为$\max\lceil\frac{|S|}{r(|S|)}\rceil$
每次增加一个元素x，每个当前的等价类A_i连边$S->A_i$。
如果y不属于A_i，$A_i+\{y\}\in I$，连边$A_i->y$。
如果y不属于A_i， z属于A_i，$A_i-\{z\}+\{y\}\in I$，连边$y->z$。

染色多项式

$$\text{number of acyclic orientations of G is } (-1)^{|V(G)|}P(G,-1)$$

$$\text{Cycle } P(C_n,t)=(t-1)^n+(-1)^n(t-1)$$

Petersen graph $P(P_5,t)=t(t-1)(t-2)(t^7-12t^6+67t^5-230t^4+529t^3-814t^2+775t-$

伯努利数

$$\sum_{k=0}^{m-1}k^n=\frac{1}{n+1}\sum_{k=0}^{n}\binom{n+1}{k}B_k m^{n+1-k}$$

$$\sum_{j=0}^{m}\binom{m+1}{j}B_j=0\qquad \frac{B_{m+p-1}}{m+p-1}\equiv\frac{B_m}{m}(\mod p)$$

高维单位球

$$A(d)=\frac{2\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})},V(d)=\frac{1}{d}A(d)$$

基本形

椭圆　标准形 $\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$，离心率 $e=\frac{c}{a},c=\sqrt{a^2-b^2}$，焦点参数 $p=\frac{b^2}{a}$

椭圆上(x,y)处曲率半径 $R=a^2b^2(\frac{x^2}{a^4}+\frac{y^2}{b^4})^{\frac{3}{2}}=\frac{(r_1r_2)^{\frac{3}{2}}}{ab}$，其中$r_i$为到焦点$F_i$距离

点$A(a,0)$ ，$M(x,y)$ 则扇形面积 $S_{OAM}=\frac{1}{2}ab\arccos\frac{x}{a}$ 弧长

$$L_{AM}=a\int_0^{\arccos\frac{x}{a}}\sqrt{1-e^2cos^2t}dt=a\int_{\arccos\frac{x}{a}}^{\frac{\pi}{2}}\sqrt{1-e^2\sin^2t}dt$$

周长 $L=2a\pi[1-(\frac{1}{2})^2e^2-(\frac{1\times 3}{2\times 4})^2\frac{e^4}{3}-\ldots]$ 极坐标方程 $r^2=\frac{b^2a^2}{b^2\cos^2\theta+a^2\sin^2\theta}$

抛物线　标准形 $y^2=2px$，曲率半径 $R=((p+2x)^{3/2})/\sqrt{p}$，其中$r_i$为到焦点$F_i$距离

点$A(a,0)$ ，$M(x,y)$ 则扇形面积 $S_{OAM}=\frac{1}{2}ab\arccos\frac{x}{a}$ 弧长

$$L_{OM}=\frac{p}{2}[\sqrt{\frac{2x}{p}(1+\frac{2x}{p})}+\ln(\frac{2x}{p}+\sqrt{1+\frac{2x}{p}})]$$

重心　半径r圆心角θ的扇形重心与圆心距离 $\frac{4r}{3\theta}\sin\frac{\theta}{2}$

半径r圆心角θ的圆弧重心与圆心距离 $\frac{4r}{3\theta-3\sin\theta}\sin^3\frac{\theta}{2}$

椭圆上半部分重心与圆心距离 $\frac{4}{3\pi}b$

树的计数　若$n+1$个点的有根树总数为a_{n+1}，　无根树总数为b_{n+1}，　$a_i = \{1,1,2,4,9,20,286,1842\dots\}$

$$S_{n,j}=\sum_{i=1}^{n/j}a_{n+1-ij}=S_{n-j,j}+a_{n+1-j}\qquad a_{n+1}=\frac{1}{n}\sum_{j=1}^{n}ja_jS_{n,j}$$

$$b_{2k+1}=a_n-\sum_{i=1}^{n/2}a_ia_{n-i}\qquad b_{2k}=a_n-\sum_{i=1}^{n/2}a_ia_{n-i}+\frac{1}{2}a_{n/2}(a_{n/2}+1)$$

组合公式

$$\sum_{k=1}^{n}k^5=\frac{1}{12}n^2(n+1)^2(2N^2+2n-1)\qquad \sum_{k=1}^{n}k^4=\frac{1}{30}n(n+1)(2n+1)(3n^2+3n-1)$$

$$\text{限位排列}Ans=\sum_{i=0}^{n}(-1)^k*r_k*(n-i)!$$

其中r_k表示把k个物品放在不能放的位置上使得每行每列至多一个的方案数

三角公式

$$\sin(\alpha\pm\beta)=\sin\alpha\cos\beta\pm\cos\alpha\sin\beta\qquad\cos(\alpha\pm\beta)=\cos\alpha\cos\beta\mp\sin\alpha\sin\beta$$

$$\tan(\alpha\pm\beta)=\frac{\tan\alpha\pm\tan\beta}{1\mp\tan\alpha\tan\beta}\qquad\tan(\alpha)\pm\tan(\beta)=\frac{\sin(\alpha\pm\beta)}{\cos\alpha\cos\beta}$$

$$\sin(n\alpha)=n\cos^{n-1}\alpha\sin\alpha-\binom{n}{3}\cos^{n-3}\alpha\sin^3\alpha+\binom{n}{5}\cos^{n-5}\alpha\sin^5\alpha-\dots$$

$$\cos(n\alpha)=\cos^n\alpha\sin\alpha-\binom{n}{2}\cos^{n-2}\alpha\sin^2\alpha+\binom{n}{4}\cos^{n-4}\alpha\sin^4\alpha-\dots$$

反演

$$a_n=\sum_{k=0}^{n}C_n^kb_k,\qquad b_n=\sum_{k=0}^{n}(-1)^{k+n}C_n^ka_k$$

$$a_n=\sum_{k=n}^{\text{inf}}C_k^nb_k,\qquad b_n=\sum_{k=n}^{\text{inf}}(-1)^{k+n}C_k^na_k$$

$$a_n=\sum_{k=0}^{n}C_{n+p}^{k+p}b_k,\qquad b_n=\sum_{k=n}^{\text{inf}}(-1)^{k+n}C_{n+p}^{k+p}a_k$$

$$a_n=\sum_{k=n}^{\text{inf}}C_{k+p}^{n+p}b_k,\qquad b_n=\sum_{k=n}^{\text{inf}}(-1)^{k+n}C_{k+p}^{n+p}a_k$$

$$f(n)=\sum_{d|n}g(d),\qquad g(n)=\sum_{d|n}\mu(d)f(\frac{n}{d})$$

杜教筛　$S(n)=\sum_{i=1}^{n}f(i)$

$$g(1)S(n)=\sum_{i=1}^{n}(f*g)(i)-\sum_{i=2}^{n}g(i)S(\lfloor\frac{n}{i}\rfloor)$$

$S(n)=\sum_{i=1}^{n}(f\cdot g)(i)$，$g(x)$ 为完全积性函数。有：

$$S(n)=\sum_{i=1}^{n}[(f*1)\cdot g](i)-\sum_{i=2}^{n}S(\lfloor\frac{n}{i}\rfloor)g(i)$$

$S(n)=\sum_{i=1}^{n}(f*g)(i)$。有：

$$S(n)=\sum_{i=1}^{n}g(i)\sum_{ij\le n}(f*1)(j)-\sum_{i=2}^{n}S(\lfloor\frac{n}{i}\rfloor)$$

题号	读	会	过	知识点	注意点
A					
B					
C					
D					
E					
F					
G					
H					
I					
J					
K					
L					
M					

题号	读	会	过	知识点	注意点
A					
B					
C					
D					
E					
F					
G					
H					
I					
J					
K					
L					
M					

题号	读	会	过	知识点	注意点
A					
B					
C					
D					
E					
F					
G					
H					
I					
J					
K					
L					
M					

题号	读	会	过	知识点	注意点
A					
B					
C					
D					
E					
F					
G					
H					
I					
J					
K					
L					
M					