

ICPC Templates For Africamonkey

Africamonkey

2018 年 7 月 31 日

目录

| | | |
|----------|-------------|-----------|
| 1 | 莫队算法 | 5 |
| 1.1 | 普通莫队 | 5 |
| 1.2 | 树上莫队 | 5 |
| 2 | 字符串 | 7 |
| 2.1 | 哈希 | 7 |
| 2.2 | KMP | 7 |
| 2.3 | 可动态修改的 KMP | 8 |
| 2.4 | 扩展 KMP | 8 |
| 2.5 | Manacher | 9 |
| 2.6 | 最小表示法 | 10 |
| 2.7 | AC 自动机 | 10 |
| 2.8 | 后缀数组 | 11 |
| 2.8.1 | 倍增算法 | 11 |
| 2.8.2 | DC3 算法 | 12 |
| 2.8.3 | 小技巧：拼接字符串 | 13 |
| 2.9 | 后缀自动机 | 14 |
| 2.9.1 | 广义后缀自动机 | 16 |
| 2.10 | 回文树 | 16 |
| 3 | 数据结构 | 18 |
| 3.1 | ST 表 | 18 |
| 3.2 | K-D Tree | 19 |
| 3.3 | 左偏树 | 22 |
| 3.4 | 线段树小技巧 | 23 |
| 3.5 | Splay | 24 |
| 3.6 | 可持久化 Treap | 26 |
| 3.7 | 可持久化并查集 | 29 |

| | | |
|----------|---------------------------------|-----------|
| 4 | 树 | 29 |
| 4.1 | 树链剖分 | 29 |
| 4.2 | 点分治 | 32 |
| 4.3 | Link Cut Tree | 34 |
| 4.4 | 求子树的直径 | 36 |
| 4.5 | 虚树 | 39 |
| 5 | 图 | 40 |
| 5.1 | 欧拉回路 | 40 |
| 5.2 | 最短路径 | 41 |
| 5.2.1 | Dijkstra | 41 |
| 5.2.2 | SPFA | 42 |
| 5.3 | K 短路 | 43 |
| 5.4 | Tarjan | 46 |
| 5.5 | 2-SAT | 46 |
| 5.6 | 统治者树 (Dominator Tree) | 49 |
| 5.7 | 网络流 | 50 |
| 5.7.1 | 最大流 | 50 |
| 5.7.2 | 上下界有源汇网络流 | 51 |
| 5.7.3 | 上下界无源汇网络流 | 51 |
| 5.7.4 | 费用流 | 52 |
| 5.7.5 | zkw 费用流 | 53 |
| 6 | 数学 | 55 |
| 6.1 | 扩展欧几里得解同余方程 | 55 |
| 6.2 | 同余方程组 | 55 |
| 6.3 | 类欧几里得算法 | 56 |
| 6.4 | 卡特兰数 | 57 |
| 6.5 | 斯特林数 | 57 |
| 6.5.1 | 第一类斯特林数 | 57 |
| 6.5.2 | 第二类斯特林数 | 57 |
| 6.6 | 错排公式 | 57 |
| 6.7 | Lucas 定理 | 58 |
| 6.8 | 线性规划 | 58 |
| 6.8.1 | 单纯形法 | 58 |
| 6.8.2 | 对偶理论 | 61 |
| 6.9 | 高斯消元 | 61 |
| 6.9.1 | 行列式 | 61 |
| 6.9.2 | Matrix-Tree 定理 | 62 |
| 6.10 | 调和级数 | 62 |
| 6.11 | 曼哈顿距离的变换 | 62 |
| 6.12 | 数论函数变换 | 62 |
| 6.13 | 莫比乌斯反演 | 63 |

| | | |
|----------|---------------------------|-----------|
| 6.14 | 线性筛素数 | 63 |
| 6.15 | 杜教筛 | 63 |
| 6.16 | FFT | 65 |
| 6.16.1 | 普通 FFT | 65 |
| 6.16.2 | 模任意素数 FFT | 67 |
| 6.17 | FWT | 68 |
| 6.18 | 求原根 | 69 |
| 6.19 | NTT | 70 |
| 6.19.1 | NTT 常用原根表 | 71 |
| 6.19.2 | 多项式求逆元 | 75 |
| 6.19.3 | 多项式取对数 | 75 |
| 6.19.4 | 多项式取指数 | 76 |
| 6.20 | Berlekamp Messay 算法求线性递推式 | 76 |
| 6.21 | 幂和 | 79 |
| 6.22 | 蔡勒公式 | 79 |
| 6.23 | 皮克定理 | 80 |
| 6.24 | 组合数 lcm | 80 |
| 6.25 | 区间 lcm 的维护 | 80 |
| 7 | 几何 | 80 |
| 7.1 | 二维计算几何 | 80 |
| 7.1.1 | 计算几何误差修正 | 80 |
| 7.1.2 | 计算几何点类 | 80 |
| 7.1.3 | 计算几何线段类 | 81 |
| 7.2 | 凸包 | 83 |
| 7.3 | 半平面交 | 84 |
| 8 | 黑科技和杂项 | 85 |
| 8.1 | 找规律 | 85 |
| 8.2 | 分数类 | 88 |
| 8.3 | 取模整数类 | 90 |
| 8.4 | 多项式类 | 91 |
| 8.5 | 高精度计算 | 93 |
| 8.6 | 读入优化 | 96 |
| 8.6.1 | 普通读入优化 | 96 |
| 8.6.2 | HDU 专用读入优化 | 97 |
| 8.7 | O2 优化 | 98 |
| 8.8 | 位运算及其运用 | 98 |
| 8.8.1 | 枚举子集 | 98 |
| 8.8.2 | 求 1 的个数 | 98 |
| 8.8.3 | 求前缀 0 的个数 | 98 |
| 8.8.4 | 求后缀 0 的个数 | 98 |

| | | |
|-----------|--|-----------|
| 9 | Sublime Text | 98 |
| 9.1 | License | 98 |
| 9.2 | Preferences.sublime-settings | 99 |
| 10 | Vim | 99 |

1 莫队算法

1.1 普通莫队

分块块数为 \sqrt{n} 是最优的。

记每次进行 `add()` 操作的复杂度为 $O(A)$ ，`del()` 操作的复杂度为 $O(D)$ ，查询答案 `answer()` 的复杂度为 $O(S)$ 。

则总复杂度为 $O(n\sqrt{n}(A + D) + qS)$ 。

S 可以大一点，但必须保证 A, D 尽可能小。

```
1 struct Q { int l, r, sqrtl, id; } q[N];
2 int n, m, v[N], ans[N], nowans;
3 bool cmp(const Q &a, const Q &b) {
4     if (a.sqrtl != b.sqrtl) return a.sqrtl < b.sqrtl;
5     return a.r < b.r;
6 }
7 void change(int x) {
8     if (!v[x]) add(x);
9     else del(x);
10    v[x] ^= 1;
11 }
12 int main() {
13     .....
14     for (int i=1;i<=m;i++) q[i].sqrtl = q[i].l / sqrt(n), q[i].id = i;
15     sort(q+1, q+m+1, cmp);
16     int L=1, R=0;
17     memset(v, 0, sizeof(v));
18     for (int i=1;i<=m;i++) {
19         while (L<q[i].l) change(L++);
20         while (L>q[i].l) change(--L);
21         while (R<q[i].r) change(++R);
22         while (R>q[i].r) change(R--);
23         ans[q[i].id] = answer();
24     }
25     .....
26 }
```

1.2 树上莫队

分块块数为 \sqrt{n} 是最优的。

记每次进行 `add()` 操作的复杂度为 $O(A)$ ，`del()` 操作的复杂度为 $O(D)$ ，查询答案 `answer()` 的复杂度为 $O(S)$ 。

则总复杂度为 $O(n\sqrt{n}(A + D) + qS)$ 。

S 可以大一点，但必须保证 A, D 尽可能小。

```
1 struct Query { int l, r, id, l_group; } query[N];
2 struct EDGE { int adj, next; } edge[N*2];
3 int n, m, top, gh[N], c[N], reorder[N], deep[N], father[N], size[N], son[N], Top[N];
4 void addedge(int x, int y) {
```

```

5     edge[++top].adj = y;
6     edge[top].next = gh[x];
7     gh[x] = top;
8 }
9 void dfs(int x, int root=0) {
10     reorder[x] = ++top; father[x] = root; deep[x] = deep[root] + 1;
11     son[x] = 0; size[x] = 1; int dd = 0;
12     for (int p=gh[x]; p; p=edge[p].next)
13         if (edge[p].adj != root) {
14             dfs(edge[p].adj, x);
15             if (size[edge[p].adj] > dd) {
16                 son[x] = edge[p].adj;
17                 dd = size[edge[p].adj];
18             }
19             size[x] += size[edge[p].adj];
20         }
21 }
22 void split(int x, int tp) {
23     Top[x] = tp;
24     if (son[x]) split(son[x], tp);
25     for (int p=gh[x]; p; p=edge[p].next)
26         if (edge[p].adj != father[x] && edge[p].adj != son[x])
27             split(edge[p].adj, edge[p].adj);
28 }
29 int lca(int x, int y) {
30     int tx = Top[x], ty = Top[y];
31     while (tx != ty) {
32         if (deep[tx] < deep[ty]) {
33             swap(tx, ty);
34             swap(x, y);
35         }
36         x = father[tx];
37         tx = Top[x];
38     }
39     if (deep[x] < deep[y]) swap(x, y);
40     return y;
41 }
42 bool cmp(const Query &a, const Query &b) {
43     if (a.l_group != b.l_group) return a.l_group < b.l_group;
44     return reorder[a.r] < reorder[b.r];
45 }
46 int v[N], ans[N];
47
48 void upd(int x) {
49     if (!v[x]) add(x);
50     else del(x);
51     v[x] ^= 1;
52 }
53
54 void go(int &u, int taru, int v) {

```

```

55     int lca0 = lca(u, taru);
56     int lca1 = lca(u, v);    upd(lca1);
57     int lca2 = lca(taru, v); upd(lca2);
58     for (int x=u; x!=lca0; x=father[x]) upd(x);
59     for (int x=taru; x!=lca0; x=father[x]) upd(x);
60     u = taru;
61 }
62 int main() {
63     memset(gh, 0, sizeof(gh));
64     scanf("%d%d", &n, &m); top = 0;
65     for (int i=1;i<n;i++) {
66         int x,y; scanf("%d%d", &x, &y);
67         addedge(x, y); addedge(y, x);
68     }
69     top = 0; dfs(1); split(1, 1);
70     for (int i=1;i<=m;i++) {
71         if (reorder[query[i].l] > reorder[query[i].r])
72             swap(query[i].l, query[i].r);
73         query[i].id = i;
74         query[i].l_group = reorder[query[i].l] / sqrt(n);
75     }
76     sort(query+1, query+m+1, cmp);
77     int L=1,R=1; upd(1);
78     for (int i=1;i<=m;i++) {
79         go(L,query[i].l,R);
80         go(R,query[i].r,L);
81         ans[query[i].id] = answer();
82     }
83     .....
84 }

```

2 字符串

2.1 哈希

```

1  const int P=31,D=1000173169;
2  int n, pow[N], f[N]; char a[N];
3  int hash(int l, int r) { return (LL) (f[r]-(LL)f[l-1]*pow[r-l+1]%D+D)%D; }
4  int main() {
5      scanf("%d%s", &n, a+1);
6      pow[0] = 1;
7      for (int i=1;i<=n;i++) pow[i] = (LL)pow[i-1]*P%D;
8      for (int i=1;i<=n;i++) f[i] = (LL) ((LL)f[i-1]*P+a[i])%D;
9  }

```

2.2 KMP

接口: void kmp(int n, char *a, int m, char *b);

输入：模式串长度 n ，模式串 a ，匹配串长度 m ，匹配串 b

输出：依次输出每个匹配成功的起始位置

下标从 0 开始。

```
1 void kmp(int n, char* a, int m, char *b) {
2     int i, j;
3     for (nxt[0] = j = -1, i = 1; i < n; nxt[i++] = j) {
4         while (~j && a[j + 1] != a[i]) j = nxt[j];
5         if (a[j + 1] == a[i]) ++j;
6     }
7     for (j = -1, i = 0; i < m; ++i) {
8         while (~j && a[j + 1] != b[i]) j = nxt[j];
9         if (a[j + 1] == b[i]) ++j;
10        if (j == n - 1) {
11            printf("%d\n", i - n + 1);
12            j = nxt[j];
13        }
14    }
15 }
```

2.3 可动态修改的 KMP

支持：加入一个字符，删除一个字符。

时间复杂度： $O(n\alpha)$ ， α 为字符集大小。

代码中的字符为 '0' - '9'，可自行修改为 'a' - 'z'

```
1 char t[N];
2 int top, nxt[N], nxt_l[N][10];
3 inline void del_letter() { --top; }
4 inline void add_letter(char x) {
5     t[top++] = x;
6     int j = top-1;
7     memset(nxt_l[top], 0, sizeof(nxt_l[top]));
8     nxt[top] = nxt_l[top-1][x-'0'];
9     memcpy(nxt_l[top], nxt_l[nxt[top]], sizeof(nxt_l[nxt[top]]));
10    nxt_l[top][t[nxt[top]]-'0'] = nxt[top]+1;
11 }
```

2.4 扩展 KMP

接口：void ExtendedKMP(char *a, char *b, int *next, int *ret);

输出：

next: a 关于自己每个后缀的最长公共前缀

ret: a 关于 b 的每个后缀的最长公共前缀

EXKMP 的 next[i] 表示：从 i 到 n-1 的字符串 st 前缀和原串前缀的最长重叠长度。

```
1 void get_next(char *a, int *next) {
2     int i, j, k;
3     int n = strlen(a);
```



```

4     for (j = 0; j+1<n && a[j]==a[j+1];j++);
5     next[1] = j;
6     k = 1;
7     for (i=2;i<n;i++) {
8         int len = k+next[k], l = next[i-k];
9         if (l < len-i) {
10             next[i] = l;
11         } else {
12             for (j = max(0, len-i);i+j<n && a[j]==a[i+j];j++);
13             next[i] = j;
14             k = i;
15         }
16     }
17 }
18 void ExtendedKMP(char *a, char *b, int *next, int *ret) {
19     get_next(a, next);
20     int n = strlen(a), m = strlen(b);
21     int i, j, k;
22     for (j=0;j<n && j<m && a[j]==b[j];j++);
23     ret[0] = j;
24     k = 0;
25     for (i=1;i<m;i++) {
26         int len = k+ret[k], l = next[i-k];
27         if (l < len-i) {
28             ret[i] = l;
29         } else {
30             for (j = max(0, len-i);j<n && i+j<m && a[j]==b[i+j];j++);
31             ret[i] = j;
32             k = i;
33         }
34     }
35 }

```

2.5 Manacher

$p[i]$ 表示以 i 为对称轴的最长回文串长度

```

1 char st[N*2], s[N];
2 int len, p[N*2];
3
4 while (scanf("%s", s) != EOF) {
5     len = strlen(s);
6     st[0] = '$', st[1] = '#';
7     for (int i=1;i<=len;i++)
8         st[i*2] = s[i-1], st[i*2+1] = '#';
9     len = len * 2 + 2;
10    int mx = 0, id = 0, ans = 0;
11    for (int i=1;i<=len;i++) {
12        p[i] = (mx > i) ? min(p[id*2-i]+1, mx-i) : 1;
13        for (; st[i+p[i]] == st[i-p[i]]; ++p[i]) ;

```

```

14         if (p[i]+i > mx) mx = p[i]+i, id = i;
15         p[i] --;
16         if (p[i] > ans) ans = p[i];
17     }
18     printf("%d\n", ans);
19 }

```

2.6 最小表示法

```

1 string smallestRepresation(string s) {
2     int i, j, k, l;
3     int n = s.length();
4     s += s;
5     for (i=0, j=1; j<n; ) {
6         for (k=0; k<n && s[i+k]==s[j+k]; k++);
7         if (k>=n) break;
8         if (s[i+k]<s[j+k]) j+=k+1;
9         else {
10             l=i+k;
11             i=j;
12             j=max(l, j)+1;
13         }
14     }
15     return s.substr(i, n);
16 }

```

2.7 AC 自动机

```

1 struct Node {
2     int next[**Size of Alphabet**];
3     int terminal, fail;
4 } node[**Number of Nodes**];
5 int top;
6 void add(char *st) {
7     int len = strlen(st), x = 1;
8     for (int i=0; i<len; i++) {
9         int ind = trans(st[i]);
10        if (!node[x].next[ind])
11            node[x].next[ind] = ++top;
12        x = node[x].next[ind];
13    }
14    node[x].terminal = 1;
15 }
16 int q[**Number of Nodes**], head, tail;
17 void build() {
18     head = 0, tail = 1; q[1] = 1;
19     while (head != tail) {
20         int x = q[++head];

```

```

21      /*(when necessary) node[x].terminal != node[node[x].fail].terminal; */
22      for (int i=0;i<n;i++)
23          if (node[x].next[i]) {
24              if (x == 1) node[node[x].next[i]].fail = 1;
25              else {
26                  int y = node[x].fail;
27                  while (y) {
28                      if (node[y].next[i]) {
29                          node[node[x].next[i]].fail = node[y].next[i];
30                          break;
31                      }
32                      y = node[y].fail;
33                  }
34                  if (!node[node[x].next[i]].fail) node[node[x].next[i]].fail = 1;
35              }
36              q[++tail] = node[x].next[i];
37          }
38      }
39  }

```

2.8 后缀数组

2.8.1 倍增算法

参数 m 表示字符集的大小, 即 $0 \leq r_i < m$

```

1  #define rank rank2
2  int n, r[N], wa[N], wb[N], ws[N], sa[N], rank[N], height[N];
3  int cmp(int *r, int a, int b, int l, int n) {
4      if (r[a]==r[b]) {
5          if (a+l<n && b+l<n && r[a+l]==r[b+l])
6              return 1;
7      }
8      return 0;
9  }
10 void suffix_array(int m) {
11     int i, j, p, *x=wa, *y=wb, *t;
12     for (i=0;i<m;i++) ws[i]=0;
13     for (i=0;i<n;i++) ws[x[i]=r[i]]++;
14     for (i=1;i<m;i++) ws[i]+=ws[i-1];
15     for (i=n-1;i>=0;i--) sa[--ws[x[i]]]=i;
16     for (j=1;p=1;p<n;m=p,j<=1) {
17         for (p=0,i=n-j;i<n;i++) y[p++]=i;
18         for (i=0;i<n;i++) if (sa[i]>=j) y[p++]=sa[i]-j;
19         for (i=0;i<m;i++) ws[i]=0;
20         for (i=0;i<n;i++) ws[x[y[i]]]++;
21         for (i=1;i<m;i++) ws[i]+=ws[i-1];
22         for (i=n-1;i>=0;i--) sa[--ws[x[y[i]]]]=y[i];
23         for (t=x,x=y,y=t,x[sa[0]]=0,i=1,p=1;i<n;i++)
24             x[sa[i]]=cmp(y,sa[i-1],sa[i],j,n)?p-1:p++;

```

```

25     }
26     for (i=0;i<n;i++) rank[sa[i]]=i;
27     rank[n] = -1;
28 }
29 void calc_height() {
30     int j=0;
31     for (int i=0;i<n;i++)
32         if (rank[i])
33         {
34             while (r[i+j]==r[sa[rank[i]-1]+j]) j++;
35             height[rank[i]]=j;
36             if (j) j--;
37         }
38 }

```

2.8.2 DC3 算法

感谢浙江大学陈靖邦提供本模板。

```

1 namespace SA {
2 int sa[N], rk[N], ht[N], s[N<<1], t[N<<1], p[N], cnt[N], cur[N];
3 #define pushS(x) sa[cur[s[x]]--] = x
4 #define pushL(x) sa[cur[s[x]]++] = x
5 #define inducedSort(v) fill_n(sa, n, -1); fill_n(cnt, m, 0); \
6     for (int i = 0; i < n; i++) cnt[s[i]]++; \
7     for (int i = 1; i < m; i++) cnt[i] += cnt[i-1]; \
8     for (int i = 0; i < m; i++) cur[i] = cnt[i]-1; \
9     for (int i = n1-1; ~i; i--) pushS(v[i]); \
10    for (int i = 1; i < m; i++) cur[i] = cnt[i-1]; \
11    for (int i = 0; i < n; i++) if (sa[i] > 0 && t[sa[i]-1]) pushL(sa[i]-1); \
12    for (int i = 0; i < m; i++) cur[i] = cnt[i]-1; \
13    for (int i = n-1; ~i; i--) if (sa[i] > 0 && !t[sa[i]-1]) pushS(sa[i]-1)
14 void sais(int n, int m, int *s, int *t, int *p) {
15     int n1 = t[n-1] = 0, ch = rk[0] = -1, *s1 = s+n;
16     for (int i = n-2; ~i; i--) t[i] = s[i] == s[i+1] ? t[i+1] : s[i] > s[i+1];
17     for (int i = 1; i < n; i++) rk[i] = t[i-1] && !t[i] ? (p[n1] = i, n1++) : -1;
18     inducedSort(p);
19     for (int i = 0, x, y; i < n; i++) if (~x = rk[sa[i]]) {
20         if (ch < 1 || p[x+1] - p[x] != p[y+1] - p[y]) ch++;
21         else for (int j = p[x], k = p[y]; j <= p[x+1]; j++, k++)
22             if ((s[j]<<1|t[j]) != (s[k]<<1|t[k])) {ch++; break;}
23         s1[y = x] = ch;
24     }
25     if (ch+1 < n1) sais(n1, ch+1, s1, t+n, p+n1);
26     else for (int i = 0; i < n1; i++) sa[s1[i]] = i;
27     for (int i = 0; i < n1; i++) s1[i] = p[sa[i]];
28     inducedSort(s1);
29 }
30 template<typename T>
31 int mapCharToInt(int n, const T *str) {

```

```

32     int m = *max_element(str, str+n);
33     fill_n(rk, m+1, 0);
34     for (int i = 0; i < n; i++) rk[str[i]] = 1;
35     for (int i = 0; i < m; i++) rk[i+1] += rk[i];
36     for (int i = 0; i < n; i++) s[i] = rk[str[i]] - 1;
37     return rk[m];
38 }
39 // Ensure that str[n] is the unique lexicographically smallest character in str.
40 template<typename T>
41 void suffixArray(int n, const T *str) {
42     int m = mapCharToInt(++n, str);
43     sais(n, m, s, t, p);
44     for (int i = 0; i < n; i++) rk[sa[i]] = i;
45     for (int i = 0, h = ht[0] = 0; i < n-1; i++) {
46         int j = sa[rk[i]-1];
47         while (i+h < n && j+h < n && s[i+h] == s[j+h]) h++;
48         if (ht[rk[i]] = h) h--;
49     }
50 }
51 };

```

2.8.3 小技巧：拼接字符串

接口：

int gao1(int l, int r, int c, int p); 区间 $[l, r)$ 中保证第 0 位到第 $c-1$ 位都是相同的（设为字符串 s ），现在我们在 s 后面接一个字符 p ，得到一个新的字符串 s' 。返回值为最小的 k 满足后缀 $sa[k]$ 前 $c+1$ 位为 s'

int gao2(int l, int r, int c, int p); 区间 $[l, r)$ 中保证第 0 位到第 $c-1$ 位都是相同的（设为字符串 s ），现在我们在 s 后面接一个后缀 $sa[p]$ ，得到一个新的字符串 s' 。返回值为最小的 k 满足后缀 $sa[k]$ 前 $c + \text{len}(sa[p])$ 位为 s'

```

1  int gao1(int l, int r, int c, int p) {
2      --l;
3      while (l+1 < r) {
4          int md = (l+r) >> 1;
5          if (sa[md] + c < n && s[sa[md] + c] >= p) r = md; else l = md;
6      }
7      return r;
8  }
9  int gao2(int l, int r, int c, int p) {
10     --l;
11     while (l+1 < r) {
12         int md = (l+r) >> 1;
13         if (sa[md] + c < n && rk[sa[md] + c] >= p) r = md; else l = md;
14     }
15     return r;
16 }

```

示例调用：

```

1 suf1[m] = -1, suf2[m] = n;
2 for (int i = m - 1; i >= 0; --i) {
3     int l = gao1(0, n, 0, t[i]), r = gao1(0, n, 0, t[i]);
4     suf1[i] = gao2(l, r, 1, suf1[i + 1]);
5     suf2[i] = gao2(l, r, 1, suf2[i + 1]);
6 }

```

2.9 后缀自动机

下面的代码是求两个串的 LCS（最长公共子串）。

```

1 #include <bits/stdc++.h>
2
3 #define N 500001
4 #define M (N << 1)
5
6 using namespace std;
7
8 char st[N];
9 int pre[M], son[26][M], step[M], refer[M], size[M], tmp[M], topo[M], last, total;
10
11 int apply(int x, int now) {
12     step[++total] = x;
13     refer[total] = now;
14     return total;
15 }
16
17 void extend(char x, int now) {
18     int p = last, np = apply(step[p]+1, now);
19     size[np] = 1;
20     for (; p && !son[x][p]; p=pre[p]) son[x][p] = np;
21     if (!p) pre[np] = 1;
22     else {
23         int q = son[x][p];
24         if (step[p]+1 == step[q]) pre[np] = q;
25         else {
26             int nq = apply(step[p]+1, now);
27             for (int i=0; i<26; i++) son[i][nq] = son[i][q];
28             pre[nq] = pre[q];
29             pre[q] = pre[np] = nq;
30             for (; p && son[x][p]==q; p=pre[p]) son[x][p] = nq;
31         }
32     }
33     last = np;
34 }
35 void init() {
36     last = total = 0;
37     last = apply(0, 0);
38     scanf("%s", st);

```

```

39     int n = strlen(st);
40     for (int i = 0; i <= n * 2; ++i) {
41         pre[i] = step[i] = refer[i] = size[i] = tmp[i] = topo[i] = 0;
42         for (int j = 0; j < 26; ++j)
43             son[j][i] = 0;
44     }
45     for (int i = 0; i < n; ++i)
46         extend(st[i] - 'a', i);
47     for (int i = 1; i <= total; ++i)
48         tmp[step[i]] ++;
49     for (int i = 1; i <= n; ++i)
50         tmp[i] += tmp[i - 1];
51     for (int i = 1; i <= total; ++i)
52         topo[tmp[step[i]]--] = i;
53     for (int i = total; i; --i)
54         size[pre[topo[i]]] += size[topo[i]];
55 }
56 int main() {
57     init();
58     int p = 1, now = 0, ans = 0;
59     scanf("%s", st);
60     for (int i=0; st[i]; i++) {
61         int index = st[i] - 'a';
62         for (; p && !son[index][p]; p = pre[p], now = step[p]) ;
63         if (!p) p = 1;
64         if (son[index][p]) {
65             p = son[index][p];
66             now++;
67             if (now > ans) ans = now;
68         }
69     }
70     printf("%d\n", ans);
71     return 0;
72 }

```

一些定义和性质 $\text{Right}(\text{str})$ 表示 str 在母串 S 中所有出现的结束位置集合

一个状态 s 表示的所有子串 Right 集合相同，为 $\text{Right}(s)$

$\text{Parent}(s)$ 满足 $\text{Right}(s)$ 是 $\text{Right}(\text{Parent}(s))$ 的真子集，并且 $\text{Right}(\text{Parent}(s))$ 的大小最小

Parent 函数可以表示一个树形结构。不妨叫它 Parent 树

一个 Right 集合和一个长度定义了一个子串

对于状态 s ，使得 $\text{Right}(s)$ 合法的子串长度是一个区间 $[\min(s), \max(s)]$

$\max(\text{Parent}(s)) = \min(s) - 1$

令 $\text{refer}(s)$ 表示产生 s 状态的字符所在位置。则 $\text{Right}(s)$ 的合法子串的起始位置为 $[\text{refer}(s) - \max(s) + 1, \text{refer}(s) - \min(s) + 1]$ ，即 $[\text{refer}(s) - \max(s) + 1, \text{refer}(s) - \max(\text{Parent}(s))]$

代码中变量含义 $\text{pre}[s]$ 为上述定义中的 $\text{Parent}(s)$

$\text{step}[s]$ 为从初始状态走到 s 状态最多需要多少步

$\text{refer}[s]$ 为上述定义中的 $\text{refer}(s)$
 $\text{size}[s]$ 为 $\text{Right}(s)$ 集合的大小
 $\text{topo}[s]$ 为 Parent 树的拓扑序，根（初始状态）在前

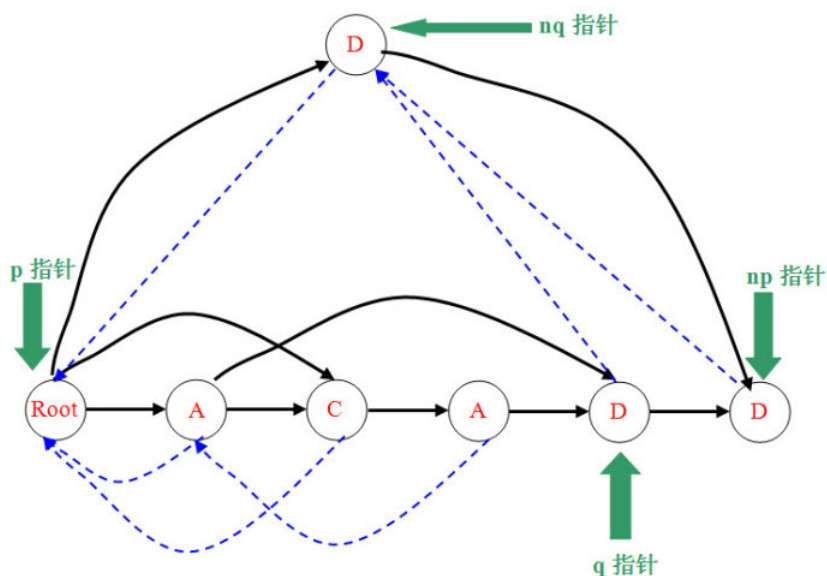


图 1: ACADD 构成的后缀自动机

我们发现 fail 构出一棵前缀树
 和后缀树相同，为了使每个前缀都是叶子结点，我们不妨在串 s 前加入一个没出现的字符 '#'

2.9.1 广义后缀自动机

先建 Trie，再按照 *BFS* 序建后缀自动机。从节点 x 开始向子树更新时，其所有儿子都从同一个 last，即 $\text{last}[x]$ 更新。

2.10 回文树

【URAL2040】Palindromes and Super Abilities 2

逐个添加字符串 S 里的字符 S_1, S_2, \dots, S_n 。每次添加字符后，他想知道添加字符后将出现多少个新的本质不同的回文子串。字符集为 $\{a, b\}$

```

1 #include <bits/stdc++.h>
2 #define N 5000020
3
4 char st[N], answer[N];
5 int n;
6
7 struct PAM {
8     int n, tot, last;
9     int len[N], fail[N], next[N][2];
10     void init() {
11         n=0; tot=1;
    
```

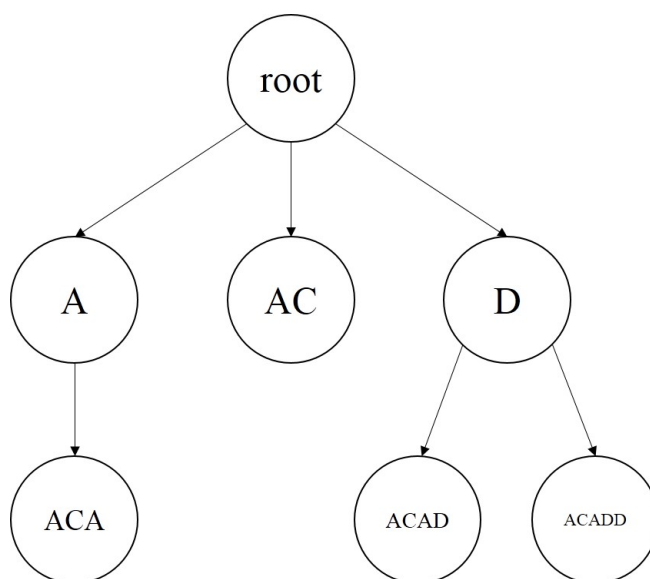



图 2: 串 ACADD 按 fail 构出的前缀树, 与图 1 对应

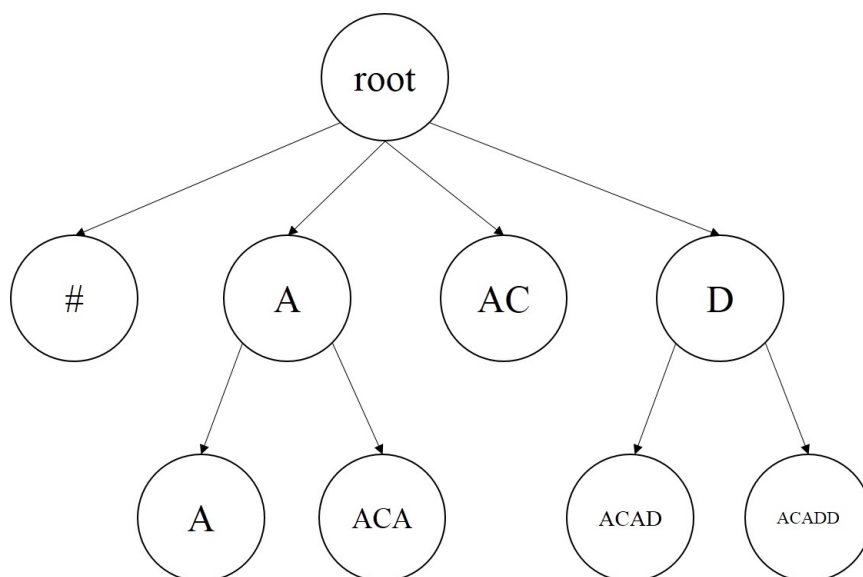


图 3: 串 #ACADD 按 fail 构出的前缀树

```

12     len[1]=-1; fail[1]=0;
13     len[0]=+0; fail[0]=1;
14     last=1;
15 }
16 int get_fail(int x) {
17     for (; st[n-len[x]-1]!=st[n]; x=fail[x]);
18     return x;
19 }
20 void insert(char c) {
21     ++n; int cur=get_fail(last); // 判断上一个串的前一个位置和新添加的位置是否相
        同, 相同则说明构成回文。否则找 fail 指针。
22     if (!next[cur][c]) {
23         ++tot;
24         len[tot]=len[cur]+2;
25         fail[tot]=next[get_fail(fail[cur])][c];
26         next[cur][c]=tot;
27         answer[n]='1';
28     } else {
29         answer[n]='0';
30     }
31     last=next[cur][c];
32 }
33 } pam;
34
35 int main() {
36     scanf("%s", st+1); n=strlen(st+1);
37     pam.init();
38     for (int i=1;i<=n;i++) pam.insert(st[i]-'a');
39     puts(answer+1);
40     return 0;
41 }

```

3 数据结构

3.1 ST 表

```

1 int Log[N], f[17][N];
2 int ask(int x, int y) {
3     int k=Log[y-x+1];
4     return max(f[k][x], f[k][y-(1<<k)+1]);
5 }
6 int main() {
7     for (int i=2; i<=n; i++) Log[i]=Log[i>>1]+1;
8     for (int j=1; j<K; j++)
9         for (int i=1; i+(1<<j-1)<=n; i++)
10             f[j][i]=max(f[j-1][i], f[j-1][i+(1<<j-1)]);
11 }

```

3.2 K-D Tree

```
1 int n, cmp_d, root, id[N];
2
3 struct node {
4     int d[2], l, r, Max[2], Min[2], val, sum, f;
5 } t[N];
6
7 inline bool cmp(const node &a, const node &b) {
8     if (a.d[cmp_d] != b.d[cmp_d]) return a.d[cmp_d] < b.d[cmp_d];
9     return a.d[cmp_d ^ 1] < b.d[cmp_d ^ 1];
10 }
11
12 inline void umax(int &a, int b) {
13     if (b > a) a = b;
14 }
15
16 inline void umin(int &a, int b) {
17     if (b < a) a = b;
18 }
19
20 inline void up(int x, int y) {
21     umax(t[x].Max[0], t[y].Max[0]);
22     umin(t[x].Min[0], t[y].Min[0]);
23     umax(t[x].Max[1], t[y].Max[1]);
24     umin(t[x].Min[1], t[y].Min[1]);
25 }
26
27 int build(int l, int r, int D, int f) {
28     int mid = (l + r) / 2;
29     cmp_d = D;
30     nth_element(t + l + 1, t + mid + 1, t + r + 1, cmp);
31     id[t[mid].f] = mid;
32     t[mid].f = f;
33     t[mid].Max[0] = t[mid].Min[0] = t[mid].d[0];
34     t[mid].Max[1] = t[mid].Min[1] = t[mid].d[1];
35     t[mid].val = t[mid].sum = 0;
36     if (l != mid) t[mid].l = build(l, mid - 1, !D, mid);
37     else t[mid].l = 0;
38     if (r != mid) t[mid].r = build(mid + 1, r, !D, mid);
39     else t[mid].r = 0;
40     if (t[mid].l) up(mid, t[mid].l);
41     if (t[mid].r) up(mid, t[mid].r);
42     return mid;
43 }
44
45 // 将编号为 x 的点的权值增加 p
46 // 请注意，此处的 x 是经过排序的。你需要将点的坐标先作映射。
47 void change(int x, int p) {
48     x = id[x];
```

```

49     for (t[x].val += p; x; x = t[x].f)
50         t[x].sum += p;
51 }
52
53 inline long long sqr(long long x) {
54     return x * x;
55 }
56
57 // 欧几里得距离的平方, 下界
58 inline long long euclid_lower_bound(const node &a, int X, int Y) {
59     return sqr(max(max(X - a.Max[0], a.Min[0] - X), 0)) +
60         sqr(max(max(Y - a.Max[1], a.Min[1] - Y), 0));
61 }
62
63 // 欧几里得距离的平方, 上界
64 inline long long euclid_upper_bound(const node &a, int X, int Y) {
65     return max(sqr(X - a.Min[0]), sqr(X - a.Max[0])) +
66         max(sqr(Y - a.Min[1]), sqr(Y - a.Max[1]));
67 }
68
69 // 曼哈顿距离, 下界
70 inline long long manhattan_lower_bound(const node &a, int X, int Y) {
71     return max(a.Min[0] - X, 0) + max(X - a.Max[0], 0) +
72         max(a.Min[1] - Y, 0) + max(Y - a.Max[1], 0);
73 }
74
75 // 曼哈顿距离, 上界
76 inline long long manhattan_upper_bound(const node &a, int X, int Y) {
77     return max(abs(X - a.Max[0]), abs(a.Min[0] - X)) +
78         max(abs(Y - a.Max[1]), abs(a.Min[1] - Y));
79 }
80
81 // 添加一个点 (注意此处的添加可能导致这棵树不平衡, 慎用!)
82 void add(int k) {
83     t[k].Max[0] = t[k].Min[0] = t[k].d[0];
84     t[k].Max[1] = t[k].Min[1] = t[k].d[1];
85     t[k].val = t[k].sum = 0;
86     t[k].l = t[k].r = t[k].f = 0;
87     if (!root) {
88         root = k;
89         return;
90     }
91     int p = root;
92     int D = 0;
93     while (1) {
94         up(p, k);
95         if (t[k].d[D] <= t[p].d[D]) {
96             if (t[p].l) p = t[p].l;
97             else {
98                 t[p].l = k;

```

```

99         t[k].f = p;
100         return;
101     }
102     } else {
103         if (t[p].r) p = t[p].r;
104         else {
105             t[p].r = k;
106             t[k].f = p;
107             return;
108         }
109     }
110     D ^= 1;
111 }
112 }
113
114 inline long long getdis(const node &a, int X, int Y) {
115     return sqr(a.d[0] - X) + sqr(a.d[1] - Y);
116 }
117
118 // 此处询问距离点 (X, Y) 最远的一个点的距离, ans 需传入无穷小
119 void ask(int p, int X, int Y, long long &ans) {
120     if (!p) return;
121     ans = max(ans, getdis(t[p], X, Y));
122     long long dl = t[p].l ? euclid_upper_bound(t[t[p].l], X, Y) : 0;
123     long long dr = t[p].r ? euclid_upper_bound(t[t[p].r], X, Y) : 0;
124     if (dl > dr) {
125         if (dl > ans) ask(t[p].l, X, Y, ans);
126         if (dr > ans) ask(t[p].r, X, Y, ans);
127     } else {
128         if (dr > ans) ask(t[p].r, X, Y, ans);
129         if (dl > ans) ask(t[p].l, X, Y, ans);
130     }
131 }
132
133 // 查询矩形范围内所有点的权值和
134 int ask(int p, int x1, int y1, int x2, int y2) {
135     if (t[p].Min[0] > x2 || t[p].Max[0] < x1 || t[p].Min[1] > y2 || t[p].Max[1] < y1
136         ) return 0;
137     if (t[p].Min[0] >= x1 && t[p].Max[0] <= x2 && t[p].Min[1] >= y1 && t[p].Max[1]
138         <= y2) return t[p].sum;
139     int s = 0;
140     if (t[p].d[0] >= x1 && t[p].d[0] <= x2 && t[p].d[1] >= y1 && t[p].d[1] <= y2) s
141         += t[p].val;
142     if (t[p].l) s += ask(t[p].l, x1, y1, x2, y2);
143     if (t[p].r) s += ask(t[p].r, x1, y1, x2, y2);
144     return s;
145 }
146
147 int main() {
148     while (~scanf("%d", &n)) {

```

```

146     for (int i = 1; i <= n; ++i) {
147         int x, y, type;
148         scanf("%d%d%d", &x, &y, &type);
149         t[i].d[0] = x;
150         t[i].d[1] = y;
151     }
152     root = build(1, n, 0, 0);
153 }
154 }

```

3.3 左偏树

左偏树是一个可并堆。

下面的程序写的是一个小根堆，如果需要改成大根堆请在注释了 here 那行修改。

接口：

void push(const T &x); 插入一个元素。

void merge(leftist &x); 合并两个堆。注意，合并后原来那个堆将不可访问。

T top() const; 返回堆顶元素。

void pop(); 删除堆顶元素。

int size() const; 返回堆的大小。

```

1  template <class T>
2  class leftist {
3  public:
4      struct node {
5          T key;
6          int dist;
7          node *l, *r;
8      };
9      leftist() : root(NULL), s(0) {}
10     void push(const T &x) {
11         leftist y;
12         y.s = 1;
13         y.root = new node;
14         y.root -> key = x;
15         y.root -> dist = 0;
16         y.root -> l = y.root -> r = NULL;
17         merge(y);
18     }
19     node* merge(node *x, node *y) {
20         if (x == NULL) return y;
21         if (y == NULL) return x;
22         if (y -> key < x -> key) swap(x, y); //here
23         x -> r = merge(x -> r, y);
24         int ld = x -> l ? x -> l -> dist : -1;
25         int rd = x -> r ? x -> r -> dist : -1;
26         if (ld < rd) swap(x -> l, x -> r);
27         if (x -> r == NULL) x -> dist = 0;
28         else x -> dist = x -> r -> dist + 1;

```

```

29     return x;
30 }
31 void merge(leftist &x) {
32     root = merge(root, x.root);
33     s += x.s;
34 }
35 T top() const {
36     if (root == NULL) return T();
37     return root -> key;
38 }
39 void pop() {
40     if (root == NULL) return;
41     node *p = root;
42     root = merge(root -> l, root -> r);
43     --s;
44     delete p;
45 }
46 int size() const {
47     return s;
48 }
49 private:
50     node* root;
51     int s;
52 };

```

3.4 线段树小技巧

给定一个序列 a ，寻找一个最大的 i 使得 $i \leq y$ 且满足一些条件（如 $a[i] \geq w$ ，那么需要在线段树维护 a 的区间最大值）

```

1 int queryl(int p, int left, int right, int y, int w) {
2     if (right <= y) {
3         if (! __condition__ ) return -1;
4         else if (left == right) return left;
5     }
6     int mid = (left + right) / 2;
7     if (y <= mid) return queryl(p<<1|0, left, mid, y, w);
8     int ret = queryl(p<<1|1, mid+1, right, y, w);
9     if (ret != -1) return ret;
10    return queryl(p<<1|0, left, mid, y, w);
11 }

```

给定一个序列 a ，寻找一个最小的 i 使得 $i \geq x$ 且满足一些条件（如 $a[i] \geq w$ ，那么需要在线段树维护 a 的区间最大值）

```

1 int queryr(int p, int left, int right, int x, int w) {
2     if (left >= x) {
3         if (! __condition__ ) return -1;
4         else if (left == right) return left;
5     }

```

```

6   int mid = (left + right) / 2;
7   if (x > mid) return queryr(p<<1|1, mid+1, right, x, w);
8   int ret = queryr(p<<1|0, left, mid, x, w);
9   if (ret != -1) return ret;
10  return queryr(p<<1|1, mid+1, right, x, w);
11 }

```

3.5 Splay

接口：

ADD $x\ y\ d$: 将 $[x, y]$ 的所有数加上 d

REVERSE $x\ y$: 将 $[x, y]$ 翻转

INSERT $x\ p$: 将 p 插入到第 x 个数的后面

DEL x : 将第 x 个数删除

```

1  struct SPLAY {
2      struct NODE {
3          int w, min;
4          int son[2], size, father, rev, lazy;
5      } node[N];
6      int top, rt;
7      void pushdown(int x) {
8          if (!x) return;
9          if (node[x].rev) {
10             node[node[x].son[0]].rev ^= 1;
11             node[node[x].son[1]].rev ^= 1;
12             swap(node[x].son[0], node[x].son[1]);
13             node[x].rev = 0;
14         }
15         if (node[x].lazy) {
16             node[node[x].son[0]].lazy += node[x].lazy;
17             node[node[x].son[1]].lazy += node[x].lazy;
18             node[x].w += node[x].lazy;
19             node[x].min += node[x].lazy;
20             node[x].lazy = 0;
21         }
22     }
23     void pushup(int x) {
24         if (!x) return;
25         pushdown(node[x].son[0]);
26         pushdown(node[x].son[1]);
27         node[x].size = node[node[x].son[0]].size + node[node[x].son[1]].size + 1;
28         node[x].min = node[x].w;
29         if (node[x].son[0]) node[x].min = min(node[x].min, node[node[x].son[0]].min)
30             ;
31         if (node[x].son[1]) node[x].min = min(node[x].min, node[node[x].son[1]].min)
32             ;
33     }
34     void sc(int x, int y, int w) {

```



```

33     node[x].son[w] = y;
34     node[y].father = x;
35     pushup(x);
36 }
37 void _ins(int w) {
38     top++;
39     node[top].w = node[top].min = w;
40     node[top].son[0] = node[top].son[1] = 0;
41     node[top].size = 1; node[top].father = 0; node[top].rev = 0;
42 }
43 void init() {
44     top = 0;
45     _ins(0); _ins(0); rt=1;
46     sc(1, 2, 1);
47 }
48 void rotate(int x) {
49     if (!x) return;
50     int y = node[x].father;
51     int w = node[y].son[1]==x;
52     sc(y, node[x].son[w^1], w);
53     sc(node[y].father, x, node[node[y].father].son[1]==y);
54     sc(x, y, w^1);
55 }
56 int q[N];
57 void flushdown(int x) {
58     int t=0; for (; x; x=node[x].father) q[++t]=x;
59     for (; t; t--) pushdown(q[t]);
60 }
61 void Splay(int x, int root=0) {
62     flushdown(x);
63     while (node[x].father != root) {
64         int y=node[x].father;
65         int w=node[y].son[1]==x;
66         if (node[y].father != root && node[node[y].father].son[w]==y) rotate(y);
67         rotate(x);
68     }
69 }
70 int find(int k) {
71     Splay(rt);
72     while (1) {
73         pushdown(rt);
74         if (node[node[rt].son[0]].size+1==k) {
75             Splay(rt);
76             return rt;
77         } else
78         if (node[node[rt].son[0]].size+1<k) {
79             k-=node[node[rt].son[0]].size+1;
80             rt=node[rt].son[1];
81         } else {
82             rt=node[rt].son[0];

```

```

83         }
84     }
85 }
86 int split(int x, int y) {
87     int fx = find(x);
88     int fy = find(y+2);
89     Splay(fx);
90     Splay(fy, fx);
91     return node[fy].son[0];
92 }
93 void add(int x, int y, int d) { //add d to each number in a[x]...a[y]
94     int t = split(x, y);
95     node[t].lazy += d;
96     Splay(t); rt=t;
97 }
98 void reverse(int x, int y) { // reverse the x-th to y-th elements
99     int t = split(x, y);
100    node[t].rev ^= 1;
101    Splay(t); rt=t;
102 }
103 void insert(int x, int p) { // insert p after the x-th element
104     int fx = find(x+1);
105     int fy = find(x+2);
106     Splay(fx);
107     Splay(fy, fx);
108     _ins(p);
109     sc(fy, top, 0);
110     Splay(top); rt=top;
111 }
112 void del(int x) { // delete the x-th element in Splay
113     int fx = find(x), fy = find(x+2);
114     Splay(fx); Splay(fy, fx);
115     node[fy].son[0] = 0;
116     Splay(fy); rt=fy;
117 }
118 } tree;

```

3.6 可持久化 Treap

接口：

void insert(int x, char c); 在当前第 x 个字符后插入 c

void del(int x, int y); 删除第 x 个字符到第 y 个字符

void copy(int l, int r, int x); 复制第 l 个字符到第 r 个字符，然后粘贴到第 x 个字符后

void reverse(int x, int y); 翻转第 x 个到第 y 个字符

char query(int k); 表示询问当前第 x 个字符是什么

```

1 #define mod 1000000007
2 struct Treap {
3     struct Node {

```

```

4      char key;
5      bool reverse;
6      int lc, rc, size; // if size is long long, remember here
7  } node[N];
8  int n, root, rd;
9  int Rand() { rd = (rd * 20372052LL + 25022087LL) % mod; return rd; }
10
11  /*
12  LL Rand() {
13      LL t1 = rand() % 32768;
14      LL t2 = rand() % 32768;
15      LL t3 = rand() % 32768;
16      LL t4 = rand() % 32768;
17      return ((t1 * 32768) + t2) * 32768 + t3) * 32768 + t4;
18  }
19  */
20
21  void init() {
22      n = root = 0;
23  }
24  inline int copy(int x) {
25      node[++n] = node[x]; return n;
26  }
27  inline void pushdown(int x) {
28      if (!node[x].reverse) return;
29      if (node[x].lc) node[x].lc = copy(node[x].lc);
30      if (node[x].rc) node[x].rc = copy(node[x].rc);
31      swap(node[x].lc, node[x].rc);
32      node[node[x].lc].reverse ^= 1;
33      node[node[x].rc].reverse ^= 1;
34      node[x].reverse = 0;
35  }
36  inline void pushup(int x) {
37      node[x].size = node[node[x].lc].size + node[node[x].rc].size + 1;
38  }
39  int merge(int u, int v) {
40      if (!u || !v) return u+v;
41      pushdown(u); pushdown(v);
42      int t = Rand() % (node[u].size + node[v].size), r; // if size is long long,
         remember here
43      if (t < node[u].size) {
44          r = copy(u);
45          node[r].rc = merge(node[u].rc, v);
46      } else {
47          r = copy(v);
48          node[r].lc = merge(u, node[v].lc);
49      }
50      pushup(r);
51      return r;
52  }

```

```

53  int split(int u, int x, int y) { // if size is long long, remember here
54      if (x > y) return 0;
55      pushdown(u);
56      if (x == 1 && y == node[u].size) return copy(u);
57      if (y <= node[node[u].lc].size) return split(node[u].lc, x, y);
58      int t = node[node[u].lc].size + 1; // if size is long long, remember here
59      if (x > t) return split(node[u].rc, x-t, y-t);
60      int num = copy(u);
61      node[num].lc = split(node[u].lc, x, t-1);
62      node[num].rc = split(node[u].rc, 1, y-t);
63      pushup(num);
64      return num;
65  }
66  void insert(int x, char c) {
67      int t1 = split(root, 1, x), t2 = split(root, x+1, node[root].size);
68      node[++n].key = c;
69      node[n].lc = node[n].rc = 0;
70      node[n].reverse = 0;
71      pushup(n);
72      root = merge(merge(t1, n), t2);
73  }
74  void del(int x, int y) {
75      int t1 = split(root, 1, x-1), t2 = split(root, y+1, node[root].size);
76      root = merge(t1, t2);
77  }
78  void copy(int l, int r, int x) {
79      int t1 = split(root, 1, x), t2 = split(root, 1, r), t3 = split(root, x+1,
80          node[root].size);
81      root = merge(merge(t1, t2), t3);
82  }
83  void reverse(int x, int y) {
84      int t1 = split(root, 1, x-1), t2 = split(root, x, y), t3 = split(root, y+1,
85          node[root].size);
86      node[t2].reverse ^= 1;
87      root = merge(merge(t1, t2), t3);
88  }
89  char query(int k) {
90      int x = root;
91      while (1) {
92          pushdown(x);
93          if (k <= node[node[x].lc].size) x = node[x].lc;
94          else
95              if (k == node[node[x].lc].size + 1) return node[x].key;
96              else
97                  k -= node[node[x].lc].size + 1, x = node[x].rc;
98      }
99  }

```

3.7 可持久化并查集

接口：

void init() 初始化

void merge(int x, int y, int time) 在 time 时刻将 x 和 y 连一条边，注意加边顺序必须按 time 从小到大加边

void GetFather(int x, int time) 询问 time 时刻及以前的连边状态中，x 所属的集合

```
1 namespace pers_union {
2     const int inf = 0x3f3f3f3f;
3     int father[N], Father[N], Time[N];
4     vector<int> e[N];
5     void init() {
6         for (int i=1;i<=n;i++) {
7             father[i] = i;
8             Father[i] = i;
9             Time[i] = inf;
10            e[i].clear();
11            e[i].push_back(i);
12        }
13    }
14    int getfather(int x) {
15        return (father[x] == x) ? x : father[x] = getfather(father[x]);
16    }
17    int GetFather(int x, int time) {
18        return (Time[x] <= time) ? GetFather(Father[x], time) : x;
19    }
20    void merge(int x, int y, int time) {
21        int fx = getfather(x), fy = getfather(y);
22        if (fx == fy) return;
23        if (e[fx].size() > e[fy].size()) swap(fx, fy);
24        father[fx] = fy;
25        Father[fx] = fy;
26        Time[fx] = time;
27        for (int i=0;i<e[fx].size();i++) {
28            e[fy].push_back(e[fx][i]);
29        }
30    }
31 };
```

4 树

4.1 树链剖分

接口：

void addedge(int x, int y); 将 x 到 y 连边，注意这是单向边
 void dfs(int x, int root = 0); 从 x 开始遍历整棵树
 void split(int x, int tp); 划分轻重链
 int lca(int x, int y); 求 x 和 y 的 lca
 int query(int x, int y); 求 x 到 y 经过的点数
 int skip(int x, int k); 求从 x 向根方向跳 k 步到达的节点（若超出根，则返回 0）
 void get_data(int x, int y); 将 x 到 y 路径上的重链找出来，存在 seg[0] 中
 Debug 技巧：换一个根来 dfs 以测试程序是否能通过 $father[i] > i$ 的数据

```

1  struct EDGE {
2      int adj, next;
3  } edge[N * 2];
4
5  int n, gh[N], top, s_top;
6  int father[N], deep[N], son[N], size[N], Top[N], dfn[N], rdfs[N];
7
8  void addedge(int x, int y) {
9      edge[++top].adj = y;
10     edge[top].next = gh[x];
11     gh[x] = top;
12 }
13
14 void dfs(int x, int root = 0) {
15     father[x] = root;
16     deep[x] = deep[root] + 1;
17     son[x] = 0;
18     size[x] = 1;
19     int dd = 0;
20     for (int p = gh[x]; p; p = edge[p].next)
21         if (edge[p].adj != root) {
22             dfs(edge[p].adj, x);
23             if (size[edge[p].adj] > dd) {
24                 dd = size[edge[p].adj];
25                 son[x] = edge[p].adj;
26             }
27             size[x] += size[edge[p].adj];
28         }
29 }
30
31 void split(int x, int tp) {
32     Top[x] = tp; dfn[x] = ++s_top; rdfs[s_top] = x;
33     if (son[x]) split(son[x], tp);
34     for (int p = gh[x]; p; p = edge[p].next)
35         if (edge[p].adj != father[x] && edge[p].adj != son[x])
36             split(edge[p].adj, edge[p].adj);
37 }
38
39 int lca(int x, int y) {
40     int tx = Top[x], ty = Top[y];

```

```

41     while (tx != ty) {
42         if (deep[tx] < deep[ty]) {
43             swap(tx, ty);
44             swap(x, y);
45         }
46         x = father[tx];
47         tx = Top[x];
48     }
49     if (deep[x] < deep[y])
50         swap(x, y);
51     return y;
52 }
53
54 int query(int x, int y) {
55     int tx = Top[x], ty = Top[y];
56     int ans = 0;
57     while (tx != ty) {
58         if (deep[tx] < deep[ty]) {
59             swap(tx, ty);
60             swap(x, y);
61         }
62         ans += dfn[x] - dfn[tx] + 1;
63         x = father[tx];
64         tx = Top[x];
65     }
66     if (deep[x] < deep[y])
67         swap(x, y);
68     ans += dfn[x] - dfn[y] + 1;
69     return ans;
70 }
71
72 int skip(int x, int k) {
73     int tx = Top[x];
74     while (tx) {
75         if (k < dfn[x] - dfn[tx] + 1) {
76             return rdfs[ dfn[x] - k ];
77         } else {
78             k -= dfn[x] - dfn[tx] + 1;
79             x = father[tx];
80             tx = Top[x];
81         }
82     }
83     return 0;
84 }
85
86 struct segment {
87     int l, r;
88     data d;
89     segment(int _l, int _r) { // from _l to _r
90         l = _l, r = _r;

```

```

91         if (l <= r) d = query(l, r, 0);
92         else d = query(r, l, 1); //reverse
93     }
94 };
95
96 vector<segment> seg[2];
97
98 void get_data(int x, int y) {
99     seg[0].clear(); seg[1].clear();
100     int tx = Top[x], ty = Top[y];
101     int s = 0;
102     while (tx != ty) {
103         if (deep[tx] < deep[ty]) {
104             swap(tx, ty);
105             swap(x, y);
106             s ^= 1;
107         }
108         if (s == 0)
109             seg[s].push_back(segment(w[x], w[tx]));
110         else
111             seg[s].push_back(segment(w[tx], w[x]));
112         x = father[tx];
113         tx = Top[x];
114     }
115     if (x != y) {
116         if (deep[x] < deep[y]) {
117             swap(x, y);
118             s ^= 1;
119         }
120         if (s == 0)
121             seg[s].push_back(segment(w[x], w[y] + 1));
122         else
123             seg[s].push_back(segment(w[y] + 1, w[x]));
124     }
125     reverse(seg[1].begin(), seg[1].end());
126     for (int i = 0; i < seg[1].size(); ++i)
127         seg[0].push_back(seg[1][i]);
128     // saved to seg[0]
129 }
130
131 void init() {
132     top = s_top = 0;
133     for (int i = 1; i <= n; ++i) gh[i] = 0;
134 }

```

4.2 点分治

初始化时须设置 $top = 1$ 。

```

1 void addedge(int x, int y) {

```



```

2     edge[++top].adj = y;
3     edge[top].valid = 1;
4     edge[top].next = gh[x];
5     gh[x] = top;
6 }
7 void get_size(int x, int root=0) {
8     size[x] = 1; son[x] = 0;
9     int dd = 0;
10    for (int p=gh[x]; p; p=edge[p].next)
11        if (edge[p].adj != root && edge[p].valid) {
12            get_size(edge[p].adj, x);
13            size[x] += size[edge[p].adj];
14            if (size[edge[p].adj] > dd) {
15                dd = size[edge[p].adj];
16                son[x] = edge[p].adj;
17            }
18        }
19 }
20 int getroot(int x) {
21     get_size(x);
22     int sz = size[x];
23     while (size[son[x]] > sz/2)
24         x = son[x];
25     return x;
26 }
27 void dc(int x) {
28     x = getroot(x);
29     static int list[N], ltop;
30     ltop = 0;
31     for (int p=gh[x]; p; p=edge[p].next)
32         if (edge[p].valid)
33             list[++ltop] = edge[p].adj;
34     clear();
35     for (int i=1; i<=ltop; i++) {
36         update();
37         modify();
38     }
39     clear();
40     for (int i=ltop; i>=1; i--) {
41         update();
42         modify();
43     }
44     //be careful about the root
45     for (int p=gh[x]; p; p=edge[p].next)
46         if (edge[p].valid) {
47             edge[p].valid = 0;
48             edge[p^1].valid = 0;
49             dc(edge[p].adj);
50         }
51 }

```

4.3 Link Cut Tree

请注意，一开始必须调用 `lct.init(0)`，否则求出的最小值一定会是 0。

```
1 struct DTree {
2     int f[N], son[N][2], sz[N], rev[N], val[N], minid[N], minval[N];
3     int tot;
4     stack<int> s;
5     void init(int i) {
6         tot = max(tot, i);
7         son[i][0] = son[i][1] = 0;
8         f[i] = sz[i] = rev[i] = 0;
9         val[i] = minval[i] = inf;
10        minid[i] = i;
11    }
12    bool isroot(int x) {
13        return !f[x] || (son[f[x]][0] != x && son[f[x]][1] != x);
14    }
15    void revl(int x) {
16        if (!x) return;
17        swap(son[x][0], son[x][1]);
18        rev[x] ^= 1;
19    }
20    void down(int x) {
21        if (!x) return;
22        if (rev[x]) revl(son[x][0]), revl(son[x][1]), rev[x] = 0;
23    }
24    void up(int x) {
25        if (!x) return;
26        down(son[x][0]); down(son[x][1]);
27        sz[x] = sz[son[x][0]] + sz[son[x][1]] + 1;
28        minval[x] = val[x]; minid[x] = x;
29        if (minval[son[x][0]] < minval[x]) minval[x] = minval[son[x][0]], minid[x] =
            minid[son[x][0]];
30        if (minval[son[x][1]] < minval[x]) minval[x] = minval[son[x][1]], minid[x] =
            minid[son[x][1]];
31    }
32    void rotate(int x) {
33        int y = f[x], w = son[y][1] == x;
34        son[y][w] = son[x][w ^ 1];
35        if (son[x][w ^ 1]) f[son[x][w ^ 1]] = y;
36        if (f[y]) {
37            int z = f[y];
38            if (son[z][0] == y) son[z][0] = x;
39            else if (son[z][1] == y) son[z][1] = x;
40        }
41        f[x] = f[y]; f[y] = x; son[x][w ^ 1] = y;
42        up(y);
```

```

43     }
44     void splay(int x) {
45         while (!s.empty()) s.pop();
46         s.push(x);
47         for (int i = x; !isroot(i); i = f[i]) s.push(f[i]);
48         while (!s.empty()) down(s.top()), s.pop();
49         while (!isroot(x)) {
50             int y = f[x];
51             if (!isroot(y)) {
52                 if ((son[f[y]][0] == y) ^ (son[y][0] == x))
53                     rotate(x);
54                 else
55                     rotate(y);
56             }
57             rotate(x);
58         }
59         up(x);
60     }
61     void access(int x) {
62         for (int y = 0; x; y = x, x = f[x]) {
63             splay(x);
64             son[x][1] = y;
65             up(x);
66         }
67     }
68     int root(int x) {
69         access(x);
70         splay(x);
71         while (son[x][0]) x = son[x][0];
72         return x;
73     }
74     void makeroot(int x) {
75         access(x);
76         splay(x);
77         rev1(x);
78     }
79     void link(int x, int y) {
80         makeroot(x);
81         f[x] = y;
82         access(x);
83     }
84     void cutf(int x) { // 它和父亲的边
85         access(x);
86         splay(x);
87         f[son[x][0]] = 0;
88         son[x][0] = 0;
89         up(x);
90     }
91     void cut(int x, int y) { // 切断 x 与 y 之间的边 (须保证 x 与 y 相邻)
92         makeroot(x);

```

```

93         cutf(y);
94     }
95     int ask(int x, int y) { // 询问 x 到 y 之间取得最小值的点
96         makeroot(x);
97         access(y);
98         splay(y);
99         return minid[y];
100    }
101    int querymin_cut(int x, int y) { // 询问 x 到 y 之间取得最小值的点，并把它删去
        (须保证该点在 x 和 y 之间，且度数恰好为 2)
102        int m = ask(x, y);
103        makeroot(x);
104        cutf(m);
105        makeroot(y);
106        cutf(m);
107        return val[m];
108    }
109    void link(int x, int y, int w) { // 在 x 和 y 之间添加一条权值为 w 的边 (将边视
        为点插入)
110        init(++tot);
111        val[tot] = minval[tot] = w;
112        link(x, tot);
113        link(y, tot);
114    }
115 } lct;

```

4.4 求子树的直径

树形 DP。

答案保存在 u, d 数组中。

$u[x].exc$ 表示切断 x 与 $father[x]$ 的边， $father[x]$ 表示的那颗子树的直径。

$d[x].exc$ 表示切断 x 与 $father[x]$ 的边， x 表示的那颗子树的直径。

```

1  #include <bits/stdc++.h>
2
3  #define N 200020
4
5  using namespace std;
6
7  vector<int> g[N];
8  int n, q, top;
9  int deep[N], father[N], son[N], size[N], Top[N], dfn[N], rdfs[N];
10
11 void dfs(int x, int root = 0) {
12     deep[x] = deep[root] + 1;
13     father[x] = root;
14     son[x] = 0; size[x] = 1;
15     if (root) g[x].erase(lower_bound(g[x].begin(), g[x].end(), root));
16     // 去根

```

```

17     int dd = 0;
18     for (int i = 0; i < g[x].size(); ++i) {
19         dfs(g[x][i], x);
20         if (size[g[x][i]] > dd) {
21             dd = size[g[x][i]];
22             son[x] = g[x][i];
23         }
24         size[x] += size[g[x][i]];
25     }
26 }
27
28 void split(int x, int tp) {
29     dfn[x] = ++top; rdfs[top] = x; Top[x] = tp;
30     if (son[x]) split(son[x], tp);
31     for (int i = 0; i < g[x].size(); ++i)
32         if (g[x][i] != son[x])
33             split(g[x][i], g[x][i]);
34 }
35
36 struct data {
37     int inc, inc_id;
38     int exc, exc_l, exc_r;
39     //inc 表示从该点出发可以走到的最远距离
40     //inc_id 表示从该点出发可以走到的最远点的编号
41     //exc 表示子树中两点最远距离
42     //exc_l, exc_r 表示子树中两点取得最远距离的两点的编号
43     data() {
44         inc = inc_id = 0;
45         exc = exc_l = exc_r = 0;
46     }
47 } u[N], d[N];
48
49 int safe(int x, int y) {
50     // 防止 inc_id = 0 的情况
51     if (x) return x;
52     return y;
53 }
54
55 void dfs1(int x) {
56     d[x].inc = 1; d[x].inc_id = x;
57     data mx1 = data(), mx2 = data();
58     // mx1, mx2 表示儿子 inc 最大、第2大值，用于更新该点 exc
59     for (int i = 0; i < g[x].size(); ++i) {
60         dfs1(g[x][i]);
61         if (d[g[x][i]].inc + 1 > d[x].inc) {
62             d[x].inc = d[g[x][i]].inc + 1;
63             d[x].inc_id = d[g[x][i]].inc_id;
64         }
65         if (d[g[x][i]].inc > mx1.inc) {
66             mx2 = mx1;

```

```

67         mx1 = d[g[x][i]];
68     } else
69     {
70         if (d[g[x][i]].inc > mx2.inc) {
71             mx2 = d[g[x][i]];
72         }
73     }
74     d[x].exc = mx1.inc + mx2.inc + 1;
75     d[x].exc_l = safe(mx1.inc_id, x);
76     d[x].exc_r = safe(mx2.inc_id, x);
77     for (int i = 0; i < g[x].size(); ++i)
78     {
79         if (d[g[x][i]].exc > d[x].exc) {
80             d[x].exc = d[g[x][i]].exc;
81             d[x].exc_l = d[g[x][i]].exc_l;
82             d[x].exc_r = d[g[x][i]].exc_r;
83         }
84     }
85 }
86
87 void dfs2(int x, data y) {
88     u[x] = y;
89     if (!y.exc) y.exc = 1, y.exc_l = y.exc_r = x;
90     data mx1 = y, mx2 = data(), mx3 = data(), mxe1 = y, mxe2 = data();
91     // mx1, mx2, mx3 表示根过来的子树中 inc 的最大、第2大、第3大值
92     // mxe1, mxe2 表示根过来的子树中 exc 的最大、第2大值
93     int mx1_id = -1, mx2_id = -1, mx3_id = -1, mxe1_id = -1, mxe2_id = -1;
94     for (int i = 0; i < g[x].size(); ++i) {
95         if (d[g[x][i]].inc > mx1.inc) {
96             mx3 = mx2; mx3_id = mx2_id;
97             mx2 = mx1; mx2_id = mx1_id;
98             mx1 = d[g[x][i]]; mx1_id = i;
99         } else
100         if (d[g[x][i]].inc > mx2.inc) {
101             mx3 = mx2; mx3_id = mx2_id;
102             mx2 = d[g[x][i]]; mx2_id = i;
103         } else
104         if (d[g[x][i]].inc > mx3.inc) {
105             mx3 = d[g[x][i]]; mx3_id = i;
106         }
107         if (d[g[x][i]].exc > mxe1.exc) {
108             mxe2 = mxe1; mxe2_id = mxe1_id;
109             mxe1 = d[g[x][i]]; mxe1_id = i;
110         } else
111         if (d[g[x][i]].exc > mxe2.exc) {
112             mxe2 = d[g[x][i]]; mxe2_id = i;
113         }
114     }
115     for (int i = 0; i < g[x].size(); ++i) {
116         data z = data();
117         if (i == mx1_id) {
118             z.exc = mx2.inc + mx3.inc + 1;
119             z.exc_l = safe(mx2.inc_id, x);

```

```

117         z.exc_r = safe(mx3.inc_id, x);
118     } else
119     if (i == mx2_id) {
120         z.exc = mx1.inc + mx3.inc + 1;
121         z.exc_l = safe(mx1.inc_id, x);
122         z.exc_r = safe(mx3.inc_id, x);
123     } else {
124         z.exc = mx1.inc + mx2.inc + 1;
125         z.exc_l = safe(mx1.inc_id, x);
126         z.exc_r = safe(mx2.inc_id, x);
127     }
128     if (i == mx1_id) {
129         if (mxe2.exc > z.exc) z = mxe2;
130     } else {
131         if (mxe1.exc > z.exc) z = mxe1;
132     }
133     if (i == mx1_id) {
134         z.inc = mx2.inc + 1;
135         z.inc_id = safe(mx2.inc_id, x);
136     } else {
137         z.inc = mx1.inc + 1;
138         z.inc_id = safe(mx1.inc_id, x);
139     }
140     dfs2(g[x][i], z);
141 }
142 }

```

4.5 虚树

设 $a[0 \cdots k-1]$ 为需要构建虚树的点。

构建出虚树的节点保存在 a 数组中, k 为节点个数。加边调用函数 `addedge(int x, int y, int w)`。

```

1 bool cmp(int x, int y) {
2     return dfn[x] < dfn[y];
3 }
4
5 stack<int> stk;
6
7 void solve() {
8     sort(a, a + k, cmp);
9     int m = k;
10    for (int j = 1; j < m; ++j)
11        a[k++] = lca(a[j - 1], a[j]);
12    sort(a, a + k, cmp);
13    k = unique(a, a + k) - a;
14    stk.push(a[0]);
15    for (int j = 1; j < k; ++j) {
16        int u = lca(stk.top(), a[j]);
17        while (dep[stk.top()] > dep[u]) --top;
18        assert(stk.top() == u);

```

```

19     stk.push(a[j]);
20     addedge(u, a[j], dis[a[j]] - dis[u]);
21 }
22 }

```

5 图

5.1 欧拉回路

欧拉回路：

无向图：每个顶点的度数都是偶数，则存在欧拉回路。

有向图：每个顶点的入度 = 出度，则存在欧拉回路。

欧拉路径：

无向图：当且仅当该图所有顶点的度数为偶数，或者除了两个度数为奇数外其余的全是偶数。

有向图：当且仅当该图所有顶点出度 = 入度或者一个顶点出度 = 入度 + 1，另一个顶点入度 = 出度 + 1，其他顶点出度 = 入度。

下面 $O(n + m)$ 求欧拉回路的代码中， n 为点数， m 为边数，若有解则依次输出经过的边的编号，若是无向图，则正数表示 x 到 y ，负数表示 y 到 x 。

```

1 namespace UndirectedGraph{
2     int n,m,i,x,y,d[N],g[N],v[M<<1],w[M<<1],vis[M<<1],nxt[M<<1],ed;
3     int ans[M],cnt;
4     void add(int x,int y,int z){
5         d[x]++;
6         v[++ed]=y;w[ed]=z;nxt[ed]=g[x];g[x]=ed;
7     }
8     void dfs(int x){
9         for(int&i=g[x];i;){
10             if(vis[i]){i=nxt[i];continue;}
11             vis[i]=vis[i^1]=1;
12             int j=w[i];
13             dfs(v[i]);
14             ans[++cnt]=j;
15         }
16     }
17     void solve(){
18         scanf("%d%d",&n,&m);
19         for(i=ed=1;i<=m;i++)scanf("%d%d",&x,&y),add(x,y,i),add(y,x,-i);
20         for(i=1;i<=n;i++)if(d[i]&1){puts("NO");return;}
21         for(i=1;i<=n;i++)if(g[i]){dfs(i);break;}
22         for(i=1;i<=n;i++)if(g[i]){puts("NO");return;}
23         puts("YES");
24         for(i=m;i;i--)printf("%d_",ans[i]);
25     }
26 }
27 namespace DirectedGraph{
28     int n,m,i,x,y,d[N],g[N],v[M],vis[M],nxt[M],ed;
29     int ans[M],cnt;

```



```

30     void add(int x,int y){
31         d[x]++;d[y]--;
32         v[++ed]=y;nxt[ed]=g[x];g[x]=ed;
33     }
34     void dfs(int x){
35         for(int&i=g[x];i;){
36             if(vis[i]){i=nxt[i];continue;}
37             vis[i]=1;
38             int j=i;
39             dfs(v[i]);
40             ans[++cnt]=j;
41         }
42     }
43     void solve(){
44         scanf("%d%d",&n,&m);
45         for(i=1;i<=m;i++)scanf("%d%d",&x,&y),add(x,y);
46         for(i=1;i<=n;i++)if(d[i]){puts("NO");return;}
47         for(i=1;i<=n;i++)if(g[i]){dfs(i);break;}
48         for(i=1;i<=n;i++)if(g[i]){puts("NO");return;}
49         puts("YES");
50         for(i=m;i;i--)printf("%d_",ans[i]);
51     }
52 }

```

5.2 最短路径

5.2.1 Dijkstra

```

1  #define LL long long
2
3  struct EDGE {
4      int adj, w, next;
5  } edge[M*2];
6
7  typedef pair<LL, int> pli;
8  priority_queue <pli, vector<pli>, greater<pli> > q;
9
10 int n, top, gh[N];
11 LL dist[N];
12
13 void addedge(int x, int y, int w) {
14     edge[++top].adj = y;
15     edge[top].w = w;
16     edge[top].next = gh[x];
17     gh[x] = top;
18 }
19
20 LL dijkstra(int s, int t) {
21     memset(dist, 63, sizeof(dist));

```

```

22     memset(v, 0, sizeof(v));
23     dist[s] = 0;
24     q.push(make_pair(dist[s], s));
25     while (!q.empty()) {
26         LL dis = q.top().first;
27         int x = q.top().second;
28         q.pop();
29         if (dis != dist[x]) continue;
30         for (int p=gh[x]; p; p=edge[p].next) {
31             if (dis + edge[p].w < dist[edge[p].adj]) {
32                 dist[edge[p].adj] = dis + edge[p].w;
33                 q.push(make_pair(dist[edge[p].adj], edge[p].adj));
34             }
35         }
36     }
37     return dist[t];
38 }

```

5.2.2 SPFA

```

1  struct EDGE {
2      int adj, w, next;
3  } edge[M*2];
4
5  int n,m,top,gh[N],v[N],cnt[N],q[N],dist[N],head,tail;
6
7  void addedge(int x, int y, int w) {
8      edge[++top].adj = y;
9      edge[top].w = w;
10     edge[top].next = gh[x];
11     gh[x] = top;
12 }
13
14 int spfa(int S, int T) {
15     memset(v, 0, sizeof(v));
16     memset(cnt, 0, sizeof(cnt));
17     memset(dist, 63, sizeof(dist));
18     head = 0, tail = 1;
19     dist[S] = 0; q[1] = S;
20     while (head != tail) {
21         (head += 1) %= N;
22         int x = q[head]; v[x] = 0;
23         ++cnt[x]; if (cnt[x] > n) return -1;
24         for (int p=gh[x]; p; p=edge[p].next)
25             if (dist[x] + edge[p].w < dist[edge[p].adj]) {
26                 dist[edge[p].adj] = dist[x] + edge[p].w;
27                 if (!v[edge[p].adj]) {
28                     v[edge[p].adj] = 1;
29                     (tail += 1) %= N;

```

```

30         q[tail] = edge[p].adj;
31     }
32 }
33 }
34 return dist[T];
35 }

```

5.3 K 短路

接口：

kthsp::init(n)：初始化并设置节点个数为 n

kthsp::add(x, y, w)：添加一条 x 到 y 的有向边

kthsp::work(S, T, k)：求 S 到 T 的第 k 短路

```

1  #define N 200020
2  #define M 400020
3  #define LOGM 20
4  #define LL long long
5  #define inf (1LL<<61)
6
7  namespace pheap {
8      struct Node {
9          int next, son[2];
10         LL val;
11     } node[M*LOGM];
12     int LOG[M];
13     int root[M], size[M*LOGM], top;
14     int add() {
15         ++top; assert(top < M*LOGM);
16         node[top].next = node[top].son[0] = node[top].son[1] = 0;
17         node[top].val = inf;
18         return top;
19     }
20     int copy(int x) {
21         int t = add();
22         node[t] = node[x];
23         return t;
24     }
25     void init() {
26         memset(root, 0, sizeof(root));
27         top = -1; add();
28         LOG[1] = 0;
29         for (int i=2;i<M;i++) LOG[i] = LOG[i>>1] + 1;
30     }
31     void upd(int x, int &next, LL &val) {
32         if (val < node[x].val) {
33             swap(val, node[x].val);
34             swap(next, node[x].next);
35         }

```

```

36     }
37     void insert(int x, int next, LL val) {
38         int sz = size[root[x]] + 1;
39         root[x] = copy(root[x]);
40         size[root[x]] = sz; x = root[x];
41         upd(x, next, val);
42         for (int i=LOG[sz]-1;i>=0;i--) {
43             int ind = (sz>>i)&1;
44             node[x].son[ind] = copy(node[x].son[ind]);
45             x = node[x].son[ind];
46             upd(x, next, val);
47         }
48     }
49 };
50
51 namespace kthsp {
52     using namespace pheap;
53     struct EDGE {
54         int adj, w, next;
55     } edge[2][M];
56     struct W {
57         int x, y, w;
58     } e[M];
59     bool has_init = 0;
60     int n, m, top[2], gh[2][N], v[N];
61     LL dist[N];
62     void init(int n1) {
63         has_init = 1;
64         n = n1; m = 0;
65         memset(top, 0, sizeof(top));
66         memset(gh, 0, sizeof(gh));
67         for (int i=1;i<=n;i++) dist[i] = inf;
68     }
69     void addedge(int id, int x, int y, int w) {
70         edge[id][++top[id]].adj = y;
71         edge[id][top[id]].w = w;
72         edge[id][top[id]].next = gh[id][x];
73         gh[id][x] = top[id];
74     }
75     void add(int x, int y, int w) {
76         assert(has_init);
77         e[++m].x=x; e[m].y=y; e[m].w=w;
78     }
79     int best[N], bestw[N];
80     typedef pair<LL, int> pli;
81     priority_queue <pli, vector<pli>, greater<pli> > q;
82
83     // you can replace dijkstra with SPFA or TOPSORT(DAG)
84     void dijkstra(int S) {
85         while (!q.empty()) q.pop();

```

```

86     dist[S] = 0; q.push(make_pair(dist[S], S));
87     while (!q.empty()) {
88         LL dis = q.top().first;
89         int x = q.top().second;
90         q.pop();
91         if (dist[x] != dis) continue;
92         for (int p=gh[1][x]; p; p=edge[1][p].next) {
93             int y = edge[1][p].adj;
94             if (dist[x] + edge[1][p].w < dist[y]) {
95                 dist[y] = dist[x] + edge[1][p].w;
96                 best[y] = x;
97                 bestw[y] = p;
98                 q.push(make_pair(dist[y], y));
99             }
100         }
101     }
102 }
103 void dfs(int x) {
104     if (v[x]) return;
105     v[x] = 1;
106     if (best[x]) root[x] = root[best[x]];
107     for (int p=gh[0][x]; p; p=edge[0][p].next)
108         if (dist[edge[0][p].adj] != inf && bestw[x] != p) {
109             insert(x, edge[0][p].adj, edge[0][p].w + dist[edge[0][p].adj] - dist
110                 [x]);
111         }
112     for (int p=gh[1][x]; p; p=edge[1][p].next)
113         if (best[edge[1][p].adj] == x)
114             dfs(edge[1][p].adj);
115 }
116 LL work(int S, int T, int k) {
117     assert(has_init);
118     n++; add(T, n, 0);
119     if (S == T) k++;
120     T = n;
121     for (int i=1; i<=m; i++) {
122         addedge(0, e[i].x, e[i].y, e[i].w);
123         addedge(1, e[i].y, e[i].x, e[i].w);
124     }
125     dijkstra(T);
126     root[T] = 0; pheap::init();
127     memset(v, 0, sizeof(v));
128     dfs(T);
129     while (!q.empty()) q.pop();
130     if (k == 1) return dist[S];
131     if (root[S]) q.push(make_pair(dist[S] + node[root[S]].val, root[S]));
132     while (k--) {
133         if (q.empty()) return inf;
134         pli now = q.top(); q.pop();
135         if (k == 1) return now.first;

```

```

135         int x = node[now.second].next, u = node[now.second].son[0], v = node[now
            .second].son[1];
136         if (root[x]) q.push(make_pair(now.first + node[root[x]].val, root[x]));
137         if (u) q.push(make_pair(now.first - node[now.second].val + node[u].val,
            u));
138         if (v) q.push(make_pair(now.first - node[now.second].val + node[v].val,
            v));
139     }
140     return 0;
141 }
142 };

```

5.4 Tarjan

割点的判断：一个顶点 u 是割点，当且仅当满足 (1) 或 (2)：

(1) u 为树根，且 u 有多于一个子树（即：存在一个儿子 v 使得 $dfn[u] + 1 \neq dfn[v]$ ）

(2) u 不为树根，且满足存在 (u, v) 为树枝边（ u 为 v 的父亲），使得 $dfn[u] \leq low[v]$

桥的判断：一条无向边 (u, v) 是桥，当且仅当 (u, v) 为树枝边，满足 $dfn[u] < low[v]$

```

1 struct EDGE { int adj, next; } edge[M];
2 int n, m, top, gh[N];
3 int dfn[N], low[N], cnt, ind, stop, instack[N], stack[N], belong[N];
4 void addedge(int x, int y) {
5     edge[++top].adj = y;
6     edge[top].next = gh[x];
7     gh[x] = top;
8 }
9 void tarjan(int x) {
10     dfn[x] = low[x] = ++ind;
11     instack[x] = 1; stack[++stop] = x;
12     for (int p=gh[x]; p; p=edge[p].next)
13         if (!dfn[edge[p].adj]) {
14             tarjan(edge[p].adj);
15             low[x] = min(low[x], low[edge[p].adj]);
16         } else if (instack[edge[p].adj]) {
17             low[x] = min(low[x], dfn[edge[p].adj]);
18         }
19     if (dfn[x] == low[x]) {
20         ++cnt; int tmp=0;
21         while (tmp!=x) {
22             tmp = stack[stop--];
23             belong[tmp] = cnt;
24             instack[tmp] = 0;
25         }
26     }
27 }

```

5.5 2-SAT

```

1  #define N number_of_vertex
2  #define M number_of_edges
3
4  struct MergePoint {
5      struct EDGE {
6          int adj, next;
7      } edge[M];
8      int ex[M], ey[M];
9      bool instack[N];
10     int gh[N], top, dfn[N], low[N], cnt, ind, stop, stack[N], belong[N];
11     void init() {
12         cnt = ind = stop = top = 0;
13         memset(dfn, 0, sizeof(dfn));
14         memset(instack, 0, sizeof(instack));
15         memset(gh, 0, sizeof(gh));
16     }
17     void addedge(int x, int y) { //reverse
18         std::swap(x, y);
19         edge[++top].adj = y;
20         edge[top].next = gh[x];
21         gh[x] = top;
22         ex[top] = x;
23         ey[top] = y;
24     }
25     void tarjan(int x) {
26         dfn[x] = low[x] = ++ind;
27         instack[x] = 1; stack[++stop] = x;
28         for (int p=gh[x]; p; p=edge[p].next)
29             if (!dfn[edge[p].adj]) {
30                 tarjan(edge[p].adj);
31                 low[x] = std::min(low[x], low[edge[p].adj]);
32             } else if (instack[edge[p].adj]) {
33                 low[x] = std::min(low[x], dfn[edge[p].adj]);
34             }
35         if (dfn[x] == low[x]) {
36             ++cnt; int tmp = 0;
37             while (tmp!=x) {
38                 tmp = stack[stop--];
39                 belong[tmp] = cnt;
40                 instack[tmp] = 0;
41             }
42         }
43     }
44     void work() {
45         for (int i = (__first__); i <= (__last__); ++i)
46             if (!dfn[i])
47                 tarjan(i);
48     }
49 } merge;

```

```

50
51 struct Topsort {
52     struct EDGE {
53         int adj, next;
54     } edge[M];
55     int n, top, gh[N], ops[N], deg[N], ans[N];
56     std::queue<int> q;
57     void init() {
58         n = merge.cnt; top = 0;
59         memset(gh, 0, sizeof(gh));
60         memset(deg, 0, sizeof(deg));
61     }
62     void addedge(int x, int y) {
63         if (x == y) return;
64         edge[++top].adj = y;
65         edge[top].next = gh[x];
66         gh[x] = top;
67         ++deg[y];
68     }
69     void work() {
70         for (int i = 1; i <= n; ++i)
71             if (!deg[i])
72                 q.push(i);
73         while (!q.empty()) {
74             int x = q.front();
75             q.pop();
76             for (int p = gh[x]; p; p = edge[p].next)
77                 if (--deg[edge[p].adj])
78                     q.push(edge[p].adj);
79             if (ans[x]) continue;
80             ans[x] = -1; //not selected
81             ans[ops[x]] = 1; //selected
82         }
83     }
84 } ts;

```

调用示例:

```

1  merge.init();
2  merge.addedge();
3  merge.work();
4  for (int i = 1; i <= n; ++i) {
5      if (merge.belong[U(i, 0)] == merge.belong[U(i, 1)]) {
6          puts("NO");
7          return 0;
8      }
9      ts.ops[merge.belong[U(i, 0)]] = merge.belong[U(i, 1)];
10     ts.ops[merge.belong[U(i, 1)]] = merge.belong[U(i, 0)];
11 }
12 ts.init();
13 ts.work();

```



```

14     puts("YES");
15     for (int i = 1; i <= n; ++i) {
16         int x = U(i, 0), y = U(i, 1);
17         x = merge.belong[x], y = merge.belong[y];
18         x = ts.ans[x], y = ts.ans[y];
19         if (x == 1) puts("0_is_selected");
20         if (y == 1) puts("1_is_selected");
21     }

```

5.6 统治者树 (Dominator Tree)

Dominator Tree 可以解决判断一类有向图必经点的问题。

$idom[x]$ 表示离 x 最近的必经点 (重编号后)。将 $idom[x]$ 作为 x 的父亲, 构成一棵 Dominator Tree

接口:

`void dominator::init(int n);` 初始化, 有向图节点数为 n

`void dominator::addedge(int u, int v);` 添加一条有向边 (u, v)

`void dominator::work(int root);` 以 $root$ 为根, 建立一棵 Dominator Tree

结果的返回:

在执行 `dominator::work(int root);` 后, 树边保存在 `vector <int> tree[N]` 中

```

1 namespace dominator {
2     vector <int> g[N], rg[N], bucket[N], tree[N];
3     int n, ind, idom[N], sdom[N], dfn[N], dsu[N], father[N], label[N], rev[N];
4     void dfs(int x) {
5         ++ind;
6         dfn[x] = ind; rev[ind] = x;
7         label[ind] = dsu[ind] = sdom[ind] = ind;
8         for (auto p : g[x]) {
9             if (!dfn[p]) dfs(p), father[dfn[p]] = dfn[x];
10            rg[dfn[p]].push_back(dfn[x]);
11        }
12    }
13    void init(int n1) {
14        n = n1; ind = 0;
15        for (int i = 1; i <= n; ++i) {
16            g[i].clear();
17            rg[i].clear();
18            bucket[i].clear();
19            tree[i].clear();
20            dfn[i] = 0;
21        }
22    }
23    void addedge(int u, int v) {
24        g[u].push_back(v);
25    }
26    int find(int x, int step=0) {
27        if (dsu[x] == x) return step ? -1 : x;

```

```

28     int y = find(dsu[x], 1);
29     if (y < 0) return x;
30     if (sdom[label[dsu[x]]] < sdom[label[x]])
31         label[x] = label[dsu[x]];
32     dsu[x] = y;
33     return step ? dsu[x] : label[x];
34 }
35 void work(int root) {
36     dfs(root); n = ind;
37     for (int i = n; i; --i) {
38         for (auto p : rg[i])
39             sdom[i] = min(sdom[i], sdom[find(p)]);
40         if (i > 1) bucket[sdom[i]].push_back(i);
41         for (auto p : bucket[i]) {
42             int u = find(p);
43             if (sdom[p] == sdom[u]) idom[p] = sdom[p];
44             else idom[p] = u;
45         }
46         if (i > 1) dsu[i] = father[i];
47     }
48     for (int i = 2; i <= n; ++i) {
49         if (idom[i] != sdom[i])
50             idom[i] = idom[idom[i]];
51         tree[rev[i]].push_back(rev[idom[i]]);
52         tree[rev[idom[i]]].push_back(rev[i]);
53     }
54 }
55 };

```

5.7 网络流

5.7.1 最大流

注意: *top* 要初始化为 1

```

1 struct EDGE { int adj, w, next; } edge[M];
2 int n, top, gh[N], nrl[N];
3 void addedge(int x, int y, int w) {
4     edge[++top].adj = y;
5     edge[top].w = w;
6     edge[top].next = gh[x];
7     gh[x] = top;
8     edge[++top].adj = x;
9     edge[top].w = 0;
10    edge[top].next = gh[y];
11    gh[y] = top;
12 }
13 int dist[N], q[N];
14 int bfs() {
15     memset(dist, 0, sizeof(dist));

```

```

16     q[1] = S; int head = 0, tail = 1; dist[S] = 1;
17     while (head != tail) {
18         int x = q[++head];
19         for (int p=gh[x]; p; p=edge[p].next)
20             if (edge[p].w && !dist[edge[p].adj]) {
21                 dist[edge[p].adj] = dist[x] + 1;
22                 q[++tail] = edge[p].adj;
23             }
24     }
25     return dist[T];
26 }
27 int dinic(int x, int delta) {
28     if (x==T) return delta;
29     for (int& p=nrl[x]; p && delta; p=edge[p].next)
30         if (edge[p].w && dist[x]+1 == dist[edge[p].adj]) {
31             int dd = dinic(edge[p].adj, min(delta, edge[p].w));
32             if (!dd) continue;
33             edge[p].w -= dd;
34             edge[p^1].w += dd;
35             return dd;
36         }
37     return 0;
38 }
39 int work() {
40     int ans = 0;
41     while (bfs()) {
42         memcpy(nrl, gh, sizeof(gh));
43         int t; while (t = dinic(S, inf)) ans += t;
44     }
45     return ans;
46 }

```

5.7.2 上下界有源汇网络流

T 向 S 连容量为正无穷的边，将有源汇转化为无源汇。

每条边容量减去下界，设 $in[i]$ 表示流入 i 的下界之和减去流出 i 的下界之和。

新建超级源汇 SS, TT ，对于 $in[i] > 0$ 的点， SS 向 i 连容量为 $in[i]$ 的边。对于 $in[i] < 0$ 的点， i 向 TT 连容量为 $-in[i]$ 的边。

求出以 SS, TT 为源汇的最大流，如果等于 $\sum in[i] (in[i] > 0)$ ，则存在可行流。再求出 S, T 为源汇的最大流即为最大流。

费用流：建完图后等价于求以 SS, TT 为源汇的费用流。

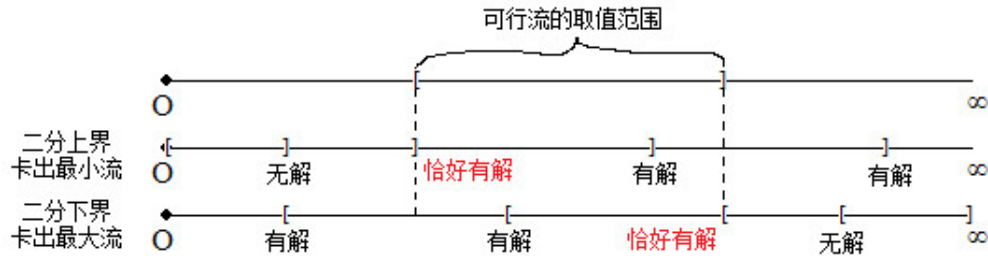
5.7.3 上下界无源汇网络流

1. 怎样求无源汇有上下界网络的可行流？

由于有源汇的网络我们先要转化成无源汇，所以本来就无源汇的网络不用再作特殊处理。

2. 怎样求无源汇有上下界网络的最大流、最小流？

一种简易的方法是采用二分思想，不断通过可行流的存在与否对 (t, s) 边的上下界 U, L 进行调整。求最大流时令 $U = \infty$ 并二分 L ；求最小流时令 $L = 0$ 并二分 U 。道理很简单，因为可行流的取值范围是一段连续的区间，我们只要通过二分找到有解和无解的分界线即可。



5.7.4 费用流

注意：top 要初始化为 1

```

1 #define inf 0x3f3f3f3f
2 struct NetWorkFlow {
3     struct EDGE {
4         int adj, w, cost, next;
5     } edge[M*2];
6     int gh[N], q[N], dist[N], v[N], pre[N], prev[N], top;
7     int S, T;
8     void addedge(int x, int y, int w, int cost) {
9         edge[++top].adj = y;
10        edge[top].w = w;
11        edge[top].cost = cost;
12        edge[top].next = gh[x];
13        gh[x] = top;
14        edge[++top].adj = x;
15        edge[top].w = 0;
16        edge[top].cost = -cost;
17        edge[top].next = gh[y];
18        gh[y] = top;
19    }
20    void clear() {
21        top = 1;
22        memset(gh, 0, sizeof(gh));
23    }
24    int spfa() {
25        memset(dist, 63, sizeof(dist));
26        memset(v, 0, sizeof(v));
27        int head = 0, tail = 1;
28        q[1] = S; v[S] = 1; dist[S] = 0;
29        while (head != tail) {
30            (head += 1) %= N;
31            int x = q[head];

```

```

32         v[x] = 0;
33         for (int p=gh[x]; p; p=edge[p].next)
34             if (edge[p].w && dist[x] + edge[p].cost < dist[edge[p].adj]) {
35                 dist[edge[p].adj] = dist[x] + edge[p].cost;
36                 pre[edge[p].adj] = x;
37                 prev[edge[p].adj] = p;
38                 if (!v[edge[p].adj]) {
39                     v[edge[p].adj] = 1;
40                     (tail += 1) %= N;
41                     q[tail] = edge[p].adj;
42                 }
43             }
44     }
45     return dist[T] != inf;
46 }
47 int work() {
48     int ans = 0;
49     while (spfa()) {
50         int mx = inf;
51         for (int x=T; x!=S; x=pre[x])
52             mx = min(edge[prev[x]].w, mx);
53         ans += dist[T] * mx;
54         for (int x=T; x!=S; x=pre[x]) {
55             edge[prev[x]].w -= mx;
56             edge[prev[x]^1].w += mx;
57         }
58     }
59     return ans;
60 }
61 } nwf;

```

5.7.5 zkw 费用流

注意: *top* 要初始化为 1, 不得用于有负权的图

```

1 #define inf 0x3f3f3f3f //modify if you use long long or double
2 template <class _tp>
3 struct NetWorkFlow {
4     struct EDGE {
5         int adj, next;
6         _tp w, cost;
7     } edge[M*2];
8     int gh[N], top;
9     int S, T;
10    void addedge(int x, int y, _tp w, _tp cost) {
11        edge[++top].adj = y;
12        edge[top].w = w;
13        edge[top].cost = cost;
14        edge[top].next = gh[x];
15        gh[x] = top;

```

```

16     edge[++top].adj = x;
17     edge[top].w = 0;
18     edge[top].cost = -cost;
19     edge[top].next = gh[y];
20     gh[y] = top;
21 }
22 void clear() {
23     top = 1;
24     memset(gh, 0, sizeof(gh));
25 }
26 int v[N];
27 _tp cost, d[N], slk[N];
28 _tp aug(int x, _tp f) {
29     _tp left = f;
30     if (x == T) {
31         cost += f * d[S];
32         return f;
33     }
34     v[x] = true;
35     for (int p=gh[x]; p; p=edge[p].next)
36         if (edge[p].w && !v[edge[p].adj]) {
37             _tp t = d[edge[p].adj] + edge[p].cost - d[x];
38             if (t == 0) {
39                 _tp delt = aug(edge[p].adj, min(left, edge[p].w));
40                 if (delt > 0) {
41                     edge[p].w -= delt;
42                     edge[p^1].w += delt;
43                     left -= delt;
44                 }
45                 if (left == 0) return f;
46             } else {
47                 if (t < slk[edge[p].adj])
48                     slk[edge[p].adj] = t;
49             }
50         }
51     return f-left;
52 }
53 bool modlabel() {
54     _tp delt = inf;
55     for (int i=1; i<=T; i++)
56         if (!v[i]) {
57             if (slk[i] < delt) delt = slk[i];
58             slk[i] = inf;
59         }
60     if (delt == inf) return true;
61     for (int i=1; i<=T; i++)
62         if (v[i]) d[i] += delt;
63     return false;
64 }
65 _tp work() {

```

```

66     cost = 0;
67     memset(d, 0, sizeof(d));
68     memset(slk, 63, sizeof(slk));
69     do {
70         do {
71             memset(v, 0, sizeof(v));
72             } while (aug(S, inf));
73         } while (!modlabel());
74     return cost;
75     }
76 };
77 NetWorkFlow<int> nwf;

```

6 数学

6.1 扩展欧几里得解同余方程

ans[] 保存的是循环节内所有的解

```

1  int exgcd(int a,int b,int&x,int&y){
2      if(!b) return x=1,y=0,a;
3      int d=exgcd(b,a%b,x,y),t=x;
4      return x=y,y=t-a/b*y,d;
5  }
6  void cal(ll a,ll b,ll n){ //ax=b(mod n)
7      ll x,y,d=exgcd(a,n,x,y);
8      if(b%d) return;
9      x=(x%n+n)%n;
10     ans[cnt=1]=x*(b/d)%(n/d);
11     for(ll i=1;i<d;i++) ans[++cnt]=(ans[1]+i*n/d)%n;
12 }

```

6.2 同余方程组

```

1  int n,flag,k,m,a,r,d,x,y;
2  int main(){
3      scanf("%d",&n);
4      flag=k=1,m=0;
5      while(n--){
6          scanf("%d%d",&a,&r); //ans%a=r
7          if(flag){
8              d=exgcd(k,a,x,y);
9              if((r-m)%d){flag=0;continue;}
10             x=(x*((r-m)/d)+a/d)%(a/d),y=k/d*a,m=((x*k+m)%y)%y;
11             if(m<0)m+=y;
12             k=y;
13         }
14     }

```

```

15     printf("%d", flag?m:-1); //若 flag=1, 说明有解, 解为 ki+m, i 为任意整数
16 }

```

6.3 类欧几里得算法

类欧几里得模板有三种形式：

$$f(a, b, c, n) = \sum_{i=0}^n \left\lfloor \frac{ai+b}{c} \right\rfloor$$

$$g(a, b, c, n) = \sum_{i=0}^n i \left\lfloor \frac{ai+b}{c} \right\rfloor$$

$$h(a, b, c, n) = \sum_{i=0}^n \left\lfloor \frac{ai+b}{c} \right\rfloor^2$$

```

1  #define LL long long
2
3  const int P = 1000000007;
4  int Inv(int x) {
5      return x == 1 ? 1 : 1ll * (P - P / x) * Inv(P % x) % P;
6  }
7  const int i2 = Inv(2);
8  const int i6 = Inv(6);
9
10 struct ifo {
11     int f, g, h;
12     ifo(int f, int g, int h) : f(f), g(g), h(h) {}
13 };
14
15 int S1(int n) {
16     return 1ll * n * (n + 1) % P * i2 % P;
17 }
18
19 int S2(int n) {
20     return 1ll * n * (n + 1) % P * (2 * n + 1) % P * i6 % P;
21 }
22
23 ifo Get(int n, int A, int B, int C) {
24     if (!A) {
25         int t = B / C;
26         int f = 1ll * (n + 1) * t % P;
27         int g = 1ll * S1(n) * t % P;
28         int h = 1ll * (n + 1) * t % P * t % P;
29         return ifo(f, g, h);
30     } else if (A >= C || B >= C) {
31         ifo Nx = Get(n, A % C, B % C, C);
32         int p = A / C, q = B / C;
33         int f = (1ll * p * S1(n) + 1ll * q * (n + 1) + Nx.f) % P;

```



```

34     int g = (1ll * p * S2(n) + 1ll * q * S1(n) + Nx.g) % P;
35     int h = (1ll * p * p % P * S2(n) + 2ll * p * q % P * S1(n) + 1ll * (n + 1) *
              q % P * q + 2ll * p * Nx.g % P + 2ll * q * Nx.f % P + Nx.h) % P;
36     return ifo(f, g, h);
37 } else {
38     int m = (1ll * A * n + B) / C;
39     ifo Nx = Get(m - 1, C, C - B - 1, A);
40     int f = (1ll * n * m - Nx.f) % P;
41     int g = (1ll * m * S1(n) - 1ll * i2 * Nx.h - 1ll * i2 * Nx.f) % P;
42     int h = (2ll * n * S1(m - 1) % P + 1ll * n * m - 2ll * Nx.g - Nx.f) % P;
43     return ifo(f, g, h);
44 }
45 }

```

6.4 卡特兰数

$$h_1 = 1, h_n = \frac{h_{n-1}(4n-2)}{n+1} = \frac{C(2n,n)}{n+1} = C(2n,n) - C(2n,n-1)$$

在一个格点阵列中, 从 $(0,0)$ 点走到 (n,m) 点且不经对角线 $x=y$ 的方案数 $(x > y)$:

$$C(n+m-1, m) - C(n+m-1, m-1)$$

在一个格点阵列中, 从 $(0,0)$ 点走到 (n,m) 点且不穿过对角线 $x=y$ 的方案数 $(x \geq y)$:

$$C(n+m, m) - C(n+m, m-1)$$

6.5 斯特林数

6.5.1 第一类斯特林数

第一类 Stirling 数 $S(p, k)$ 的一个组合学解释是: 将 p 个物体排成 k 个非空循环排列的方法数。

$S(p, k)$ 的递推公式: $S(p, k) = (p-1)S(p-1, k) + S(p-1, k-1), 1 \leq k \leq p-1$

边界条件: $S(p, 0) = 0, p \geq 1, S(p, p) = 1, p \geq 0$

6.5.2 第二类斯特林数

第二类 Stirling 数 $S(p, k)$ 的一个组合学解释是: 将 p 个物体划分成 k 个非空的不可辨别 (可以理解为盒子没有编号) 集合的方法数。

$S(p, k)$ 的递推公式: $S(p, k) = kS(p-1, k) + S(p-1, k-1), 1 \leq k \leq p-1$

边界条件: $S(p, 0) = 0, p \geq 1, S(p, p) = 1, p \geq 0$

也有卷积形式:

$$S(n, m) = \frac{1}{m!} \sum_{k=0}^m (-1)^k C(m, k) (m-k)^n = \sum_{k=0}^m \frac{(-1)^k (m-k)^n}{k! (m-k)!} = \sum_{k=0}^m \frac{(-1)^k}{k!} \times \frac{(m-k)^n}{(m-k)!}$$

6.6 错排公式

$$D_1 = 0, D_2 = 1, D_n = (n-1)(D_{n-2} + D_{n-1})$$

6.7 Lucas 定理

接口:

初始化: `void lucas::init();`

计算 $C(n, m) \% \text{mod}$ 的值: `LL lucas::Lucas(LL n, LL m);`

```
1 #define mod 110119
2 #define LL long long
3 namespace lucas {
4     LL fac[mod+1], facv[mod+1];
5     LL power(LL base, LL times) {
6         LL ans = 1;
7         while (times) {
8             if (times&1) (ans *= base) %= mod;
9             (base *= base) %= mod;
10            times >>= 1;
11        }
12        return ans;
13    }
14    void init() {
15        fac[0] = 1; for (int i=1; i<mod; i++) fac[i] = (fac[i-1] * i) % mod;
16        facv[mod-1] = power(fac[mod-1], mod-2);
17        for (int i=mod-2; i>=0; --i) facv[i] = (facv[i+1] * (i+1)) % mod;
18    }
19    LL C(unsigned LL n, unsigned LL m) {
20        if (n < m) return 0;
21        return (fac[n] * facv[m] % mod * facv[n-m] % mod) % mod;
22    }
23    LL Lucas(unsigned LL n, unsigned LL m)
24    {
25        if (m == 0) return 1;
26        return (C(n%mod, m%mod) * Lucas(n/mod, m/mod)) % mod;
27    }
28 };
```

6.8 线性规划

6.8.1 单纯形法

单纯形法用于解决线性规划问题:

$$\begin{aligned} \max_{x_1, x_2, \dots, x_n} \quad & x_0 = c_1 x_1 + c_2 x_2 + \dots + c_n x_n \\ \text{s.t.} \quad & \begin{cases} A_{i1}x_1 + A_{i2}x_2 + \dots + A_{in}x_n \leq b_i, & i = 1, 2, \dots, m \\ x_j \geq 0, & j = 1, 2, \dots, n \end{cases} \end{aligned}$$

小心: 单纯形法通常能解决 $n \leq 500, m \leq 500$ 的数据规模的问题。若规模过大, 可能导致精度爆炸。

小心：单纯形法只能解决一般线性规划问题，不能解决整数规划问题（NP Hard）。若要用单纯形法解决整数规划问题，必须先证明一般线性规划的解不比整数规划好。

若 $b_i \geq 0, i = 1, 2, \dots, n$ ，则不需要执行 init，因为至少有一组解 $x_1 = x_2 = \dots = x_n = 0$ 。

输入格式 .

| | | | | | |
|----------|----------|----------|----------|----------|----------|
| n | m | | | | |
| c_1 | c_2 | c_3 | \dots | c_n | |
| A_{11} | A_{12} | A_{13} | \dots | A_{1n} | b_1 |
| A_{21} | A_{22} | A_{23} | \dots | A_{2n} | b_2 |
| \vdots | \vdots | \vdots | \ddots | \vdots | \vdots |
| A_{m1} | A_{m2} | A_{m3} | \dots | A_{mn} | b_m |

输出格式 .

若无解，输出 Infeasible。

若 x_0 无界，输出 Unbounded。

第一行输出答案 x_0 。

接下来一行输出 n 个实数表示 x_1, x_2, \dots, x_n 。

```

1 #include <bits/stdc++.h>
2
3 #define N 25
4 #define M 25
5
6 using namespace std;
7
8 const double eps = 1e-8, INF = 1e15;
9
10 int n, m;
11 double a[M][N], ans[N + M];
12 int id[N + M];
13
14 void pivot(int l, int e) {
15     swap(id[n + 1], id[e]);
16     double t = a[l][e];
17     a[l][e] = 1;
18     for (int j = 0; j <= n; ++j) a[l][j] /= t;
19     for (int i = 0; i <= m; ++i)
20         if (i != l && abs(a[i][e]) > eps) {
21             t = a[i][e];
22             a[i][e] = 0;
23             for (int j = 0; j <= n; ++j) a[i][j] -= a[l][j] * t;
24         }
25 }
26
27 bool init() {
28     while (1) {
29         int e = 0, l = 0;

```

```

30     for (int i = 1; i <= m; ++i)
31         if (a[i][0] < -eps && (!l || (rand() & 1)))
32             l = i;
33     if (!l) break;
34     for (int j = 1; j <= n; ++j)
35         if (a[l][j] < -eps && (!e || (rand() & 1)))
36             e = j;
37     if (!e) return false; // Infeasible
38     pivot(l, e);
39 }
40 return true;
41 }
42
43 bool simplex() {
44     while (1) {
45         int l = 0, e = 0;
46         double mn = INF;
47         for (int j = 1; j <= n; ++j)
48             if (a[0][j] > eps) {
49                 e = j;
50                 break;
51             }
52         if (!e) break;
53         for (int i = 1; i <= m; ++i)
54             if (a[i][e] > eps && a[i][0] / a[i][e] < mn) {
55                 mn = a[i][0] / a[i][e];
56                 l = i;
57             }
58         if (!l) return false; // Unbounded
59         pivot(l, e);
60     }
61     return true;
62 }
63
64 int main() {
65     scanf("%d%d", &n, &m);
66     for (int i = 1; i <= n; ++i) scanf("%lf", &a[0][i]);
67     for (int i = 1; i <= m; ++i) {
68         for (int j = 1; j <= n; ++j) scanf("%lf", &a[i][j]);
69         scanf("%lf", &a[i][0]);
70     }
71     for (int i = 0; i <= n + m; ++i) id[i] = 0;
72     for (int i = 1; i <= n; ++i) id[i] = i;
73     if (!init()) {
74         puts("Infeasible");
75         return 0;
76     }
77     if (!simplex()) {
78         puts("Unbounded");
79         return 0;

```

```

80     }
81     printf("%.10lf\n", -a[0][0]);
82     for (int i = 0; i <= n + m; ++i) ans[i] = 0;
83     for (int i = 1; i <= m; ++i) ans[id[n + i]] = a[i][0];
84     for (int i = 1; i <= n; ++i) printf("%.10lf_", ans[i]);
85     puts("");
86 }

```

6.8.2 对偶理论

原始问题：

$$\begin{aligned}
 & \max_{x_1, x_2, \dots, x_n} x_0 = c_1 x_1 + c_2 x_2 + \dots + c_n x_n \\
 & s.t. \begin{cases} A_{i1}x_1 + A_{i2}x_2 + \dots + A_{in}x_n \leq b_i, & i = 1, 2, \dots, m \\ x_j \geq 0, & j = 1, 2, \dots, n \end{cases}
 \end{aligned}$$

对偶问题：

$$\begin{aligned}
 & \min_{w_1, w_2, \dots, w_m} w_0 = b_1 x_1 + b_2 x_2 + \dots + b_m x_m \\
 & s.t. \begin{cases} A_{i1}^T w_1 + A_{i2}^T w_2 + \dots + A_{im}^T w_m \geq c_i, & i = 1, 2, \dots, n \\ w_j \geq 0, & j = 1, 2, \dots, m \end{cases}
 \end{aligned}$$

6.9 高斯消元

6.9.1 行列式

```

1  int ans = 1;
2  for (int i=0; i<n; i++) {
3      for (int j=i; j<n; j++)
4          if (g[j][i]) {
5              for (int k=i; k<n; k++)
6                  swap(g[i][k], g[j][k]);
7              if (j != i) ans *= -1;
8              break;
9          }
10     if (g[i][i] == 0) {
11         ans = 0;
12         break;
13     }
14     for (int j=i+1; j<n; j++) {
15         while (g[j][i]) {
16             int t = g[i][i] / g[j][i];
17             for (int k=i; k<n; k++)
18                 g[i][k] = (g[i][k] + mod - ((LL)t * g[j][k] % mod)) % mod;
19             for (int k=i; k<n; k++)
20                 swap(g[i][k], g[j][k]);

```

```

21         ans *= -1;
22     }
23 }
24 }
25 for (int i=0;i<n;i++)
26     ans = ((LL)ans * g[i][i]) % mod;
27 ans = (ans % mod + mod) % mod;
28 printf("%d\n", ans);

```

6.9.2 Matrix-Tree 定理

对于一张图，建立矩阵 C ， $C[i][i] = i$ 的度数，若 i, j 之间有边，那么 $C[i][j] = -1$ ，否则为 0。这张图的生成树个数等于矩阵 C 的 $n-1$ 阶行列式的值。

6.10 调和级数

$\sum_{i=1}^n \frac{1}{i}$ 在 n 较大时约等于 $\ln(n) + r$ ， r 为欧拉常数，约等于 0.5772156649015328。

6.11 曼哈顿距离的变换

$$|x_1 - x_2| + |y_1 - y_2| = \max(|(x_1 + y_1) - (x_2 + y_2)|, |(x_1 - y_1) - (x_2 - y_2)|)$$

6.12 数论函数变换

常见积性函数：

欧拉函数 $\phi(n)$ 为不超过 n 的与 n 互质的正整数个数

$$\text{莫比乌斯函数 } \mu(n) = \begin{cases} 1, & \text{若 } n = 1 \\ (-1)^k, & \text{若 } n \text{ 无平方数因数, 且 } n = p_1 p_2 \cdots p_k \\ 0, & \text{若 } n \text{ 有大于 } 1 \text{ 的平方数因数} \end{cases}$$

常见积性函数的性质：

$$n = \sum_{d|n} \phi(d)$$

$$\sum_{d|n} \mu(d) = \begin{cases} 1, & n = 1 \\ 0, & n > 1 \end{cases}$$

$$\sum_{i=1}^n \sum_{j=1}^m i \times j [\gcd(i, j) = d] = \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{m}{d} \rfloor} id \times jd [\gcd(i, j) = 1]$$

$$\phi(n) = \sum_{d|n} \mu(d) \frac{n}{d}$$

6.13 莫比乌斯反演

$F(n)$ 和 $f(n)$ 是定义在非负整数集合上的两个函数，则：

$$F(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d) F\left(\frac{n}{d}\right)$$

$$F(n) = \sum_{n|d} f(d) \Rightarrow f(n) = \sum_{n|d} \mu\left(\frac{d}{n}\right) F(d)$$

6.14 线性筛素数

```

1 mu[1]=phi[1]=1;top=0;
2 for (int i=2;i<N;i++) {
3     if (!v[i]) prime[++top]=i, mu[i] = -1, phi[i] = i-1;
4     for (int j=1;i*prime[j]<N && j<=top;j++) {
5         v[i*prime[j]] = 1;
6         if (i%prime[j]) {
7             mu[i*prime[j]] = -mu[i];
8             phi[i*prime[j]] = phi[i] * (prime[j]-1);
9         } else {
10            mu[i*prime[j]] = 0;
11            phi[i*prime[j]] = phi[i] * prime[j];
12            break;
13        }
14    }
15 }
```

6.15 杜教筛

$\text{getphi}(t, x)$ 表示求 $\sum_{i=1}^x i^t \phi(i)$ 。

推导过程：

记 $S(n) = \sum_{i=1}^n f(i)$ ，取任意函数 g 有恒等式

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

其中， $(f \cdot g)$ 表示 f 和 g 的狄利克雷卷积：即： $(f \cdot g)(n) = \sum_{d|n} f(d)g\left(\frac{n}{d}\right)$

关于恒等式的证明：

将 $\sum_{i=2}^n g(i) S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$ 移到左边去，即只需证

$$\sum_{i=1}^n (f \cdot g)(i) = \sum_{i=1}^n g(i) S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

将狄利克雷卷积展开，得：

$$\sum_{i=1}^n \sum_{d|i} g(d) f\left(\frac{i}{d}\right) = \sum_{i=1}^n g(i) S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

即：

$$\sum_{d=1}^n g(d) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} f(i) = \sum_{i=1}^n g(i) S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

显然相等，恒等式证完。

取 $f(i) = i^p \phi(i), g(i) = i^p$ ，则有：

$$S(n) = \sum_{i=1}^n i^p \phi(i) = \sum_{i=1}^n i^{p+1} - \sum_{i=2}^n i^p S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

其中有用到等式 $\sum_{d|n} \phi(d) = n$

另外，莫比乌斯函数 $\mu(n)$ 也可以使用杜教筛求前缀和，记 $S'(n) = \sum_{i=1}^n \mu(i)$ ，则 $S'(n) = 1 -$

$$\sum_{i=2}^n S'\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$$

```

1  #include <bits/stdc++.h>
2
3  #define N 5000020
4  #define LL long long
5  #define mod 1000000007
6  #define div2 ((mod+1)/2)
7  #define div6 ((mod+1)/6)
8
9  using namespace std;
10
11 int n, prime[N], v[N];
12 LL phi[3][N];
13
14 map<int, int> mp[3];
15
16 int sum(int t, int x) { //calculate 1^t + 2^t + ... + x^t
17     if (t == 0) return x;
18     if (t == 1) return 1ll * x * (x + 1) % mod * div2 % mod;
19     if (t == 2) return 1ll * x * (x + 1) % mod * (2ll * x % mod + 1) % mod * div6 %
20         mod;
21     if (t == 3) return 1ll * x * x % mod * (x + 1) % mod * (x + 1) % mod * div2 %
22         mod * div2 % mod;
23 }
24
25 int getphi(int t, int x) {
26     if (x < N) return phi[t][x];
27     if (mp[t].find(x) != mp[t].end()) return mp[t][x];
28     LL ans = 0; int r = 0;
29     for (int l = 2; l <= x; l = r + 1) {
30         r = x / (x / l);
31         ans += sum(t, r) * (phi[t][x] - phi[t][l-1]) % mod;
32     }
33     mp[t][x] = ans;
34     return ans;
35 }
36
37 int main() {
38     int t;
39     while (t = read(), t) {
40         n = read();
41         for (int i = 2; i <= n; i++) {
42             if (!prime[i]) prime[i] = i;
43             for (int j = i; j <= n; j += i) {
44                 v[j] = 1;
45             }
46         }
47         for (int i = 2; i <= n; i++) {
48             int p = i;
49             while (p % i == 0) p /= i;
50             phi[0][i] = i - 1;
51             for (int j = 2; j <= t; j++) {
52                 phi[j][i] = 0;
53                 for (int k = 1; k <= j; k++) {
54                     phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
55                 }
56             }
57         }
58         for (int i = 2; i <= n; i++) {
59             int p = i;
60             while (p % i == 0) p /= i;
61             phi[0][i] = i - 1;
62             for (int j = 2; j <= t; j++) {
63                 phi[j][i] = 0;
64                 for (int k = 1; k <= j; k++) {
65                     phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
66                 }
67             }
68         }
69         for (int i = 2; i <= n; i++) {
70             int p = i;
71             while (p % i == 0) p /= i;
72             phi[0][i] = i - 1;
73             for (int j = 2; j <= t; j++) {
74                 phi[j][i] = 0;
75                 for (int k = 1; k <= j; k++) {
76                     phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
77                 }
78             }
79         }
80         for (int i = 2; i <= n; i++) {
81             int p = i;
82             while (p % i == 0) p /= i;
83             phi[0][i] = i - 1;
84             for (int j = 2; j <= t; j++) {
85                 phi[j][i] = 0;
86                 for (int k = 1; k <= j; k++) {
87                     phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
88                 }
89             }
90         }
91         for (int i = 2; i <= n; i++) {
92             int p = i;
93             while (p % i == 0) p /= i;
94             phi[0][i] = i - 1;
95             for (int j = 2; j <= t; j++) {
96                 phi[j][i] = 0;
97                 for (int k = 1; k <= j; k++) {
98                     phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
99                 }
100            }
101        }
102        for (int i = 2; i <= n; i++) {
103            int p = i;
104            while (p % i == 0) p /= i;
105            phi[0][i] = i - 1;
106            for (int j = 2; j <= t; j++) {
107                phi[j][i] = 0;
108                for (int k = 1; k <= j; k++) {
109                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
110                }
111            }
112        }
113        for (int i = 2; i <= n; i++) {
114            int p = i;
115            while (p % i == 0) p /= i;
116            phi[0][i] = i - 1;
117            for (int j = 2; j <= t; j++) {
118                phi[j][i] = 0;
119                for (int k = 1; k <= j; k++) {
120                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
121                }
122            }
123        }
124        for (int i = 2; i <= n; i++) {
125            int p = i;
126            while (p % i == 0) p /= i;
127            phi[0][i] = i - 1;
128            for (int j = 2; j <= t; j++) {
129                phi[j][i] = 0;
130                for (int k = 1; k <= j; k++) {
131                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
132                }
133            }
134        }
135        for (int i = 2; i <= n; i++) {
136            int p = i;
137            while (p % i == 0) p /= i;
138            phi[0][i] = i - 1;
139            for (int j = 2; j <= t; j++) {
140                phi[j][i] = 0;
141                for (int k = 1; k <= j; k++) {
142                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
143                }
144            }
145        }
146        for (int i = 2; i <= n; i++) {
147            int p = i;
148            while (p % i == 0) p /= i;
149            phi[0][i] = i - 1;
150            for (int j = 2; j <= t; j++) {
151                phi[j][i] = 0;
152                for (int k = 1; k <= j; k++) {
153                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
154                }
155            }
156        }
157        for (int i = 2; i <= n; i++) {
158            int p = i;
159            while (p % i == 0) p /= i;
160            phi[0][i] = i - 1;
161            for (int j = 2; j <= t; j++) {
162                phi[j][i] = 0;
163                for (int k = 1; k <= j; k++) {
164                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
165                }
166            }
167        }
168        for (int i = 2; i <= n; i++) {
169            int p = i;
170            while (p % i == 0) p /= i;
171            phi[0][i] = i - 1;
172            for (int j = 2; j <= t; j++) {
173                phi[j][i] = 0;
174                for (int k = 1; k <= j; k++) {
175                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
176                }
177            }
178        }
179        for (int i = 2; i <= n; i++) {
180            int p = i;
181            while (p % i == 0) p /= i;
182            phi[0][i] = i - 1;
183            for (int j = 2; j <= t; j++) {
184                phi[j][i] = 0;
185                for (int k = 1; k <= j; k++) {
186                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
187                }
188            }
189        }
190        for (int i = 2; i <= n; i++) {
191            int p = i;
192            while (p % i == 0) p /= i;
193            phi[0][i] = i - 1;
194            for (int j = 2; j <= t; j++) {
195                phi[j][i] = 0;
196                for (int k = 1; k <= j; k++) {
197                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
198                }
199            }
200        }
201        for (int i = 2; i <= n; i++) {
202            int p = i;
203            while (p % i == 0) p /= i;
204            phi[0][i] = i - 1;
205            for (int j = 2; j <= t; j++) {
206                phi[j][i] = 0;
207                for (int k = 1; k <= j; k++) {
208                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
209                }
210            }
211        }
212        for (int i = 2; i <= n; i++) {
213            int p = i;
214            while (p % i == 0) p /= i;
215            phi[0][i] = i - 1;
216            for (int j = 2; j <= t; j++) {
217                phi[j][i] = 0;
218                for (int k = 1; k <= j; k++) {
219                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
220                }
221            }
222        }
223        for (int i = 2; i <= n; i++) {
224            int p = i;
225            while (p % i == 0) p /= i;
226            phi[0][i] = i - 1;
227            for (int j = 2; j <= t; j++) {
228                phi[j][i] = 0;
229                for (int k = 1; k <= j; k++) {
230                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
231                }
232            }
233        }
234        for (int i = 2; i <= n; i++) {
235            int p = i;
236            while (p % i == 0) p /= i;
237            phi[0][i] = i - 1;
238            for (int j = 2; j <= t; j++) {
239                phi[j][i] = 0;
240                for (int k = 1; k <= j; k++) {
241                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
242                }
243            }
244        }
245        for (int i = 2; i <= n; i++) {
246            int p = i;
247            while (p % i == 0) p /= i;
248            phi[0][i] = i - 1;
249            for (int j = 2; j <= t; j++) {
250                phi[j][i] = 0;
251                for (int k = 1; k <= j; k++) {
252                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
253                }
254            }
255        }
256        for (int i = 2; i <= n; i++) {
257            int p = i;
258            while (p % i == 0) p /= i;
259            phi[0][i] = i - 1;
260            for (int j = 2; j <= t; j++) {
261                phi[j][i] = 0;
262                for (int k = 1; k <= j; k++) {
263                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
264                }
265            }
266        }
267        for (int i = 2; i <= n; i++) {
268            int p = i;
269            while (p % i == 0) p /= i;
270            phi[0][i] = i - 1;
271            for (int j = 2; j <= t; j++) {
272                phi[j][i] = 0;
273                for (int k = 1; k <= j; k++) {
274                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
275                }
276            }
277        }
278        for (int i = 2; i <= n; i++) {
279            int p = i;
280            while (p % i == 0) p /= i;
281            phi[0][i] = i - 1;
282            for (int j = 2; j <= t; j++) {
283                phi[j][i] = 0;
284                for (int k = 1; k <= j; k++) {
285                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
286                }
287            }
288        }
289        for (int i = 2; i <= n; i++) {
290            int p = i;
291            while (p % i == 0) p /= i;
292            phi[0][i] = i - 1;
293            for (int j = 2; j <= t; j++) {
294                phi[j][i] = 0;
295                for (int k = 1; k <= j; k++) {
296                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
297                }
298            }
299        }
300        for (int i = 2; i <= n; i++) {
301            int p = i;
302            while (p % i == 0) p /= i;
303            phi[0][i] = i - 1;
304            for (int j = 2; j <= t; j++) {
305                phi[j][i] = 0;
306                for (int k = 1; k <= j; k++) {
307                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
308                }
309            }
310        }
311        for (int i = 2; i <= n; i++) {
312            int p = i;
313            while (p % i == 0) p /= i;
314            phi[0][i] = i - 1;
315            for (int j = 2; j <= t; j++) {
316                phi[j][i] = 0;
317                for (int k = 1; k <= j; k++) {
318                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
319                }
320            }
321        }
322        for (int i = 2; i <= n; i++) {
323            int p = i;
324            while (p % i == 0) p /= i;
325            phi[0][i] = i - 1;
326            for (int j = 2; j <= t; j++) {
327                phi[j][i] = 0;
328                for (int k = 1; k <= j; k++) {
329                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
330                }
331            }
332        }
333        for (int i = 2; i <= n; i++) {
334            int p = i;
335            while (p % i == 0) p /= i;
336            phi[0][i] = i - 1;
337            for (int j = 2; j <= t; j++) {
338                phi[j][i] = 0;
339                for (int k = 1; k <= j; k++) {
340                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
341                }
342            }
343        }
344        for (int i = 2; i <= n; i++) {
345            int p = i;
346            while (p % i == 0) p /= i;
347            phi[0][i] = i - 1;
348            for (int j = 2; j <= t; j++) {
349                phi[j][i] = 0;
350                for (int k = 1; k <= j; k++) {
351                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
352                }
353            }
354        }
355        for (int i = 2; i <= n; i++) {
356            int p = i;
357            while (p % i == 0) p /= i;
358            phi[0][i] = i - 1;
359            for (int j = 2; j <= t; j++) {
360                phi[j][i] = 0;
361                for (int k = 1; k <= j; k++) {
362                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
363                }
364            }
365        }
366        for (int i = 2; i <= n; i++) {
367            int p = i;
368            while (p % i == 0) p /= i;
369            phi[0][i] = i - 1;
370            for (int j = 2; j <= t; j++) {
371                phi[j][i] = 0;
372                for (int k = 1; k <= j; k++) {
373                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
374                }
375            }
376        }
377        for (int i = 2; i <= n; i++) {
378            int p = i;
379            while (p % i == 0) p /= i;
380            phi[0][i] = i - 1;
381            for (int j = 2; j <= t; j++) {
382                phi[j][i] = 0;
383                for (int k = 1; k <= j; k++) {
384                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
385                }
386            }
387        }
388        for (int i = 2; i <= n; i++) {
389            int p = i;
390            while (p % i == 0) p /= i;
391            phi[0][i] = i - 1;
392            for (int j = 2; j <= t; j++) {
393                phi[j][i] = 0;
394                for (int k = 1; k <= j; k++) {
395                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
396                }
397            }
398        }
399        for (int i = 2; i <= n; i++) {
400            int p = i;
401            while (p % i == 0) p /= i;
402            phi[0][i] = i - 1;
403            for (int j = 2; j <= t; j++) {
404                phi[j][i] = 0;
405                for (int k = 1; k <= j; k++) {
406                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
407                }
408            }
409        }
410        for (int i = 2; i <= n; i++) {
411            int p = i;
412            while (p % i == 0) p /= i;
413            phi[0][i] = i - 1;
414            for (int j = 2; j <= t; j++) {
415                phi[j][i] = 0;
416                for (int k = 1; k <= j; k++) {
417                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
418                }
419            }
420        }
421        for (int i = 2; i <= n; i++) {
422            int p = i;
423            while (p % i == 0) p /= i;
424            phi[0][i] = i - 1;
425            for (int j = 2; j <= t; j++) {
426                phi[j][i] = 0;
427                for (int k = 1; k <= j; k++) {
428                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
429                }
430            }
431        }
432        for (int i = 2; i <= n; i++) {
433            int p = i;
434            while (p % i == 0) p /= i;
435            phi[0][i] = i - 1;
436            for (int j = 2; j <= t; j++) {
437                phi[j][i] = 0;
438                for (int k = 1; k <= j; k++) {
439                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
440                }
441            }
442        }
443        for (int i = 2; i <= n; i++) {
444            int p = i;
445            while (p % i == 0) p /= i;
446            phi[0][i] = i - 1;
447            for (int j = 2; j <= t; j++) {
448                phi[j][i] = 0;
449                for (int k = 1; k <= j; k++) {
450                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
451                }
452            }
453        }
454        for (int i = 2; i <= n; i++) {
455            int p = i;
456            while (p % i == 0) p /= i;
457            phi[0][i] = i - 1;
458            for (int j = 2; j <= t; j++) {
459                phi[j][i] = 0;
460                for (int k = 1; k <= j; k++) {
461                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
462                }
463            }
464        }
465        for (int i = 2; i <= n; i++) {
466            int p = i;
467            while (p % i == 0) p /= i;
468            phi[0][i] = i - 1;
469            for (int j = 2; j <= t; j++) {
470                phi[j][i] = 0;
471                for (int k = 1; k <= j; k++) {
472                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
473                }
474            }
475        }
476        for (int i = 2; i <= n; i++) {
477            int p = i;
478            while (p % i == 0) p /= i;
479            phi[0][i] = i - 1;
480            for (int j = 2; j <= t; j++) {
481                phi[j][i] = 0;
482                for (int k = 1; k <= j; k++) {
483                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
484                }
485            }
486        }
487        for (int i = 2; i <= n; i++) {
488            int p = i;
489            while (p % i == 0) p /= i;
490            phi[0][i] = i - 1;
491            for (int j = 2; j <= t; j++) {
492                phi[j][i] = 0;
493                for (int k = 1; k <= j; k++) {
494                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
495                }
496            }
497        }
498        for (int i = 2; i <= n; i++) {
499            int p = i;
500            while (p % i == 0) p /= i;
501            phi[0][i] = i - 1;
502            for (int j = 2; j <= t; j++) {
503                phi[j][i] = 0;
504                for (int k = 1; k <= j; k++) {
505                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
506                }
507            }
508        }
509        for (int i = 2; i <= n; i++) {
510            int p = i;
511            while (p % i == 0) p /= i;
512            phi[0][i] = i - 1;
513            for (int j = 2; j <= t; j++) {
514                phi[j][i] = 0;
515                for (int k = 1; k <= j; k++) {
516                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
517                }
518            }
519        }
520        for (int i = 2; i <= n; i++) {
521            int p = i;
522            while (p % i == 0) p /= i;
523            phi[0][i] = i - 1;
524            for (int j = 2; j <= t; j++) {
525                phi[j][i] = 0;
526                for (int k = 1; k <= j; k++) {
527                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
528                }
529            }
530        }
531        for (int i = 2; i <= n; i++) {
532            int p = i;
533            while (p % i == 0) p /= i;
534            phi[0][i] = i - 1;
535            for (int j = 2; j <= t; j++) {
536                phi[j][i] = 0;
537                for (int k = 1; k <= j; k++) {
538                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
539                }
540            }
541        }
542        for (int i = 2; i <= n; i++) {
543            int p = i;
544            while (p % i == 0) p /= i;
545            phi[0][i] = i - 1;
546            for (int j = 2; j <= t; j++) {
547                phi[j][i] = 0;
548                for (int k = 1; k <= j; k++) {
549                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
550                }
551            }
552        }
553        for (int i = 2; i <= n; i++) {
554            int p = i;
555            while (p % i == 0) p /= i;
556            phi[0][i] = i - 1;
557            for (int j = 2; j <= t; j++) {
558                phi[j][i] = 0;
559                for (int k = 1; k <= j; k++) {
560                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
561                }
562            }
563        }
564        for (int i = 2; i <= n; i++) {
565            int p = i;
566            while (p % i == 0) p /= i;
567            phi[0][i] = i - 1;
568            for (int j = 2; j <= t; j++) {
569                phi[j][i] = 0;
570                for (int k = 1; k <= j; k++) {
571                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
572                }
573            }
574        }
575        for (int i = 2; i <= n; i++) {
576            int p = i;
577            while (p % i == 0) p /= i;
578            phi[0][i] = i - 1;
579            for (int j = 2; j <= t; j++) {
580                phi[j][i] = 0;
581                for (int k = 1; k <= j; k++) {
582                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
583                }
584            }
585        }
586        for (int i = 2; i <= n; i++) {
587            int p = i;
588            while (p % i == 0) p /= i;
589            phi[0][i] = i - 1;
590            for (int j = 2; j <= t; j++) {
591                phi[j][i] = 0;
592                for (int k = 1; k <= j; k++) {
593                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
594                }
595            }
596        }
597        for (int i = 2; i <= n; i++) {
598            int p = i;
599            while (p % i == 0) p /= i;
600            phi[0][i] = i - 1;
601            for (int j = 2; j <= t; j++) {
602                phi[j][i] = 0;
603                for (int k = 1; k <= j; k++) {
604                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
605                }
606            }
607        }
608        for (int i = 2; i <= n; i++) {
609            int p = i;
610            while (p % i == 0) p /= i;
611            phi[0][i] = i - 1;
612            for (int j = 2; j <= t; j++) {
613                phi[j][i] = 0;
614                for (int k = 1; k <= j; k++) {
615                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
616                }
617            }
618        }
619        for (int i = 2; i <= n; i++) {
620            int p = i;
621            while (p % i == 0) p /= i;
622            phi[0][i] = i - 1;
623            for (int j = 2; j <= t; j++) {
624                phi[j][i] = 0;
625                for (int k = 1; k <= j; k++) {
626                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
627                }
628            }
629        }
630        for (int i = 2; i <= n; i++) {
631            int p = i;
632            while (p % i == 0) p /= i;
633            phi[0][i] = i - 1;
634            for (int j = 2; j <= t; j++) {
635                phi[j][i] = 0;
636                for (int k = 1; k <= j; k++) {
637                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
638                }
639            }
640        }
641        for (int i = 2; i <= n; i++) {
642            int p = i;
643            while (p % i == 0) p /= i;
644            phi[0][i] = i - 1;
645            for (int j = 2; j <= t; j++) {
646                phi[j][i] = 0;
647                for (int k = 1; k <= j; k++) {
648                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
649                }
650            }
651        }
652        for (int i = 2; i <= n; i++) {
653            int p = i;
654            while (p % i == 0) p /= i;
655            phi[0][i] = i - 1;
656            for (int j = 2; j <= t; j++) {
657                phi[j][i] = 0;
658                for (int k = 1; k <= j; k++) {
659                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
660                }
661            }
662        }
663        for (int i = 2; i <= n; i++) {
664            int p = i;
665            while (p % i == 0) p /= i;
666            phi[0][i] = i - 1;
667            for (int j = 2; j <= t; j++) {
668                phi[j][i] = 0;
669                for (int k = 1; k <= j; k++) {
670                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
671                }
672            }
673        }
674        for (int i = 2; i <= n; i++) {
675            int p = i;
676            while (p % i == 0) p /= i;
677            phi[0][i] = i - 1;
678            for (int j = 2; j <= t; j++) {
679                phi[j][i] = 0;
680                for (int k = 1; k <= j; k++) {
681                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
682                }
683            }
684        }
685        for (int i = 2; i <= n; i++) {
686            int p = i;
687            while (p % i == 0) p /= i;
688            phi[0][i] = i - 1;
689            for (int j = 2; j <= t; j++) {
690                phi[j][i] = 0;
691                for (int k = 1; k <= j; k++) {
692                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
693                }
694            }
695        }
696        for (int i = 2; i <= n; i++) {
697            int p = i;
698            while (p % i == 0) p /= i;
699            phi[0][i] = i - 1;
700            for (int j = 2; j <= t; j++) {
701                phi[j][i] = 0;
702                for (int k = 1; k <= j; k++) {
703                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
704                }
705            }
706        }
707        for (int i = 2; i <= n; i++) {
708            int p = i;
709            while (p % i == 0) p /= i;
710            phi[0][i] = i - 1;
711            for (int j = 2; j <= t; j++) {
712                phi[j][i] = 0;
713                for (int k = 1; k <= j; k++) {
714                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
715                }
716            }
717        }
718        for (int i = 2; i <= n; i++) {
719            int p = i;
720            while (p % i == 0) p /= i;
721            phi[0][i] = i - 1;
722            for (int j = 2; j <= t; j++) {
723                phi[j][i] = 0;
724                for (int k = 1; k <= j; k++) {
725                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
726                }
727            }
728        }
729        for (int i = 2; i <= n; i++) {
730            int p = i;
731            while (p % i == 0) p /= i;
732            phi[0][i] = i - 1;
733            for (int j = 2; j <= t; j++) {
734                phi[j][i] = 0;
735                for (int k = 1; k <= j; k++) {
736                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
737                }
738            }
739        }
740        for (int i = 2; i <= n; i++) {
741            int p = i;
742            while (p % i == 0) p /= i;
743            phi[0][i] = i - 1;
744            for (int j = 2; j <= t; j++) {
745                phi[j][i] = 0;
746                for (int k = 1; k <= j; k++) {
747                    phi[j][i] += phi[k][i] * phi[j-k][i] % mod;
748                }
749            }
750        }
751        for (int i = 2; i <= n; i++) {
752            int p = i;
753            while (p % i == 0) p /= i;
754            phi[0][i] = i - 1;
755            for (int j = 
```



```

29         ans += 111 * getphi(t, x / 1) * (((LL)sum(t, r) - sum(t, 1 - 1) + mod) % mod
30             ) % mod;
31         ans %= mod;
32     }
33     ans = (LL)sum(t + 1, x) - ans + mod;
34     ans %= mod;
35     mp[t][x] = ans;
36     return (int)ans;
37 }
38 int main() {
39     memset(v, 0, sizeof(v));
40     int top = 0;
41     phi[0][1] = 1, phi[1][1] = 1, phi[2][1] = 1;
42     for (int i = 2; i < N; ++i) {
43         if (!v[i]) prime[++top] = i, phi[0][i] = i - 1, phi[1][i] = 111 * i * phi
44             [0][i] % mod, phi[2][i] = 111 * i * phi[1][i] % mod;
45         for (int j = 1; j <= top && prime[j] * i < N; ++j) {
46             v[i * prime[j]] = 1;
47             if (i % prime[j] == 0) {
48                 phi[0][i * prime[j]] = phi[0][i] * prime[j];
49                 phi[1][i * prime[j]] = 111 * phi[1][i] * prime[j] % mod * prime[j] %
50                     mod;
51                 phi[2][i * prime[j]] = 111 * phi[2][i] * prime[j] % mod * prime[j] %
52                     mod * prime[j] % mod;
53                 break;
54             } else {
55                 phi[0][i * prime[j]] = phi[0][i] * (prime[j] - 1);
56                 phi[1][i * prime[j]] = 111 * phi[1][i] * (prime[j] - 1) % mod *
57                     prime[j] % mod;
58                 phi[2][i * prime[j]] = 111 * phi[2][i] * (prime[j] - 1) % mod *
59                     prime[j] % mod * prime[j] % mod;
60             }
61         }
62     }
63 }

```

6.16 FFT

6.16.1 普通 FFT

```

1 namespace FFT {
2     const int maxn = 65537;
3     const double pi = acos(-1.0);

```

```

4
5  struct comp {
6      double real , imag;
7      comp() {}
8      comp(double real , double imag): real(real) , imag(imag) {}
9      friend inline comp operator+(const comp &a , const comp &b) {
10         return comp(a.real + b.real , a.imag + b.imag);
11     }
12     friend inline comp operator-(const comp &a , const comp &b) {
13         return comp(a.real - b.real , a.imag - b.imag);
14     }
15     friend inline comp operator*(const comp &a , const comp &b) {
16         return comp(a.real * b.real - a.imag * b.imag , a.real * b.imag + a.imag
17             * b.real);
18     }
19 };
20
21 comp A[maxn] , B[maxn];
22 int rev[maxn], m, len;
23
24 inline void init(int n) {
25     for (m = 1, len = 0; m < n + n; m <= 1 , len ++);
26     for (int i = 0; i < m; ++i) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (len -
27         1));
28     for (int i = 0; i < m; ++i) A[i] = B[i] = comp(0, 0);
29 }
30
31 inline void dft(comp *a , int v) {
32     for (int i = 0; i < m; ++i) if (i < rev[i]) swap(a[i] , a[rev[i]]);
33     for (int s = 2; s <= m; s <= 1) {
34         comp g(cos(2 * pi / s) , v * sin(2 * pi / s));
35         for (int k = 0; k < m; k += s) {
36             comp w(1 , 0);
37             for (int j = 0; j < s / 2; ++j) {
38                 comp &u = a[k + j + s / 2], &v = a[k + j];
39                 comp t = w * u;
40                 u = v - t;
41                 v = v + t;
42                 w = w * g;
43             }
44         }
45     }
46     if (v == -1)
47         for (int i = 0; i < m; ++i) a[i].real /= m , a[i].imag /= m;
48 }

```

6.16.2 模任意素数 FFT

注意：调用 *mulmod* 前先调用 *init* 。调用 *mulmod* 前请确保 *a, b* 数组足够大（比 $2n$ 大的 2 的整数次幂）且经过初始化。

```
1 namespace FFT {
2     const long double pi = acos(-1.0);
3
4     struct comp {
5         long double real, imag;
6         comp() {}
7         comp(long double real, long double imag) : real(real), imag(imag) {}
8         friend inline comp operator + (const comp &a, const comp &b) {
9             return comp(a.real + b.real, a.imag + b.imag);
10        }
11        friend inline comp operator - (const comp &a, const comp &b) {
12            return comp(a.real - b.real, a.imag - b.imag);
13        }
14        friend inline comp operator * (const comp &a, const comp &b) {
15            return comp(a.real * b.real - a.imag * b.imag, a.real * b.imag + a.imag
16                * b.real);
17        }
18        inline comp conj() {
19            return comp(real, -imag);
20        }
21    };
22
23    comp A[maxn], B[maxn];
24    int rev[maxn], m, len;
25
26    inline void init(int n) {
27        for (m = 1, len = 0; m < n + n; m <= 1, ++len);
28        for (int i = 0; i < m; ++i) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (len -
29            1));
30        for (int i = 0; i < m; ++i) A[i] = B[i] = comp(0, 0);
31    }
32
33    inline void dft(comp *a, int v) {
34        for (int i = 0; i < m; ++i) if (i < rev[i]) swap(a[i], a[rev[i]]);
35        for (int s = 2; s <= m; s <= 1) {
36            comp g(cos(2 * pi / s), v * sin(2 * pi / s));
37            for (int k = 0; k < m; k += s) {
38                comp w(1, 0);
39                for (int j = 0; j < s / 2; ++j) {
40                    comp &u = a[k + j + s / 2], &v = a[k + j];
41                    comp t = w * u;
42                    u = v - t;
43                    v = v + t;
44                    w = w * g;
45                }
46            }
47        }
48    }
```

```

45     }
46     if (v == -1)
47         for (int i = 0; i < m; ++i) a[i].real /= m, a[i].imag /= m;
48 }
49
50 inline void mulmod(int *a, int *b, int *c) { // c = a * b % mod, c不能为a或b
51     int M = sqrt(mod);
52     for (int i = 0; i < m; ++i) {
53         A[i] = comp(a[i] / M, a[i] % M);
54         B[i] = comp(b[i] / M, b[i] % M);
55     }
56     dft(A, 1); dft(B, 1);
57     static comp t[maxn];
58     for (int i = 0; i < m; ++i) {
59         int j = i ? m - i : 0;
60         t[i] = ((A[i] + A[j].conj()) * (B[j].conj() - B[i]) + (A[j].conj() - A[i]
61             ]) * (B[i] + B[j].conj())) * comp(0, 0.25);
62     }
63     dft(t, -1);
64     for (int i = 0; i < m; ++i)
65         c[i] = (LL) (t[i].real + 0.5) % mod * M % mod;
66     for (int i = 0; i < m; ++i) {
67         int j = i ? m - i : 0;
68         t[i] = (A[j].conj() - A[i]) * (B[j].conj() - B[i]) * comp(-0.25, 0) +
69             comp(0, 0.25) * (A[i] + A[j].conj()) * (B[i] + B[j].conj());
70     }
71     dft(t, -1);
72     for (int i = 0; i < m; ++i)
73         c[i] = (1ll * c[i] + (LL) (t[i].real + 0.5) + (LL) (t[i].imag + 0.5) % mod
74             * M * M % mod) % mod;
75 }
76 };

```

6.17 FWT

给定长度为 2^n 的序列 $A[0 \dots 2^n - 1], B[0 \dots 2^n - 1]$ ，求这两序列的

or 卷积: $C_k = \sum_{i \text{ or } j = k} A_i B_j$

and 卷积: $C_k = \sum_{i \text{ and } j = k} A_i B_j$

xor 卷积: $C_k = \sum_{i \text{ xor } j = k} A_i B_j$

```

1 void FWT(int *a, int n) {
2     for (int d = 1; d < n; d <= 1)
3         for (int m = d << 1, i = 0; i < n; i += m)
4             for (int j = 0; j < d; ++j) {
5                 int x = a[i + j], y = a[i + j + d];
6                 //or: a[i + j + d] = x + y;
7                 //and: a[i + j] = x + y;
8                 //xor: a[i + j] = x + y, a[i + j + d] = x - y;

```

```

9         // 如答案要求取模, 此处记得取模
10     }
11 }
12
13 void UFWT(int *a, int n) {
14     for (int d = 1; d < n; d <= 1)
15         for (int m = d << 1, i = 0; i < n; i += m)
16             for (int j = 0; j < d; ++j) {
17                 int x = a[i + j], y = a[i + j + d];
18                 //or: a[i + j + d] = y - x;
19                 //and: a[i + j] = x - y;
20                 //xor: a[i + j] = (x + y) / 2, a[i + j + d] = (x - y) / 2;
21                 // 如答案要求取模, 此处记得取模
22             }
23 }

```

6.18 求原根

接口: LL p_root(LL p);

输入: 一个素数 p

输出: p 的原根

```

1 #include <bits/stdc++.h>
2 #define LL long long
3
4 using namespace std;
5
6 vector <LL> a;
7
8 LL pow_mod(LL base, LL times, LL mod) {
9     LL ret = 1;
10    while (times) {
11        if (times&1) ret = ret * base % mod;
12        base = base * base % mod;
13        times>>=1;
14    }
15    return ret;
16 }
17
18 bool g_test(LL g, LL p) {
19     for (LL i = 0; i < a.size(); ++i)
20         if (pow_mod(g, (p-1)/a[i], p) == 1) return 0;
21     return 1;
22 }
23
24 LL p_root(LL p) {
25     LL tmp = p - 1;
26     for (LL i = 2; i <= tmp / i; ++i)
27         if (tmp % i == 0) {

```

```

28         a.push_back(i);
29         while (tmp % i == 0)
30             tmp /= i;
31     }
32     if (tmp != 1) a.push_back(tmp);
33     LL g = 1;
34     while (1) {
35         if (g_test(g, p)) return g;
36         ++g;
37     }
38 }
39
40 int main() {
41     LL p;
42     cin >> p;
43     cout << p_root(p) << endl;
44 }

```

6.19 NTT

NTT 公式:

$$y_n = \sum_{i=0}^{d-1} x_i (g^{\frac{P-1}{d}})^{in} \bmod P$$

```

1  #define mod 998244353
2  #define gg 3
3
4  int power(int base, int times) {
5      int ans = 1;
6      while (times) {
7          if (times & 1) ans = 1ll * ans * base % mod;
8          base = 1ll * base * base % mod;
9          times >>= 1;
10     }
11     return ans;
12 }
13
14 void NTT(int *x, int n, int reverse) {
15     static int rev[N];
16     int m = 1, len = 0;
17     for (; m < n + n; m <= 1, ++len);
18     for (int i = 0; i < m; ++i) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (len - 1));
19
20     for (int i = 0; i < m; ++i)
21         if (i < rev[i])
22             swap(x[i], x[rev[i]]);
23     for (int h = 2; h <= m; h <= 1) {
24         int wn = power(gg, (mod - 1) / h);
25         if (reverse == -1) wn = power(wn, mod - 2);

```

```

25     for (int i = 0; i < m; i += h) {
26         int w = 1;
27         for (int j = i; j < i + h / 2; ++j) {
28             int u = x[j];
29             int v = 1ll * w * x[j + h / 2] % mod;
30             x[j] = (u + v) % mod;
31             x[j + h / 2] = (u - v + mod) % mod;
32             w = 1ll * w * wn % mod;
33         }
34     }
35 }
36 if (reverse == -1) {
37     int t = power(m, mod - 2);
38     for (int i = 0; i < m; ++i)
39         x[i] = 1ll * x[i] * t % mod;
40 }
41 }
42
43 int A[N], B[N];
44 int main() {
45     memset(A, 0, sizeof(A));
46     memset(B, 0, sizeof(B));
47     NTT(A, len, 1); NTT(B, len, 1);
48     for (int i=0;i<len;i++) A[i] = 1ll * A[i] * B[i] % mod;
49     NTT(A, len, -1);
50 }

```

6.19.1 NTT 常用原根表

这张表格仅包含 $2^{18}k + 1$ 的质数。

| 模数 | 最大长度 | 原根 | 模数 | 最大长度 | 原根 | 模数 | 最大长度 | 原根 |
|----------|---------|----|----------|---------|----|----------|---------|----|
| 786433 | 262144 | 10 | 5767169 | 524288 | 3 | 7340033 | 1048576 | 3 |
| 8650753 | 262144 | 10 | 10223617 | 262144 | 5 | 11272193 | 262144 | 3 |
| 13631489 | 1048576 | 15 | 14155777 | 524288 | 7 | 14942209 | 262144 | 11 |
| 16515073 | 262144 | 5 | 21495809 | 524288 | 3 | 22806529 | 262144 | 13 |
| 23068673 | 2097152 | 3 | 26214401 | 1048576 | 3 | 27000833 | 262144 | 3 |
| 28311553 | 1048576 | 5 | 29884417 | 524288 | 5 | 33292289 | 262144 | 3 |
| 35389441 | 262144 | 7 | 36175873 | 524288 | 7 | 37224449 | 524288 | 3 |
| 38535169 | 262144 | 11 | 40370177 | 524288 | 3 | 41680897 | 262144 | 5 |
| 42729473 | 262144 | 3 | 52166657 | 262144 | 3 | 63700993 | 262144 | 5 |
| 64749569 | 262144 | 6 | 68681729 | 524288 | 3 | 69206017 | 2097152 | 5 |
| 70254593 | 1048576 | 3 | 72613889 | 262144 | 3 | 74711041 | 262144 | 7 |
| 77070337 | 524288 | 13 | 81788929 | 2097152 | 7 | 83361793 | 524288 | 5 |
| 83623937 | 262144 | 3 | 85196801 | 262144 | 3 | 87293953 | 262144 | 7 |
| 90439681 | 262144 | 7 | 93585409 | 262144 | 13 | 93847553 | 524288 | 3 |

| | | | | | | | | |
|-----------|----------|----|-----------|---------|----|-----------|---------|----|
| 100139009 | 524288 | 3 | 101711873 | 1048576 | 3 | 103284737 | 524288 | 3 |
| 104857601 | 4194304 | 3 | 107216897 | 262144 | 3 | 111149057 | 2097152 | 3 |
| 113246209 | 4194304 | 7 | 114032641 | 262144 | 11 | 115081217 | 262144 | 3 |
| 117964801 | 524288 | 14 | 118751233 | 262144 | 5 | 120324097 | 262144 | 7 |
| 120586241 | 1048576 | 6 | 125042689 | 262144 | 13 | 126615553 | 262144 | 10 |
| 127664129 | 262144 | 3 | 130809857 | 262144 | 3 | 132120577 | 2097152 | 5 |
| 136314881 | 2097152 | 3 | 138412033 | 4194304 | 5 | 140771329 | 262144 | 7 |
| 141557761 | 1048576 | 26 | 142344193 | 262144 | 7 | 145489921 | 262144 | 7 |
| 147849217 | 1048576 | 5 | 151257089 | 262144 | 3 | 155189249 | 4194304 | 6 |
| 156499969 | 262144 | 7 | 158072833 | 262144 | 5 | 158334977 | 1048576 | 3 |
| 159645697 | 262144 | 15 | 163577857 | 4194304 | 23 | 167510017 | 262144 | 5 |
| 167772161 | 33554432 | 3 | 169869313 | 2097152 | 5 | 173801473 | 262144 | 5 |
| 175636481 | 524288 | 3 | 178782209 | 524288 | 3 | 184811521 | 262144 | 13 |
| 185597953 | 1048576 | 5 | 186646529 | 2097152 | 3 | 187432961 | 262144 | 6 |
| 189530113 | 262144 | 5 | 191365121 | 524288 | 3 | 199229441 | 2097152 | 3 |
| 200540161 | 262144 | 7 | 204472321 | 1048576 | 19 | 206307329 | 262144 | 3 |
| 207880193 | 262144 | 3 | 211025921 | 262144 | 6 | 211812353 | 2097152 | 3 |
| 214171649 | 262144 | 3 | 215482369 | 524288 | 13 | 215744513 | 262144 | 3 |
| 217317377 | 262144 | 3 | 218628097 | 524288 | 5 | 219676673 | 524288 | 3 |
| 221249537 | 1048576 | 3 | 222035969 | 262144 | 3 | 224133121 | 262144 | 23 |
| 228065281 | 524288 | 7 | 230424577 | 262144 | 5 | 230686721 | 4194304 | 6 |
| 231473153 | 262144 | 3 | 234356737 | 524288 | 7 | 236716033 | 262144 | 5 |
| 239337473 | 262144 | 3 | 239861761 | 262144 | 11 | 240648193 | 524288 | 5 |
| 244842497 | 524288 | 3 | 246415361 | 1048576 | 3 | 249561089 | 2097152 | 3 |
| 253493249 | 262144 | 3 | 254279681 | 524288 | 3 | 256376833 | 524288 | 7 |
| 257949697 | 2097152 | 5 | 260571137 | 524288 | 3 | 261881857 | 262144 | 7 |
| 263454721 | 262144 | 11 | 269221889 | 262144 | 3 | 270532609 | 2097152 | 22 |
| 270794753 | 262144 | 3 | 274726913 | 2097152 | 3 | 276037633 | 262144 | 15 |
| 277086209 | 262144 | 6 | 284950529 | 262144 | 3 | 285474817 | 262144 | 7 |
| 288882689 | 524288 | 3 | 290455553 | 1048576 | 3 | 302252033 | 262144 | 3 |
| 302776321 | 262144 | 17 | 305135617 | 1048576 | 5 | 306708481 | 524288 | 19 |
| 311427073 | 1048576 | 7 | 319291393 | 524288 | 5 | 323223553 | 262144 | 5 |
| 325844993 | 262144 | 3 | 328728577 | 524288 | 10 | 329515009 | 262144 | 13 |
| 329777153 | 524288 | 5 | 330301441 | 1048576 | 22 | 332660737 | 262144 | 10 |
| 336068609 | 524288 | 3 | 336855041 | 262144 | 3 | 340000769 | 262144 | 3 |
| 347078657 | 1048576 | 3 | 349962241 | 262144 | 7 | 351797249 | 524288 | 3 |
| 359661569 | 1048576 | 3 | 360972289 | 262144 | 7 | 361758721 | 1048576 | 29 |
| 371458049 | 262144 | 3 | 374603777 | 262144 | 3 | 376963073 | 524288 | 3 |
| 377487361 | 8388608 | 7 | 383778817 | 2097152 | 5 | 384040961 | 262144 | 3 |
| 386400257 | 524288 | 3 | 387186689 | 262144 | 3 | 387973121 | 2097152 | 6 |
| 390332417 | 262144 | 3 | 391643137 | 524288 | 5 | 395837441 | 524288 | 6 |

| | | | | | | | | |
|-----------|----------|----|-----------|---------|----|-----------|---------|----|
| 399507457 | 1048576 | 5 | 404226049 | 524288 | 7 | 409993217 | 1048576 | 3 |
| 413925377 | 262144 | 3 | 415236097 | 4194304 | 5 | 416808961 | 524288 | 37 |
| 424148993 | 524288 | 3 | 429391873 | 524288 | 10 | 433586177 | 524288 | 3 |
| 434896897 | 262144 | 15 | 438829057 | 524288 | 5 | 442761217 | 262144 | 5 |
| 444334081 | 262144 | 37 | 447741953 | 1048576 | 3 | 452198401 | 262144 | 11 |
| 455344129 | 262144 | 13 | 458752001 | 524288 | 6 | 459276289 | 2097152 | 11 |
| 460849153 | 524288 | 5 | 462684161 | 262144 | 3 | 463470593 | 2097152 | 3 |
| 464781313 | 262144 | 5 | 466354177 | 262144 | 10 | 468713473 | 1048576 | 5 |
| 469762049 | 67108864 | 3 | 471072769 | 262144 | 7 | 473694209 | 262144 | 6 |
| 475267073 | 262144 | 3 | 478937089 | 262144 | 13 | 483131393 | 262144 | 3 |
| 483655681 | 262144 | 14 | 487063553 | 524288 | 3 | 489422849 | 262144 | 3 |
| 493879297 | 1048576 | 10 | 495452161 | 524288 | 11 | 498597889 | 524288 | 7 |
| 500432897 | 262144 | 5 | 511967233 | 262144 | 5 | 517472257 | 524288 | 5 |
| 518520833 | 524288 | 3 | 524812289 | 524288 | 3 | 526123009 | 262144 | 7 |
| 529268737 | 262144 | 5 | 531628033 | 1048576 | 5 | 533463041 | 262144 | 3 |
| 536608769 | 262144 | 3 | 537133057 | 262144 | 5 | 539754497 | 262144 | 3 |
| 540540929 | 524288 | 3 | 541327361 | 262144 | 3 | 549978113 | 524288 | 3 |
| 551288833 | 262144 | 5 | 552861697 | 262144 | 5 | 555220993 | 524288 | 7 |
| 561774593 | 262144 | 3 | 564658177 | 524288 | 5 | 568066049 | 262144 | 3 |
| 569638913 | 262144 | 3 | 570163201 | 262144 | 7 | 570949633 | 524288 | 5 |
| 576454657 | 262144 | 10 | 576716801 | 2097152 | 6 | 581959681 | 1048576 | 11 |
| 582746113 | 262144 | 5 | 583794689 | 262144 | 3 | 584581121 | 524288 | 3 |
| 590872577 | 524288 | 3 | 595591169 | 8388608 | 3 | 597688321 | 2097152 | 11 |
| 605028353 | 1048576 | 3 | 605552641 | 524288 | 17 | 606339073 | 262144 | 5 |
| 607911937 | 262144 | 7 | 608698369 | 524288 | 7 | 611844097 | 524288 | 5 |
| 612892673 | 524288 | 3 | 615776257 | 262144 | 5 | 619184129 | 524288 | 3 |
| 621281281 | 524288 | 7 | 626262017 | 262144 | 3 | 629932033 | 262144 | 14 |
| 632553473 | 262144 | 3 | 635437057 | 2097152 | 11 | 637009921 | 524288 | 17 |
| 638058497 | 524288 | 3 | 639369217 | 262144 | 5 | 639631361 | 2097152 | 6 |
| 644087809 | 262144 | 11 | 645922817 | 8388608 | 3 | 648019969 | 2097152 | 17 |
| 649592833 | 524288 | 5 | 651952129 | 262144 | 7 | 655360001 | 1048576 | 3 |
| 657719297 | 262144 | 3 | 660078593 | 524288 | 3 | 663224321 | 524288 | 3 |
| 665583617 | 262144 | 3 | 666894337 | 4194304 | 5 | 675545089 | 262144 | 11 |
| 675807233 | 524288 | 3 | 681312257 | 262144 | 3 | 683409409 | 262144 | 13 |
| 683671553 | 4194304 | 3 | 684982273 | 262144 | 5 | 687603713 | 262144 | 3 |
| 690749441 | 262144 | 3 | 692846593 | 262144 | 5 | 699138049 | 262144 | 19 |
| 699924481 | 524288 | 17 | 703070209 | 524288 | 11 | 704905217 | 262144 | 3 |
| 710410241 | 524288 | 3 | 710934529 | 2097152 | 17 | 712769537 | 262144 | 3 |
| 714342401 | 262144 | 3 | 715128833 | 2097152 | 3 | 717488129 | 262144 | 3 |
| 718274561 | 1048576 | 3 | 720633857 | 262144 | 3 | 725876737 | 262144 | 7 |
| 730595329 | 262144 | 17 | 734527489 | 524288 | 7 | 737673217 | 524288 | 11 |

| | | | | | | | | |
|------------|---------|----|------------|---------|----|------------|----------|----|
| 740294657 | 2097152 | 3 | 741605377 | 262144 | 11 | 745537537 | 1048576 | 5 |
| 748158977 | 524288 | 3 | 753664001 | 262144 | 3 | 754974721 | 16777216 | 11 |
| 758906881 | 262144 | 11 | 759693313 | 524288 | 5 | 760741889 | 524288 | 3 |
| 763887617 | 524288 | 3 | 769130497 | 524288 | 15 | 770703361 | 1048576 | 11 |
| 771489793 | 262144 | 10 | 772538369 | 262144 | 6 | 775421953 | 524288 | 5 |
| 781975553 | 262144 | 3 | 782499841 | 262144 | 11 | 786432001 | 2097152 | 7 |
| 790364161 | 262144 | 14 | 792199169 | 524288 | 3 | 793509889 | 262144 | 11 |
| 795082753 | 262144 | 5 | 798228481 | 262144 | 13 | 799014913 | 2097152 | 13 |
| 800063489 | 1048576 | 3 | 800849921 | 262144 | 6 | 801374209 | 262144 | 14 |
| 802160641 | 1048576 | 11 | 808714241 | 262144 | 3 | 810024961 | 524288 | 13 |
| 811859969 | 262144 | 3 | 813170689 | 524288 | 13 | 813432833 | 262144 | 3 |
| 818937857 | 1048576 | 5 | 820248577 | 262144 | 5 | 820510721 | 524288 | 3 |
| 821297153 | 262144 | 3 | 824180737 | 2097152 | 5 | 824442881 | 262144 | 3 |
| 825753601 | 524288 | 23 | 828112897 | 262144 | 10 | 829685761 | 262144 | 19 |
| 833617921 | 1048576 | 13 | 835452929 | 262144 | 3 | 839385089 | 524288 | 3 |
| 842530817 | 524288 | 3 | 844627969 | 524288 | 17 | 844890113 | 262144 | 3 |
| 848560129 | 262144 | 22 | 850395137 | 1048576 | 3 | 851705857 | 262144 | 5 |
| 860618753 | 262144 | 3 | 862978049 | 1048576 | 3 | 863764481 | 262144 | 3 |
| 864550913 | 524288 | 3 | 867434497 | 262144 | 5 | 872153089 | 262144 | 7 |
| 873725953 | 262144 | 10 | 875298817 | 262144 | 5 | 879230977 | 524288 | 15 |
| 880803841 | 8388608 | 26 | 881590273 | 262144 | 5 | 883949569 | 1048576 | 7 |
| 885522433 | 524288 | 5 | 888668161 | 524288 | 14 | 889454593 | 262144 | 15 |
| 894959617 | 524288 | 10 | 896008193 | 524288 | 3 | 897318913 | 262144 | 5 |
| 897581057 | 8388608 | 3 | 899678209 | 2097152 | 7 | 900464641 | 262144 | 7 |
| 903086081 | 262144 | 3 | 907018241 | 1048576 | 3 | 907542529 | 524288 | 7 |
| 907804673 | 262144 | 3 | 908328961 | 262144 | 26 | 909377537 | 262144 | 3 |
| 913309697 | 1048576 | 3 | 914096129 | 262144 | 3 | 918552577 | 4194304 | 5 |
| 919339009 | 262144 | 59 | 919601153 | 1048576 | 3 | 924844033 | 2097152 | 5 |
| 925892609 | 1048576 | 3 | 932970497 | 262144 | 3 | 935329793 | 4194304 | 3 |
| 938475521 | 1048576 | 3 | 940572673 | 1048576 | 7 | 943718401 | 4194304 | 7 |
| 946339841 | 524288 | 3 | 948699137 | 262144 | 3 | 950009857 | 2097152 | 7 |
| 951582721 | 524288 | 14 | 957349889 | 1048576 | 6 | 958136321 | 262144 | 3 |
| 958922753 | 524288 | 3 | 962592769 | 2097152 | 7 | 962854913 | 262144 | 3 |
| 969146369 | 262144 | 3 | 971243521 | 262144 | 28 | 972029953 | 1048576 | 10 |
| 975175681 | 2097152 | 17 | 976224257 | 1048576 | 3 | 977534977 | 262144 | 5 |
| 979107841 | 262144 | 11 | 980156417 | 262144 | 3 | 983826433 | 262144 | 11 |
| 985661441 | 4194304 | 3 | 993263617 | 262144 | 5 | 995622913 | 524288 | 5 |
| 998244353 | 8388608 | 3 | 1004535809 | 2097152 | 3 | 1005060097 | 524288 | 5 |
| 1006108673 | 524288 | 3 | 1007681537 | 1048576 | 3 | 1010565121 | 262144 | 7 |
| 1012924417 | 2097152 | 5 | 1015283713 | 262144 | 5 | 1018429441 | 262144 | 11 |
| 1019478017 | 262144 | 3 | 1023148033 | 262144 | 7 | 1036779521 | 262144 | 3 |

| | | | | | | | | |
|------------|---------|----|------------|---------|----|------------|---------|---|
| 1037303809 | 262144 | 21 | 1045430273 | 1048576 | 3 | 1049100289 | 524288 | 7 |
| 1051721729 | 1048576 | 6 | 1052508161 | 262144 | 3 | 1053818881 | 1048576 | 7 |
| 1056178177 | 262144 | 5 | 1056440321 | 524288 | 3 | 1062469633 | 262144 | 5 |
| 1068236801 | 262144 | 3 | 1073479681 | 262144 | 11 | | | |

6.19.2 多项式求逆元

对于一个多项式 $A(x)$ ，如果存在 $B(x)$ 满足 $\deg(B) \leq \deg(A)$ 并且 $A(x)B(x) \equiv 1 \pmod{x^n}$ ，那么称 $B(x)$ 为 $A(x)$ 在 $\text{mod } x^n$ 意义下的逆元，记为 $A^{-1}(x)$ 。

```

1 // x := 1 / y
2 void inverse(int n0, int *x, const int *y) {
3     static int fy[N];
4     x[0] = power(y[0], mod - 2);
5     for (int i = 1; i < n0; i <= 1) {
6         for (int j = 0; j < 4 * i; ++j) {
7             fy[j] = (j < 2 * i) ? y[j] : 0;
8             if (j >= i) x[j] = 0;
9         }
10        NTT(fy, 2 * i, 1);
11        NTT(x, 2 * i, 1);
12        for (int j = 0; j < 4 * i; ++j) {
13            x[j] = (2 * x[j] - 1ll * x[j] * x[j] % mod * fy[j]) % mod;
14            if (x[j] < 0) x[j] += mod;
15        }
16        NTT(x, 2 * i, -1);
17    }
18 }

```

6.19.3 多项式取对数

```

1 // x := log(y)
2 void logarithm(int n0, int *x, int *y) {
3     static int tmp[N];
4     static int invs[N];
5     inverse(n0, x, y);
6     for (int i = 0; i < n0 * 2; ++i) {
7         tmp[i] = i < n0 - 1 ? 1ll * y[i + 1] * (i + 1) % mod : 0;
8         if (i >= n0) x[i] = 0;
9     }
10    NTT(tmp, n0, 1);
11    NTT(x, n0, 1);
12    for (int i = 0; i < n0 * 2; ++i)
13        x[i] = 1ll * x[i] * tmp[i] % mod;
14    NTT(x, n0, -1);
15    invs[1] = 1;
16    for (int i = 2; i < n0; ++i)
17        invs[i] = (mod - 1ll * mod / i * invs[mod % i] % mod) % mod;

```

```

18     for (int i = n0 - 1; i; --i)
19         x[i] = 1ll * x[i - 1] * invs[i] % mod;
20     x[0] = 0;
21 }

```

6.19.4 多项式取指数

```

1 // a := exp(b)
2 void exponent(int n0, int *a, int *b) {
3     static int fb[N], x[N], y[N];
4     a[0] = 1;
5     for (int i = 1; i < n0; i <= 1) {
6         for (int j = 0; j < i * 2; ++j)
7             y[j] = (j < i) ? a[j] : 0;
8         logarithm(i * 2, x, y);
9         for (int j = 0; j < 4 * i; ++j) {
10             fb[j] = !j;
11             if (j < 2 * i) {
12                 fb[j] = (fb[j] + b[j]) % mod;
13                 fb[j] = (fb[j] + mod - x[j]) % mod;
14             }
15             if (j >= i) a[j] = 0;
16         }
17         NTT(a, 2 * i, 1);
18         NTT(fb, 2 * i, 1);
19         for (int j = 0; j < 4 * i; ++j)
20             a[j] = 1ll * a[j] * fb[j] % mod;
21         NTT(a, 2 * i, -1);
22     }
23 }

```

6.20 Berlekamp Messay 算法求线性递推式

适合所有 $S_n = \sum_{i=1}^L a_i S_{n-i}$ 的递推式。只需在 $vector < int > t$ 中输入前 $2L$ 项，即可计算出第 m 项的值 modulo MOD。

时间复杂度 $O(L^2 \log(m))$ 。

异常处理：若提示 48 行 assertion error (`assert(l * 2 + 1 < s.size())`)，则表示输入项数不足 $2L + 2$ 项，需要更多的项来确定线性递推式。

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4 typedef long long ll;
5
6 int MOD;
7
8 int bin(int a, int n) {

```

```

9      int res = 1;
10     while (n) {
11         if (n & 1) res = 1LL * res * a % MOD;
12         a = 1LL * a * a % MOD;
13         n >>= 1;
14     }
15     return res;
16 }
17
18 int inv(int x) {
19     return bin(x, MOD - 2);
20 }
21
22 vector<int> berlekamp(vector<int> s) {
23     int l = 0;
24     vector<int> la(1, 1);
25     vector<int> b(1, 1);
26     for (int r = 1; r <= (int)s.size(); r++) {
27         int delta = 0;
28         for (int j = 0; j <= l; j++) {
29             delta = (delta + 1LL * s[r - 1 - j] * la[j]) % MOD;
30         }
31         b.insert(b.begin(), 0);
32         if (delta != 0) {
33             vector<int> t(max(la.size(), b.size()));
34             for (int i = 0; i < (int)t.size(); i++) {
35                 if (i < (int)la.size()) t[i] = (t[i] + la[i]) % MOD;
36                 if (i < (int)b.size()) t[i] = (t[i] - 1LL * delta * b[i] % MOD + MOD
37                     ) % MOD;
38             }
39             if (2 * l <= r - 1) {
40                 b = la;
41                 int od = inv(delta);
42                 for (int &x : b) x = 1LL * x * od % MOD;
43                 l = r - 1;
44             }
45             la = t;
46         }
47     }
48     assert(la.size() == l + 1);
49     assert(l * 2 + 1 < s.size());
50     reverse(la.begin(), la.end());
51     return la;
52 }
53
54 vector<int> mul(vector<int> a, vector<int> b) {
55     vector<int> c(a.size() + b.size() - 1);
56     for (int i = 0; i < (int)a.size(); i++) {
57         for (int j = 0; j < (int)b.size(); j++) {
58             c[i + j] = (c[i + j] + 1LL * a[i] * b[j]) % MOD;
59         }
60     }
61     return c;
62 }

```

```

58     }
59 }
60 vector<int> res(c.size());
61 for (int i = 0; i < (int)res.size(); i++) res[i] = c[i] % MOD;
62 return res;
63 }
64
65 vector<int> mod(vector<int> a, vector<int> b) {
66     if (a.size() < b.size()) a.resize(b.size() - 1);
67
68     int o = inv(b.back());
69     for (int i = (int)a.size() - 1; i >= b.size() - 1; i--) {
70         if (a[i] == 0) continue;
71         int coef = 1LL * o * (MOD - a[i]) % MOD;
72         for (int j = 0; j < (int)b.size(); j++) {
73             a[i - (int)b.size() + 1 + j] = (a[i - (int)b.size() + 1 + j] + 1LL *
74                 coef * b[j]) % MOD;
75         }
76     }
77     while (a.size() >= b.size()) {
78         assert(a.back() == 0);
79         a.pop_back();
80     }
81     return a;
82 }
83
84 vector<int> bin(int n, vector<int> p) {
85     vector<int> res(1, 1);
86     vector<int> a(2); a[1] = 1;
87     while (n) {
88         if (n & 1) res = mod(mul(res, a), p);
89         a = mod(mul(a, a), p);
90         n >>= 1;
91     }
92     return res;
93 }
94
95 void solve() {
96     int m = 22;
97     vector<int> t;
98     t.push_back(1);
99     t.push_back(9);
100    t.push_back(41);
101    t.push_back(109);
102    t.push_back(205);
103    t.push_back(325);
104    t.push_back(473);
105    t.push_back(649);
106    t.push_back(853);
107    t.push_back(1085);

```

```

107     t.push_back(1345);
108     t.push_back(1633);
109     t.push_back(1949);
110     t.push_back(2293);
111
112     MOD = 998244353;
113     vector<int> v = berlekamp(t);
114     vector<int> o = bin(m - 1, v);
115     int res = 0;
116     for (int i = 0; i < (int)o.size(); i++) res = (res + 1LL * o[i] * t[i]) % MOD;
117     printf("%d\n", res);
118 }
119
120 int main() {
121     solve();
122     return 0;
123 }

```

6.21 幂和

$$\begin{aligned}
 \sum_{i=1}^n i^1 &= \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n \\
 \sum_{i=1}^n i^2 &= \frac{n(n+1)(2n+1)}{6} = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n \\
 \sum_{i=1}^n i^3 &= \frac{n^2(n+1)^2}{4} = \frac{1}{4}n^4 + \frac{1}{2}n^3 + \frac{1}{4}n^2 \\
 \sum_{i=1}^n i^4 &= \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30} = \frac{1}{5}n^5 + \frac{1}{2}n^4 + \frac{1}{3}n^3 - \frac{1}{30}n \\
 \sum_{i=1}^n i^5 &= \frac{n^2(n+1)^2(2n^2+2n-1)}{12} = \frac{1}{6}n^6 + \frac{1}{2}n^5 + \frac{5}{12}n^4 - \frac{1}{12}n^2 \\
 \sum_{i=1}^n i^6 &= \frac{n(n+1)(2n+1)(3n^4+6n^3-3n+1)}{42} = \frac{1}{7}n^7 + \frac{1}{2}n^6 + \frac{1}{2}n^5 - \frac{1}{6}n^3 + \frac{1}{42}n
 \end{aligned}$$

6.22 蔡勒公式

$$w = (\lfloor \frac{c}{4} \rfloor - 2c + y + \lfloor \frac{y}{4} \rfloor + \lfloor \frac{13(m+1)}{5} \rfloor + d - 1) \bmod 7$$

w : 0 星期日, 1 星期一, 2 星期二, 3 星期三, 4 星期四, 5 星期五, 6 星期六

c : 年份前两位数

y : 年份后两位数

m : 月 ($3 \leq m \leq 14$, 即在蔡勒公式中, 1、2 月要看作上一年的 13、14 月来计算)

d : 日

6.23 皮克定理

给定顶点坐标均是整点（或正方形格点）的简单多边形（凸多边形），皮克定理说明了其面积 S 和内部格点数目 n 、边上格点数目 s 的关系： $S = n + \frac{s}{2} + 1$ 。

6.24 组合数 lcm

$$(n+1)lcm(C(n,0), C(n,1), \dots, C(n,k)) = lcm(n+1, n, n-1, \dots, n-k+1)$$

6.25 区间 lcm 的维护

对于一个数，将其分解质因数，若有因子 p^k ，那么拆分出 k 个数 p, p^2, \dots, p^k ，权值都为 p ，那么查询区间 $[l, r]$ 内所有数的 lcm 的答案 = 所有在该区间中出现过的数的权值之积，可持久化线段树维护即可。

7 几何

7.1 二维计算几何

7.1.1 计算几何误差修正

```
1 const double pi = acos(-1.0);
2 const double eps = 1e-8;
3
4 inline double sqr(double x) {
5     return x * x;
6 }
7
8 inline int sgn(double x) {
9     if (x < -eps) return -1;
10    return x > eps;
11 }
12
13 inline int cmp(double x, double y) {
14     return sgn(x - y);
15 }
```

7.1.2 计算几何点类

成员函数：

read() 输入一个点

norm() 计算向量的模长

相关函数：

double sqr(double x)

计算一个数的平方

double det(const point &a, const point &b)

计算两个向量的叉积

double dot(const point &a, const point &b)

计算两个向量的点积

double dis(const point &a, const point &b)

计算两个点的距离

point rotate_point(const point &p, double A) \overrightarrow{OP} 绕原点逆时针旋转 A 弧度


```

1 struct point {
2     double x, y;
3     point() : x(0), y(0) {}
4     point(double a, double b) : x(a), y(b) {}
5     inline void read() {
6         scanf("%lf%lf", &x, &y);
7     }
8     inline friend point operator + (const point &a, const point &b) {
9         return point(a.x + b.x, a.y + b.y);
10    }
11    inline friend point operator - (const point &a, const point &b) {
12        return point(a.x - b.x, a.y - b.y);
13    }
14    inline friend bool operator == (const point &a, const point &b) {
15        return cmp(a.x, b.x) == 0 && cmp(a.y, b.y) == 0;
16    }
17    inline friend point operator * (const double &a, const point &b) {
18        return point(a * b.x, a * b.y);
19    }
20    inline friend point operator / (const point &a, const double &b) {
21        return point(a.x / b, a.y / b);
22    }
23    inline double norm() const {
24        return sqrt(sqr(x) + sqr(y));
25    }
26 };
27
28 inline double det(const point &a, const point &b) {
29     return a.x * b.y - a.y * b.x;
30 }
31
32 inline double dot(const point &a, const point &b) {
33     return a.x * b.x + a.y * b.y;
34 }
35
36 inline double dis(const point &a, const point &b) {
37     return (a - b).norm();
38 }
39
40 inline point rotate_point(const point &p, double A) {
41     double tx = p.x, ty = p.y;
42     return point(tx * cos(A) - ty * sin(A), tx * sin(A) + ty * cos(A));
43 }

```

7.1.3 计算几何线段类

相关函数:

bool point_on_segment(const point &p, const segment &l) 判断点 p 是否在线段 l 上 (含端点)

double point_to_segment_dist(const point &p, const segment &l) 求点 p 到线段 l 的距离
point sym_point(const point &p, const segment &l) 求点 p 关于线段 l 的对称点
point point_proj_line(const point &p, const segment &l) 求点 p 到线段 l 的垂足
bool parallel(const segment &a, const segment &b) 判断线段 a 和线段 b 是否平行
point intersect_point(const segment &a, const segment &b) 求直线 a 与直线 b 的交点 (如要求线段 a 与线段 b 的交点, 应先判断是否有)
bool is_segment_intersect(const segment &l1, const segment &l2) 判断线段 a 与线段 b 是否相交 (含端点) (如不含端点, 将 \leq 改为 $<$)
bool is_line_intersect_segment(const point &p1, const point &p2, const segment &l) 判断直线 p_1p_2 是否与线段 l 相交
bool is_half_line_intersect_segment(const point &p1, const point &p2, const segment &l) 判断射线 p_1p_2 是否与线段 l 相交 (含端点 p_1) (如不含端点 p_1 , 将 \geq 改为 $>$)

```

1  struct segment {
2      point a, b;
3      segment() {}
4      segment(point x, point y) : a(x), b(y) {}
5      void read() {
6          a.read(); b.read();
7      }
8  };
9
10 // determine whether point p is on segment l
11 bool point_on_segment(const point &p, const segment &l) {
12     if ((cmp(l.a.x, p.x) <= 0 || cmp(l.b.x, p.x) <= 0) &&
13         (cmp(l.a.x, p.x) >= 0 || cmp(l.b.x, p.x) >= 0) &&
14         (cmp(l.a.y, p.y) <= 0 || cmp(l.b.y, p.y) <= 0) &&
15         (cmp(l.a.y, p.y) >= 0 || cmp(l.b.y, p.y) >= 0)) {
16         return sgn(det(p - l.a, l.b - l.a)) == 0;
17     }
18     return 0;
19 }
20
21 // determine the distance from the point p to segment l
22 double point_to_segment_dist(const point &p, const segment &l) {
23     if (dis(l.a, l.b) < eps) return dis(p, l.a);
24     if (sgn(dot(l.b - l.a, p - l.a)) < 0) return dis(l.a, p);
25     if (sgn(dot(l.a - l.b, p - l.b)) < 0) return dis(l.b, p);
26     return fabs(det(l.b - l.a, p - l.a)) / dis(l.b, l.a);
27 }
28
29 // determine the symmetrical point of point p on segment l
30 point sym_point(const point &p, const segment &l) {
31     double a = l.b.x - l.a.x;
32     double b = l.b.y - l.a.y;
33     double t = ((p.x - l.a.x) * a + (p.y - l.a.y) * b) / (a * a + b * b);
34     return point(2 * l.a.x + 2 * a * t - p.x, 2 * l.a.y + 2 * b * t - p.y);
35 }
36

```

```

37 point point_proj_line(const point &p, const segment &l) {
38     double r = dot((l.b - l.a), (p - l.a)) / dot(l.b - l.a, l.b - l.a);
39     return l.a + r * (l.b - l.a);
40 }
41
42 bool parallel(const segment &a, const segment &b) {
43     return sgn(det(a.a - a.b, b.a - b.b)) == 0;
44 }
45
46 point intersect_point(const segment &a, const segment &b) {
47     double s1 = det(a.a - b.a, b.b - b.a);
48     double s2 = det(a.b - b.a, b.b - b.a);
49     return (s1 * a.b - s2 * a.a) / (s1 - s2);
50 }
51
52 // determine whether segment l1 intersects with segment l2
53 bool is_segment_intersect(const segment &l1, const segment &l2) {
54     const point &s1 = l1.a, &e1 = l1.b;
55     const point &s2 = l2.a, &e2 = l2.b;
56     if ( cmp( min(s1.x, e1.x), max(s2.x, e2.x) ) <= 0 &&
57         cmp( min(s1.y, e1.y), max(s2.y, e2.y) ) <= 0 &&
58         cmp( min(s2.x, e2.x), max(s1.x, e1.x) ) <= 0 &&
59         cmp( min(s2.y, e2.y), max(s1.y, e1.y) ) <= 0 &&
60         sgn( det(s2 - s1, e2 - s1) ) * sgn( det(s2 - e1, e2 - e1) ) <= 0 &&
61         sgn( det(s1 - s2, e1 - s2) ) * sgn( det(s1 - e2, e1 - e2) ) <= 0)
62         return 1;
63     return 0;
64 }
65
66 // determine whether line plp2 intersects with segment l
67 bool is_line_intersect_segment(const point &p1, const point &p2, const segment &l) {
68     assert(!(p1 == p2));
69     return sgn( det(p1 - l.a, p2 - l.a) ) * sgn( det(p1 - l.b, p2 - l.b) ) <= 0;
70 }
71
72 // determine whether half-line plp2 intersects with segment l
73 bool is_half_line_intersect_segment(const point &p1, const point &p2, const segment
74     &l) {
75     return is_line_intersect_segment(p1, p2, l) && sgn( det(p1 - l.a, p2 - l.a) ) *
76         sgn( det(p1 - l.a, l.b - l.a) ) >= 0;
77 }

```

7.2 凸包

```

1 typedef complex<int> point;
2 #define X real()
3 #define Y imag()
4 int n;
5 long long cross(point a, point b) {

```

```

6     return 1ll * a.X * b.Y - 1ll * a.Y * b.X;
7 }
8 bool cmp(point a, point b) {
9     return make_pair(a.X, a.Y) < make_pair(b.X, b.Y);
10 }
11 int convexHull(point p[], int n, point ch[]) {
12     sort(p, p + n, cmp);
13     int m = 0;
14     for(int i = 0; i < n; ++i) {
15         while(m > 1 && cross(ch[m-1] - ch[m-2], p[i] - ch[m-2]) <= 0) m--;
16         ch[m++] = p[i];
17     }
18     int k = m;
19     for(int i = n - 2; i >= 0; --i) {
20         while(m > k && cross(ch[m-1] - ch[m-2], p[i] - ch[m-2]) <= 0) m--;
21         ch[m++] = p[i];
22     }
23     if(n > 1) m--;
24     return m;
25 }

```

7.3 半平面交

输入 vec1 表示所有的半平面 $y \geq kx + b$ 的参数 k 和 b 。

输出 vec2 表示下凸壳 (对应 $y \geq kx + b$) 或者上凸壳 (对应 $y \leq kx + b$)。

```

1 vector< pair< LL, LL > > vec1, vec2;
2
3 LL getval(int t, LL x) {
4     return vec2[t].first * x + vec2[t].second;
5 }
6
7 void solve() {
8     // vec1 stores pair< k, b > for all plane y >=(or <=) kx + b
9     sort(vec1.begin(), vec1.end());
10    // reverse(vec1.begin(), vec1.end()); // if y <= kx + b
11    for (int i = 0; i < vec1.size(); ++i) {
12        while (vec2.size() >= 2) {
13            LL k1 = vec2[vec2.size() - 2].first;
14            LL b1 = vec2[vec2.size() - 2].second;
15            LL k2 = vec2[vec2.size() - 1].first;
16            LL b2 = vec2[vec2.size() - 1].second;
17            LL k3 = vec1[i].first;
18            LL b3 = vec1[i].second;
19            if ((b2 - b1) * (k2 - k3) >= (b3 - b2) * (k1 - k2))
20                vec2.pop_back();
21            else
22                break;
23        }
24        vec2.push_back(vec1[i]);

```

```
25     }
26 }
```

8 黑科技和杂项

8.1 找规律

此方法已过时，请参照“数学 > Berlekamp Messay 算法求线性递推式”。本法使用矩阵快速幂，效率 $O(L^3 \log(m))$ ，而用 Berlekamp 加多项式快速幂可以做到 $O(L^2 \log(m))$ ，故不推荐使用本法。

有些题目，只给一个正整数 n ，然后要求输出一个答案。这时，我们可以暴力得到小数据的解，用高斯消元得到递推式，然后用矩阵快速幂求解。

使用方法：

首先在 `gauss.in` 中输入小数据的解 ($n=1$ 时, $n=2$ 时, ...)，以 `EOF` 结束。

依次运行 `gauss.epp`，`matrix.epp`，得到 `matrix.out`。

将 `matrix.out` 中的文件粘贴在 `main.epp` 中相应的位置中。注意模数一定要是质数。

```
1 //gauss.cpp
2 #include <bits/stdc++.h>
3 #define N 102
4 #define mod 1000000007
5 //caution: you can use this program iff mod is a prime.
6
7 using namespace std;
8
9 int n, m, k, a[N], g[N][N];
10
11 int power(int base, int times) {
12     int ret = 1;
13     while (times) {
14         if (times & 1) ret = 1ll * ret * base % mod;
15         base = 1ll * base * base % mod;
16         times >>= 1;
17     }
18     return ret;
19 }
20
21 int test() {
22     for (int i=0; i<m; i++) {
23         for (int j=i; j<=m; j++)
24             if (g[j][i]) {
25                 for (int k=i; k<=m; k++)
26                     swap(g[i][k], g[j][k]);
27                 break;
28             }
29     }
30     if (g[i][i] == 0)
31         return 0;
32     for (int j=i+1; j<n; j++) {
33         while (g[j][i]) {
```

```

33         int t = 111 * g[i][i] * power(g[j][i], mod - 2) % mod;
34         for (int k=i; k<n; k++)
35             g[i][k] = (g[i][k] + mod - (111 * t * g[j][k] % mod)) % mod;
36         for (int k=i; k<=m; k++)
37             swap(g[i][k], g[j][k]);
38     }
39 }
40 int t = power(g[i][i], mod - 2);
41 for (int j = 0; j <= m; ++j)
42     g[i][j] = 111 * g[i][j] * t % mod;
43 }
44 for (int i = m; i < n; ++i)
45     if (g[i][m]) return 0;
46 for (int i = m - 1; i >= 0; --i) {
47     int t = power(g[i][i], mod - 2);
48     g[i][i] = 1;
49     g[i][m] = 111 * g[i][m] * t % mod;
50     for (int j = 0; j < i; ++j)
51         g[j][m] = (g[j][m] + mod - 111 * g[i][m] * g[j][i] % mod) % mod;
52 }
53 printf("%d\n", m);
54 for (int i = 0; i < m; ++i)
55     printf("%d_", g[i][m]);
56 puts("");
57 for (int i = 0; i < m - 1; ++i)
58     printf("%d_", a[i]);
59 puts("1");
60 return 1;
61 }
62
63 int main() {
64     freopen("gauss.in", "r", stdin);
65     freopen("gauss.out", "w", stdout);
66     k = 0;
67     while (~scanf("%d", &a[k++])) ;
68     for (int sm = 1; sm <= k - sm; ++sm) {
69         n = k - sm - 1;
70         m = sm + 1;
71         for (int i = 0; i < n; ++i) {
72             for (int j = 0; j <= sm; ++j)
73                 g[i][j] = a[i + j];
74             g[i][m] = 1;
75             swap(g[i][m - 1], g[i][m]);
76         }
77         if (test()) return 0;
78     }
79     puts("no_solution");
80     return 0;
81 }

```

```

1 //matrix.cpp
2 #include <bits/stdc++.h>
3 #define N 102
4 using namespace std;
5
6 int n, a[N];
7
8 int main() {
9     freopen("gauss.out", "r", stdin);
10    freopen("matrix.out", "w", stdout);
11    scanf("%d", &n);
12    for (int i = 0; i < n; ++i) scanf("%d", &a[i]);
13    printf("#define_M_%d\n", n);
14    printf("const_int_trans[M][M]_=_{\n");
15    for (int i = 0; i < n; ++i) {
16        printf("\t{");
17        for (int j = 0; j < n; ++j) {
18            int t;
19            if (j < n - 2) t = i == j + 1;
20            else if (j == n - 2) t = a[i];
21            else t = i == n - 1;
22            printf("%s%d", j == 0 ? "" : ",_", t);
23        }
24        printf("}%s\n", i == n - 1 ? "" : ",");
25    }
26    printf("};\n");
27    printf("const_int_pref[M]_=_{");
28    for (int i = 0; i < n; ++i) {
29        int x;
30        scanf("%d", &x);
31        printf("%d%s", x, i == n - 1 ? "};\n" : ",_");
32    }
33    return 0;
34 }

```

```

1 //main.cpp
2 #include <bits/stdc++.h>
3 using namespace std;
4
5 /* paste matrix.out here. */
6
7 #define mod 1000000007
8
9 struct Matrix {
10     int c[M][M];
11     void clear() { memset(c, 0, sizeof(c)); }
12     void identity() { clear(); for (int i = 0; i < M; ++i) c[i][i] = 1; }
13     void base() { memcpy(c, trans, sizeof(trans)); }
14     friend Matrix operator * (const Matrix &a, const Matrix &b) {

```

```

15     Matrix c; c.clear();
16     for (int i = 0; i < M; ++i)
17         for (int j = 0; j < M; ++j)
18             for (int k = 0; k < M; ++k)
19                 c.c[i][j] = (c.c[i][j] + 1ll * a.c[i][k] * b.c[k][j] % mod) %
                                     mod;
20     return c;
21 }
22 } start, base;
23
24 Matrix power(Matrix base, int times) {
25     Matrix ret; ret.identity();
26     while (times) {
27         if (times & 1) ret = ret * base;
28         base = base * base;
29         times >>= 1;
30     }
31     return ret;
32 }
33
34 int main() {
35     int tot;
36     scanf("%d", &tot);
37     while (tot--) {
38         int n;
39         scanf("%d", &n);
40         start.clear();
41         for (int i = 0; i < M; ++i) start.c[0][i] = pref[i];
42         base.base();
43         base = power(base, n - 1);
44         start = start * base;
45         printf("%d\n", start.c[0][0]);
46     }
47     return 0;
48 }

```

8.2 分数类

```

1  #define LL long long
2
3  struct frac {
4      LL x, y;
5      frac(LL _x = 0, LL _y = 1) {
6          x = _x;
7          y = _y;
8          LL g = __gcd(abs(x), abs(y));
9          x /= g;
10         y /= g;
11         if (y < 0) {

```



```

12         x = -x;
13         y = -y;
14     }
15 }
16
17 inline friend frac operator + (const frac &lhs, const frac &rhs) {
18     return frac(lhs.x * rhs.y + rhs.x * lhs.y, lhs.y * rhs.y);
19 }
20
21 inline friend frac operator - (const frac &lhs, const frac &rhs) {
22     return frac(lhs.x * rhs.y - rhs.x * lhs.y, lhs.y * rhs.y);
23 }
24
25 inline friend frac operator - (const frac &lhs) {
26     return frac(-lhs.x, lhs.y);
27 }
28
29 inline friend frac operator * (const frac &lhs, const frac &rhs) {
30     return frac(lhs.x * rhs.x, lhs.y * rhs.y);
31 }
32
33 inline friend frac operator / (const frac &lhs, const frac &rhs) {
34     return frac(lhs.x * rhs.y, lhs.y * rhs.x);
35 }
36
37 inline friend bool operator == (const frac &lhs, const frac &rhs) {
38     return lhs.x * rhs.y == rhs.x * lhs.y;
39 }
40
41 inline friend bool operator != (const frac &lhs, const frac &rhs) {
42     return lhs.x * rhs.y != rhs.x * lhs.y;
43 }
44
45 inline friend bool operator < (const frac &lhs, const frac &rhs) {
46     return lhs.x * rhs.y < rhs.x * lhs.y;
47 }
48
49 inline friend bool operator > (const frac &lhs, const frac &rhs) {
50     return lhs.x * rhs.y > rhs.x * lhs.y;
51 }
52
53 inline friend bool operator <= (const frac &lhs, const frac &rhs) {
54     return lhs.x * rhs.y <= rhs.x * lhs.y;
55 }
56
57 inline friend bool operator >= (const frac &lhs, const frac &rhs) {
58     return lhs.x * rhs.y >= rhs.x * lhs.y;
59 }
60
61 inline void print() const {

```

```

62     printf("%lld/%lld\n", x, y);
63 }
64 };

```

8.3 取模整数类

如果需要用模意义下的除法，需定义常量 D 为除数的最大值，并执行 `init_inv()`。

```

1  struct mod;
2  mod* inv;
3
4  struct mod {
5      static constexpr int MOD = 1000 * 1000 * 1000 + 7; // std=c++11
6      mod(int x_) : x((x_ % MOD + MOD) % MOD) {}
7      mod() = default;
8      int x = 0;
9      inline mod operator *(mod other) const {
10         return ((long long)x * other.x) % MOD;
11     }
12     inline mod& operator *=(mod other) {
13         x = ((long long)x * other.x) % MOD;
14         return *this;
15     }
16     inline mod operator +(mod other) const {
17         int res = x + other.x;
18         if (res >= MOD) {
19             res -= MOD;
20         }
21         return res;
22     }
23     inline mod& operator +=(mod other) {
24         if ((x += other.x) >= MOD) {
25             x -= MOD;
26         }
27         return *this;
28     }
29     inline mod operator -(mod other) const {
30         int res = x - other.x;
31         if (res < 0) {
32             res += MOD;
33         }
34         return res;
35     }
36     inline mod& operator -=(mod other) {
37         if ((x -= other.x) < 0) {
38             x += MOD;
39         }
40         return *this;
41     }
42     inline mod operator /(mod other) const {

```

```

43     return (*this) * inv[other.x];
44 }
45 inline mod& operator /=(mod other) {
46     return *this *= inv[other.x];
47 }
48 inline bool operator ==(mod other) const {
49     return x == other.x;
50 }
51 inline mod operator -() const {
52     return x != 0 ? MOD - x : 0;
53 }
54 };
55
56 void init_inv() {
57     inv = new mod[D];
58     inv[1] = 1;
59     for (int i = 2; i < D; i++) {
60         inv[i] = (mod::MOD - (long long)mod::MOD / i * inv[mod::MOD % i].x % mod::
        MOD) % mod::MOD;
61     }
62 }

```

8.4 多项式类

```

1 struct poly {
2     vector<mod> C;
3     poly() {}
4     explicit poly(const vector<mod> &C_) : C(C_) {}
5     static const poly zero;
6     inline int deg() const {
7         return (int)C.size() - 1;
8     }
9     inline mod operator[](int x) const {
10         return (x < 0 || x > deg()) ? mod(0) : C[x];
11     }
12     inline mod& operator[](int x) {
13         if (x > deg()) {
14             C.resize(x + 1);
15         }
16         return C[x];
17     }
18     inline friend poly operator +(const poly& a, const poly& b) {
19         vector<mod> c(max(a.deg(), b.deg()) + 1);
20         for (int i = 0; i < c.size(); i++) {
21             c[i] = a[i] + b[i];
22         }
23         return poly(c);
24     }
25     inline friend poly operator -(const poly& a, const poly& b) {

```

```

26     vector<mod> c(max(a.deg(), b.deg()) + 1);
27     for (int i = 0; i < c.size(); i++) {
28         c[i] = a[i] - b[i];
29     }
30     return poly(c);
31 }
32 inline bool isZero() const {
33     return C.empty();
34 }
35 inline friend poly operator *(const poly& a, const poly& b) {
36     if (a.isZero() || b.isZero()) {
37         return zero;
38     }
39     vector<mod> c(1 + a.deg() + b.deg());
40     for (int i = 0; i <= a.deg(); i++) {
41         for (int j = 0; j <= b.deg(); j++) {
42             c[i + j] += a[i] * b[j];
43         }
44     }
45     return poly(c);
46 }
47 inline poly derivative() const {
48     if (isZero()) {
49         return zero;
50     }
51     vector<mod> res(deg());
52     for (int i = 0; i < deg(); i++) {
53         res[i] = C[i + 1] * (i + 1);
54     }
55     return poly(res);
56 }
57 inline poly primitive() const {
58     if (isZero()) {
59         return zero;
60     }
61     vector<mod> res(2 + deg());
62     for (int i = 1; i <= 1 + deg(); i++) {
63         res[i] = C[i - 1] / i;
64     }
65     return poly(res);
66 }
67 inline mod operator()(mod x) const {
68     mod res = 0;
69     for (int i = deg(); i >= 0; i--) {
70         res = res * x + C[i];
71     }
72     return res;
73 }
74 // Expand  $P(x+t)$ .
75 inline poly shift(int t) const {

```

```

76     poly res;
77     res[deg()];
78     vector<mod> binomial(deg() + 1, 0);
79     binomial[0] = 1;
80     for (int i = 0; i <= deg(); i++) {
81         mod cur = 1;
82         for (int j = i; j >= 0; j--) {
83             res[j] += C[i] * binomial[j] * cur;
84             cur *= t;
85         }
86         if (i == deg()) {
87             break;
88         }
89         for (int j = i + 1; j > 0; j--) {
90             binomial[j] += binomial[j - 1];
91         }
92     }
93     return res;
94 }
95 };

```

8.5 高精度计算

```

1  #include<algorithm>
2  using namespace std;
3  const int N_huge=850,base=1000000000;
4  char s[N_huge*10];
5  struct huge{
6      typedef long long value;
7      value a[N_huge];int len;
8      void clear(){len=1;a[len]=0;}
9      huge(){clear();}
10     huge(value x){*this=x;}
11     huge operator =(huge b){
12         len=b.len;for (int i=1;i<=len;++i)a[i]=b.a[i]; return *this;
13     }
14     huge operator =(value x){
15         len=0;
16         while (x)a[++len]=x%base,x/=base;
17         if (!len)a[++len]=0;
18         return *this;
19     }
20     huge operator +(huge b){
21         int L=len>b.len?len:b.len;huge tmp;
22         for (int i=1;i<=L+1;++i)tmp.a[i]=0;
23         for (int i=1;i<=L;++i){
24             if (i>len)tmp.a[i]+=b.a[i];
25             else if (i>b.len)tmp.a[i]+=a[i];
26             else {

```

```

27         tmp.a[i]+=a[i]+b.a[i];
28         if (tmp.a[i]>=base){
29             tmp.a[i]-=base;++tmp.a[i+1];
30         }
31     }
32 }
33 if (tmp.a[L+1])tmp.len=L+1;
34     else tmp.len=L;
35 return tmp;
36 }
37 huge operator -(huge b){
38     int L=len>b.len?len:b.len;huge tmp;
39     for (int i=1;i<=L+1;++i)tmp.a[i]=0;
40     for (int i=1;i<=L;++i){
41         if (i>b.len)b.a[i]=0;
42         tmp.a[i]+=a[i]-b.a[i];
43         if (tmp.a[i]<0){
44             tmp.a[i]+=base;--tmp.a[i+1];
45         }
46     }
47     while (L>1&&!tmp.a[L])--L;
48     tmp.len=L;
49     return tmp;
50 }
51 huge operator *(huge b){
52     int L=len+b.len;huge tmp;
53     for (int i=1;i<=L;++i)tmp.a[i]=0;
54     for (int i=1;i<=len;++i)
55         for (int j=1;j<=b.len;++j){
56             tmp.a[i+j-1]+=a[i]*b.a[j];
57             if (tmp.a[i+j-1]>=base){
58                 tmp.a[i+j]+=tmp.a[i+j-1]/base;
59                 tmp.a[i+j-1]%=base;
60             }
61         }
62     tmp.len=len+b.len;
63     while (tmp.len>1&&!tmp.a[tmp.len])--tmp.len;
64     return tmp;
65 }
66 pair<huge,huge> divide(huge a,huge b){
67     int L=a.len;huge c,d;
68     for (int i=L;i--i){
69         c.a[i]=0;d=d*base;d.a[1]=a.a[i];
70         int l=0,r=base-1,mid;
71         while (l<r){
72             mid=(l+r+1)>>1;
73             if (b*mid<=d)l=mid;
74             else r=mid-1;
75         }
76         c.a[i]=1;d-=b*1;

```

```

77     }
78     while (L>1&&!c.a[L])--L;c.len=L;
79     return make_pair(c,d);
80 }
81 huge operator / (value x){
82     value d=0;huge tmp;
83     for (int i=len;i;--i){
84         d=d*base+a[i];
85         tmp.a[i]=d/x;d%=x;
86     }
87     tmp.len=len;
88     while (tmp.len>1&&!tmp.a[tmp.len])--tmp.len;
89     return tmp;
90 }
91 value operator %(value x){
92     value d=0;
93     for (int i=len;i;--i)d=(d*base+a[i])%x;
94     return d;
95 }
96 huge operator / (huge b){return divide(*this,b).first;}
97 huge operator % (huge b){return divide(*this,b).second;}
98 huge &operator += (huge b) {*this=*this+b;return *this;}
99 huge &operator -= (huge b) {*this=*this-b;return *this;}
100 huge &operator *= (huge b) {*this=*this*b;return *this;}
101 huge &operator ++ () {huge T;T=1;*this=*this+T;return *this;}
102 huge &operator -- () {huge T;T=1;*this=*this-T;return *this;}
103 huge operator ++ (int) {huge T,tmp=*this;T=1;*this=*this+T;return tmp;}
104 huge operator -- (int) {huge T,tmp=*this;T=1;*this=*this-T;return tmp;}
105 huge operator + (value x) {huge T;T=x;return *this+T;}
106 huge operator - (value x) {huge T;T=x;return *this-T;}
107 huge operator * (value x) {huge T;T=x;return *this*T;}
108 huge operator *= (value x) {*this=*this*x;return *this;}
109 huge operator += (value x) {*this=*this+x;return *this;}
110 huge operator -= (value x) {*this=*this-x;return *this;}
111 huge operator /= (value x) {*this=*this/x;return *this;}
112 huge operator %=(value x) {*this=*this%x;return *this;}
113 bool operator ==(value x) {huge T;T=x;return *this==T;}
114 bool operator !=(value x) {huge T;T=x;return *this!=T;}
115 bool operator <=(value x) {huge T;T=x;return *this<=T;}
116 bool operator >=(value x) {huge T;T=x;return *this>=T;}
117 bool operator <(value x) {huge T;T=x;return *this<T;}
118 bool operator >(value x) {huge T;T=x;return *this>T;}
119 bool operator < (huge b) {
120     if (len<b.len)return 1;
121     if (len>b.len)return 0;
122     for (int i=len;i;--i){
123         if (a[i]<b.a[i])return 1;
124         if (a[i]>b.a[i])return 0;
125     }
126     return 0;

```

```

127     }
128     bool operator ==(huge b){
129         if (len!=b.len)return 0;
130         for (int i=len;i--i)
131             if (a[i]!=b.a[i])return 0;
132         return 1;
133     }
134     bool operator !=(huge b){return !(*this==b);}
135     bool operator >(huge b){return !(*this<b||*this==b);}
136     bool operator <=(huge b){return (*this<b)||(*this==b);}
137     bool operator >=(huge b){return (*this>b)||(*this==b);}
138     void str(char s[]){
139         int l=strlen(s);value x=0,y=1;len=0;
140         for (int i=l-1;i>=0;--i){
141             x=x+(s[i]-'0')*y;y*=10;
142             if (y==base)a[++len]=x,x=0,y=1;
143         }
144         if (!len||x)a[++len]=x;
145     }
146     void read(){
147         scanf("%s",s);this->str(s);
148     }
149     void print(){
150         printf("%d", (int)a[len]);
151         for (int i=len-1;i--i){
152             for (int j=base/10;j>=10;j/=10){
153                 if (a[i]<j)printf("0");
154                 else break;
155             }
156             printf("%d", (int)a[i]);
157         }
158         printf("\n");
159     }
160 }f[1005];
161 int main(){
162     f[1]=f[2]=1;
163     for(int i=3;i<=1000;i++)f[i]=f[i-1]+f[i-2];
164 }

```

8.6 读入优化

8.6.1 普通读入优化

```

1 #define rd RD<int>
2 #define rdll RD<long long>
3 template <typename Type>
4 inline Type RD() {
5     Type x = 0;
6     int flag = 0;

```



```

7   char c = getchar();
8   while (!isdigit(c) && c != '-')
9       c = getchar();
10  (c == '-') ? (flag = 1) : (x = c - '0');
11  while (isdigit(c = getchar()))
12      x = x * 10 + c - '0';
13  return flag ? -x : x;
14 }
15 inline char rdch() {
16     char c = getchar();
17     while (!isalpha(c)) c = getchar();
18     return c;
19 }

```

8.6.2 HDU 专用读入优化

接口：

int rd(int &x); 读入一个整数，保存在变量 x 中。如正常读入，返回值为 1，否则返回 EOF(-1)
int rdll(long long &x);

```

1  #define rd RD<int>
2  #define rdll RD<long long>
3
4  const int S = 2000000; // 2MB
5
6  char s[S], *h = s, *t = h;
7
8  inline char getchrr(void) {
9      if(h == t) {
10         if (t != s + S) return EOF;
11         t = s + fread(s, 1, S, stdin);
12         h = s;
13     }
14     return *h++;
15 }
16
17 template <class T>
18 inline int RD(T &x) {
19     char c = 0;
20     int sign = 0;
21     for (; !isdigit(c); c = getchrr()) {
22         if (c == EOF)
23             return -1;
24         if (c == '-')
25             sign ^= 1;
26     }
27     x = 0;
28     for (; isdigit(c); c = getchrr())
29         x = x * 10 + c - '0';
30     if (sign) x = -x;

```

```

31     return 1;
32 }

```

8.7 O2 优化

```

1 #define OPTIM __attribute__((optimize("-O2")))

```

8.8 位运算及其运用

8.8.1 枚举子集

枚举 i 的非空子集 j

```

1 for (int j = i; j; j = (j - 1) & i);

```

8.8.2 求 1 的个数

```

1 int __builtin_popcount(unsigned int x);

```

8.8.3 求前缀 0 的个数

```

1 int __builtin_clz(unsigned int x);

```

8.8.4 求后缀 0 的个数

```

1 int __builtin_ctz(unsigned int x);

```

9 Sublime Text

9.1 License

```

1  -- BEGIN LICENSE --
2  TwitterInc
3  200 User License
4  EA7E-890007
5  1D77F72E 390CDD93 4DCBA022 FAF60790
6  61AA12C0 A37081C5 D0316412 4584D136
7  94D7F7D4 95BC8C1C 527DA828 560BB037
8  D1EDDD8C AE7B379F 50C9D69D B35179EF
9  2FE898C4 8E4277A8 555CE714 E1FB0E43
10 D5D52613 C3D12E98 BC49967F 7652EED2
11 9D2D2E61 67610860 6D338B72 5CF95C69
12 E36B85CC 84991F19 7575D828 470A92AB
13 -- END LICENSE --

```

9.2 Preferences.sublime-settings

```
1 {  
2     "font_size": 13,  
3     "show_encoding": true,  
4     "update_check": false  
5 }
```

10 Vim

```
1 syntax on  
2 set cindent  
3 set nu  
4 set tabstop=4  
5 set shiftwidth=4  
6 set background=dark  
7  
8 inoremap <C-j> <down>  
9 inoremap <C-k> <up>  
10 inoremap <C-h> <left>  
11 inoremap <C-l> <right>
```