

## 一、常用

头文件  
快读  
对拍  
\_\_int128

## 二、字符串

kmp  
manacher  
最小表示法  
Z函数  
AC自动机  
SA(nlogn)  
SAIS  
SAM  
ExSAM  
PAM

## 三、图论

dinic  
二分图最大匹配  
2—SAT—Tarjan  
SCC hosoraju  
SCC Tarjan  
边双连通分量  
割点  
无向图欧拉图  
有向图欧拉图  
笛卡尔树  
dfs序求lca  
点分治

## 四、数论

exgcd  
整数分块  
欧拉筛（质数）  
欧拉筛(约数个数)  
欧拉筛（最小素因子）  
 $ax-by=1$ 的解  
pollard\_rho

## 五、数据结构

ST表  
树状数组  
并查集  
二维树状数组维护区间查询，修改  
线段树（区间查询最小值，最小值个数）  
线段树（区间修改加法，区间查询）  
线段树（区间修改加与乘，区间查询）  
DynamicSegmentTree  
pds

## 六、简单计算几何

点

## 七、杂项

矩阵快速幂  
组合数

## 八、python

一些基本数据结构  
math库  
快速幂  
并查集  
线段树区间加区间和  
字符串  
二维列表  
list  
常用函数

## 一、常用

### 头文件

```
1  /*
2
3  _/      _/      _/      _/      _/      _/      _/_/_/_/_/      _/_/      _/
4  _/
5  _/      _/      _/      _/      _/      _/      _/      _/      _/      _/_/
6  _/_/
7  _/_/      _/_/_/_/_/_/      _/      _/      _/      _/      _/
8  _/
9  _/      _/      _/      _/      _/      _/      _/_/      _/
10 _/
11 */
12 #pragma GCC optimize("unroll-loops")
13 #pragma GCC optimize("Ofast")
14 #include<bits/stdc++.h>
15 using namespace std;
16 typedef long long ll;
17 typedef unsigned long long ull;
18 #define rep(i,a,n) for(int i=a;i<n;i++)
19 #define per(i,a,n) for(int i=n-1;i>=a;i--)
20 #define fastio ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
21 #define multi int _;cin>>_;while(--)
22 #define debug(x) cerr << #x << " = " << (x) << endl;
23 #define int long long
24 #define pb push_back
25 #define eb emplace_back
26 ll gcd(ll a,ll b){ return b?gcd(b,a%b):a;}
27 mt19937 rand(random_device{}());
28 int rnd(int x){ return rand() % x; }
29 void test() {cerr << "\n";}
30 template<typename T, typename... Args>
```

```

31 void test(T x, Args... args) {cerr << x << " ";test(args...);}
32 const ll MOD = 998244353;
33 // const ll MOD = 1e9+7;
34 int ksm(int x,int y){int ans=1;x%=MOD;while(y)
    {if(y&1)ans=ans*x%MOD;x=x*x%MOD,y/=2;}return ans;}
35
36 const ll P1 = 999971, base1 = 101;
37 const ll P2 = 999973, base2 = 103;
38 const ll N = 200005;
39 //head
40
41
42 signed main()
43 {
44 #ifdef localfreopen
45     // freopen("1.in","r",stdin);
46 #endif
47     fastio
48
49     return 0;
50 }

```

## 快读

```

1 inline int read()
2 {
3     int x=0,f=1;char ch=getchar();
4     while (ch<'0' || ch>'9'){if (ch=='-') f=-1;ch=getchar();}
5     while (ch>='0'&&ch<='9'){x=x*10+ch-48;ch=getchar();}
6     return x*f;
7 }

```

## 对拍

```

1 :loop
2 data.exe > 1.in
3 my.exe <1.in >my.out
4 std.exe <1.in >std.out
5 fc my.out std.out
6 if not errorlevel 1 goto loop
7 pause
8 goto loop

```

## \_\_int128

```

1 __int128 read()
2 {
3     __int128 f=1,w=0;
4     char ch=getchar();
5     while(ch<'0' || ch>'9')
6     {

```

```

7         if(ch=='-')
8             f=-1;
9             ch=getchar();
10    }
11    while(ch<='9'&&ch>='0')
12    {
13        w=w*10+ch-'0';
14        ch=getchar();
15    }
16    return f*w;
17 }
18
19 void print(__int128 x)
20 {
21     if(x<0)
22     {
23         putchar('-');
24         x=-x;
25     }
26     if(x>9)print(x/10);
27     putchar(x%10+'0');
28 }

```

## 二、字符串

### kmp

```

1  vector<int> kmp(string s)
2  { //string的形式为 '#' + t1 + '#' + s
3      int n = s.size() - 1;
4      vector<int> nxt(s.size());
5      int j = 0;
6      for(int i = 2 ; i <= n ; i++){
7          while(j && s[j + 1] != s[i]) j = nxt[j];
8          if(s[j + 1] == s[i]) j++;
9          nxt[i] = j;
10     }
11     return nxt;
12 } //从第lent + 2 位 到 lent + lens + 1位为 s

```

### manacher

```

1  vector<int> manacher(string s)
2  { //string为 #A#B#C#...#Z#
3      int n = s.size();
4      vector<int> d1(n);
5      for (int i = 0, l = 0, r = -1; i < n; i++)
6      {
7          int k = (i > r) ? 1 : min(d1[l + r - i], r - i + 1);
8          while (0 <= i - k && i + k < n && s[i - k] == s[i + k]) k++;
9          d1[i] = k--;

```

```

10         if (i + k > r)
11         {
12             l = i - k;
13             r = i + k;
14         }
15     }
16     return d1;
17 }

```

## 最小表示法

```

1  string minrep(string s)
2  { //s从s[0]开始存
3      int k = 0, i = 0, j = 1, n = s.size();
4      while (k < n && i < n && j < n) {
5          if (s[(i + k) % n] == s[(j + k) % n]) {
6              k++;
7          } else {
8              s[(i + k) % n] > s[(j + k) % n] ? i = i + k + 1 : j = j + k + 1;
9              if (i == j) i++;
10             k = 0;
11         }
12     }
13     i = min(i, j);
14     return s.substr(i, N) + s.substr(0, i);
15 }

```

## Z函数

```

1  vector<int> exkmp(string s)
2  {
3      vector<int> p(s.size());
4      int n = s.size() - 1;
5      int L = 1, R = 0;
6      p[1] = 0;
7      for(int i = 2 ; i <= n ; i++)
8      {
9          if(i > R)
10             {
11                 p[i] = 0;
12             }else{
13                 int k = i - L + 1;
14                 p[i] = min(p[k], R - i + 1);
15             }
16             while(i + p[i] <= n && s[p[i] + 1] == s[i + p[i]])
17             {
18                 ++p[i];
19             }
20             if(i + p[i] - 1 > R)
21             {
22                 L = i;
23                 R = i + p[i] - 1;
24             }
25     }

```

```

26     return p;
27 } //从lent + 2位到lent + lens + 1位为 s
28 //*****p[1] = 0, 但实际从第一位往后能匹配lent的总长

```

## AC自动机

```

1  struct ACautomaton {
2      vector<vector<int>> nxt, end;
3      vector<int> fail;
4      int vtot = 0;
5      ACautomaton() : nxt(1, vector<int>(26, 0)), end(1), fail(1){
6
7      }
8      ACautomaton(vector<string> ss){
9          ACautomaton();
10         for (auto s : ss) {
11             insert(s);
12         }
13         buildfail();
14     }
15     int newnode() {
16         int cur = ++vtot;
17         nxt.push_back(vector<int>(26, 0));
18         end.push_back(vector<int>(0));
19         fail.emplace_back(0);
20         return cur;
21     }
22     void insert(string s, int id = 0) {
23         int now = 0;
24         for (auto c : s) {
25             int x = c - 'a';
26             if (!nxt[now][x]) {
27                 nxt[now][x] = newnode();
28             }
29             now = nxt[now][x];
30         }
31         end[now].emplace_back(id);
32     }
33     void buildfail() {
34         queue<int> q;
35         for (int i = 0; i <= 25; i++) {
36             if (nxt[0][i]) {
37                 fail[nxt[0][i]] = 0;
38                 q.push(nxt[0][i]);
39             }
40         }
41         while (!q.empty()) {
42             int now = q.front();
43             q.pop();
44             for (int i = 0; i <= 25; i++) {
45                 if (nxt[now][i]) {
46                     fail[nxt[now][i]] = nxt[fail[now]][i];
47                     q.push(nxt[now][i]);
48                 } else {
49                     nxt[now][i] = nxt[fail[now]][i];

```

```

50     }
51     }
52     }
53 }
54 int query(string s) {
55     int now = 0, ans = 0;
56     for (int i = 0; i < s.size(); i++) {
57         char c = s[i];
58         int x = c - 'a';
59         now = nxt[now][x];
60         ///自定义
61     }
62     return ans;
63 }
64 };// root = 0, ***记得buildfail

```

## SA(nlogn)

```

1 struct SA{
2     vector<int> sa, rk, oldrk, id, key1, cnt, ht;
3     vector<vector<int>> st;
4     int i, m = 127, p, w;
5     bool cmp(int x, int y, int w) {
6         return oldrk[x] == oldrk[y] && oldrk[x + w] == oldrk[y + w];
7     }// key1[i] = rk[id[i]] (作为基数排序的第一关键字数组)
8     int n;
9     SA(string s)
10    {
11        n = s.size() - 1;
12        oldrk.resize(2 * n + 5);
13        sa.resize(n + 2);
14        rk.resize(n + 2);
15        id.resize(n + 2);
16        key1.resize(n + 2);
17        cnt.resize(max(n + 5, 13011));
18        for (i = 1; i <= n; ++i) ++cnt[rk[i] = s[i]];
19        for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
20        for (i = n; i >= 1; --i) sa[cnt[rk[i]]--] = i;
21        for (w = 1;; w <= 1, m = p) { // m=p 就是优化计数排序值域
22            for (p = 0, i = n; i > n - w; --i) id[++p] = i;
23            for (i = 1; i <= n; ++i)
24                if (sa[i] > w) id[++p] = sa[i] - w;
25            fill(cnt.begin(), cnt.end(), 0);
26            for (i = 1; i <= n; ++i) ++cnt[key1[i] = rk[id[i]]];
27            // 注意这里px[i] != i, 因为rk没有更新, 是上一轮的排名数组
28
29            for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
30            for (i = n; i >= 1; --i) sa[cnt[key1[i]]--] = id[i];
31            for(int i = 1 ; i <= n ; i++)
32            {
33                oldrk[i] = rk[i];
34            }
35            for (p = 0, i = 1; i <= n; ++i)

```

```

36         rk[sa[i]] = cmp(sa[i], sa[i - 1], w) ? p : ++p;
37         if (p == n) {
38             break;
39         }
40     }
41     // height数组构建
42     ht.resize(n + 2);
43     int k = 0;
44     for(int i = 1 ; i <= n ; i++ )
45     {
46         k = max(k - 1, 0);
47         if(rk[i] == 1) continue;
48         int j = sa[rk[i] - 1];
49         while(s[i + k] == s[j + k]) k++;
50         ht[rk[i]] = k;
51     }
52
53     // LCPst表构建
54     st.resize(24);
55     st[0].resize(n + 5);
56     for(int i = 1 ; i <= n ; i++ )
57     {
58         st[0][i] = ht[i];
59     }
60     for(int j = 1 ; j <= 22 ; j++ )
61     {
62         st[j].resize(n + 5);
63         for(int i = 1 ; i + (1 << j) - 1 <= n ; i++ )
64         {
65             st[j][i] = min(st[j - 1][i], st[j - 1][i + (1 << j - 1)]);
66         }
67     }
68
69
70 }
71 int LCP(int u, int v)
72 {
73     if(u == v) return n - u + 1;
74     if(rk[u] > rk[v]) swap(u, v);
75     int l = rk[u] + 1, r = rk[v];
76     int len = __lg(r - l + 1);
77     return min(st[len][l], st[len][r - (1 << len) + 1]);
78 }
79 };
80 //字符串存在1~n
81 //如果要用vector<int>. 记得离散化

```

## SAIS

```

1 char str[1000010];
2 int n, a[2000100], sa[2000100], typ[2000100], c[1000100], p[2000100],
  sbuc[1000100], lbuc[1000100], name[1000100];
3 inline int islms(int *typ, int i)
4 {
5     return !typ[i] && (i == 1 || typ[i - 1]);

```



```

6  }
7  int cmp(int *s, int *typ, int p, int q)
8  {
9      do {
10         if (s[p] != s[q]) return 1;
11         p++; q++;
12     } while (!islms(typ, p) && !islms(typ, q));
13     return (!islms(typ, p) || !islms(typ, q) || s[p] != s[q]);
14 }
15
16 void isort(int *s, int *sa, int *typ, int *c, int n, int m)
17 {
18     int i;
19     for (lbuc[0] = sbuc[0] = c[0], i = 1; i <= m; i++) {
20         lbuc[i] = c[i - 1] + 1;
21         sbuc[i] = c[i];
22     }
23     for (i = 1; i <= n; i++)
24         if (sa[i]>1 && typ[sa[i] - 1])
25             sa[lbuc[sa[i] - 1]]++ = sa[i] - 1;
26     for (i = n; i >= 1; i--)
27         if (sa[i]>1 && !typ[sa[i] - 1])
28             sa[sbuc[sa[i] - 1]]-- = sa[i] - 1;
29 }
30
31 void build_sa(int *s, int *sa, int *typ, int *c, int *p, int n, int m)
32 {
33     int i;
34     for (i = 0; i <= m; i++) c[i] = 0;
35     for (i = 1; i <= n; i++) c[s[i]]++;
36     for (i = 1; i <= m; i++) c[i] += c[i - 1];
37     typ[n] = 0;
38     for (i = n - 1; i >= 1; i--)
39         if (s[i]<s[i + 1]) typ[i] = 0;
40         else if (s[i]>s[i + 1]) typ[i] = 1;
41         else typ[i] = typ[i + 1];
42     int cnt = 0;
43     for (i = 1; i <= n; i++)
44         if (!typ[i] && (i == 1 || typ[i - 1])) p[++cnt] = i;
45     for (i = 1; i <= n; i++) sa[i] = 0;
46     for (i = 0; i <= m; i++) sbuc[i] = c[i];
47     for (i = 1; i <= cnt; i++)
48         sa[sbuc[s[p[i]]]]-- = p[i];
49     isort(s, sa, typ, c, n, m);
50     int last = 0, t = -1, x;
51     for (i = 1; i <= n; i++)
52     {
53         x = sa[i];
54         if (!typ[x] && (x == 1 || typ[x - 1]))
55         {
56             if (!last || cmp(s, typ, x, last))
57                 name[x] = ++t;
58             else name[x] = t;
59             last = x;
60         }
61     }

```

```

62     for (i = 1; i <= cnt; i++)
63         s[n + i] = name[p[i]];
64     if (t < cnt - 1) build_sa(s + n, sa + n, typ + n, c + m + 1, p + n, cnt,
t);
65     else
66         for (i = 1; i <= cnt; i++)
67             sa[n + s[n + i] + 1] = i;
68     for (i = 0; i <= m; i++) sbuc[i] = c[i];
69     for (i = 1; i <= n; i++) sa[i] = 0;
70     for (i = cnt; i >= 1; i--)
71         sa[sbuc[s[p[sa[n + i]]]]--] = p[sa[n + i]];
72     isort(s, sa, typ, c, n, m);
73 }
74
75 int main()
76 {
77     scanf("%s", str);
78     n = strlen(str);
79     int i;
80     for (i = 1; i <= n; i++)
81         a[i] = str[i - 1];
82     a[++n] = 0;
83     build_sa(a, sa, typ, c, p, n, 200);
84     for (i = 2; i <= n; i++)
85         printf("%d%s", sa[i], i < n ? " " : "\n");
86     return 0;
87 }

```

## SAM

```

1  struct SuffixAutomaton
2  {
3      int tot, last;
4      vector<int> len, link, sz;
5      vector<vector<int>> nxt;
6      //vector<pii> order;
7      int n;
8      SuffixAutomaton(int _n) : n(_n), sz(2 * _n + 5), len(2 * _n + 5), link(2
* _n + 5), nxt(2 * _n + 5, vector<int>(33, 0))
9      {
10         len[1] = 0;
11         link[1] = -1;
12         nxt[1].clear();
13         nxt[1].resize(33);
14         tot = 2;
15         last = 1;
16     }
17     void extend(int c)
18     {
19         int cur = tot++, p;
20         len[cur] = len[last] + 1;
21         nxt[cur].clear();
22         nxt[cur].resize(33);

```

```

23     for (p = last; p != -1 && !nxt[p][c]; p = link[p])
24         nxt[p][c] = cur;
25     if (p == -1) link[cur] = 1;
26     else
27     {
28         int q = nxt[p][c];
29         if (len[p] + 1 == len[q]) link[cur] = q;
30         else
31         {
32             int clone = tot++;
33             len[clone] = len[p] + 1;
34             link[clone] = link[q];
35             nxt[clone] = nxt[q];
36             for (; p != -1 && nxt[p][c] == q; p = link[p])
37                 nxt[p][c] = clone;
38             link[q] = link[cur] = clone;
39         }
40     }
41     last = cur;
42     sz[cur] = 1;
43 }
44 vector<vector<int>> adj;
45 void buildLinkTree()
46 {
47     adj.resize(tot + 1);
48     for (int i = 2; i <= tot; i++)
49     {
50         adj[link[i]].push_back(i);
51     }
52 }
53 }; //sam的root为1

```

## ExSAM

```

1  struct EXSAM
2  {
3      const int CHAR_NUM = 30;    // 字符集个数，注意修改下方的 (-'a')
4      int tot;                    // 节点总数: [0, tot)
5      int n;
6      vector<int> len, link;
7      vector<vector<int>> nxt;
8      EXSAM (int _n) : n(_n), len(_n * 2 + 5), link(_n * 2 + 5), nxt(n * 2 +
9      5, vector<int>(CHAR_NUM + 1, 0))
10     {
11         tot = 2;
12         link[1] = -1;
13     }
14     int insertSAM(int last, int c)    // last 为父 c 为子
15     {
16         int cur = nxt[last][c];
17         if (!len[cur]) return cur;
18         len[cur] = len[last] + 1;
19         int p = link[last];
20         while (p != -1)

```

```

21         if (!nxt[p][c])
22             nxt[p][c] = cur;
23         else
24             break;
25         p = link[p];
26     }
27     if (p == -1)
28     {
29         link[cur] = 1;
30         return cur;
31     }
32     int q = nxt[p][c];
33     if (len[p] + 1 == len[q])
34     {
35         link[cur] = q;
36         return cur;
37     }
38     int clone = tot++;
39     for (int i = 0; i < CHAR_NUM; ++i)
40         nxt[clone][i] = len[nxt[q][i]] != 0 ? nxt[q][i] : 0;
41     len[clone] = len[p] + 1;
42     while (p != -1 && nxt[p][c] == q)
43     {
44         nxt[p][c] = clone;
45         p = link[p];
46     }
47     link[clone] = link[q];
48     link[cur] = clone;
49     link[q] = clone;
50     return cur;
51 }
52
53 int insertTrie(int cur, int c)
54 {
55     if (nxt[cur][c]) return nxt[cur][c]; // 已有该节点 直接返回
56     return nxt[cur][c] = tot++; // 无该节点 建立节点
57 }
58
59 void insert(const string &s)
60 {
61     int root = 1;
62     for (auto ch : s) root = insertTrie(root, ch - 'a');
63 }
64
65 void insert(const char *s, int n)
66 {
67     int root = 1;
68     for (int i = 0; i < n; ++i)
69         root =
70             insertTrie(root, s[i] - 'a'); // 一边插入一边更改所插入新节点的父
节点
71 }
72
73 void build()
74 {
75     queue<pair<int, int>> q;

```

```

76     for (int i = 0; i < 26; ++i)
77         if (nxt[1][i]) q.push({i, 1});
78     while (!q.empty())    // 广搜遍历
79     {
80         auto item = q.front();
81         q.pop();
82         auto last = insertSAM(item.second, item.first);
83         for (int i = 0; i < 26; ++i)
84             if (nxt[last][i]) q.push({i, last});
85     }
86 }
87 };

```

## PAM

```

1  const int N = 5e5 + 10, Sigma = 26;
2  char s[N];
3  int lastans, n;
4  struct Palindrome_Automaton {
5      int ch[N][Sigma], fail[N], len[N], sum[N], cnt, last;
6      Palindrome_Automaton() {
7          cnt = 1;
8          fail[0] = 1, fail[1] = 1, len[1] = -1;
9      }
10     int getfail(int x, int i) {
11         while(i - len[x] - 1 < 0 || s[i - len[x] - 1] != s[i]) x = fail[x];
12         return x;
13     }
14     void insert(char c, int i) {
15         int x = getfail(last, i), w = c - 'a';
16         if(!ch[x][w]) {
17             len[++cnt] = len[x] + 2;
18             int tmp = getfail(fail[x], i);
19             fail[cnt] = ch[tmp][w];
20             sum[cnt] = sum[fail[cnt]] + 1;
21             ch[x][w] = cnt;
22         }
23         last = ch[x][w];
24     }
25 } PAM;

```

## 三、图论

### dinic

```

1  const int v = 1010;
2  const int E = 101000;
3  using ll = long long;
4
5  template<typename T>
6  struct MaxFlow

```

```

7 {
8     int s, t, vtot;
9     int head[V], etot;
10    int dis[V], cur[V];
11    struct edge
12    {
13        int v, nxt;
14        T f;
15    }e[E * 2];
16    void addedge(int u, int v, T f)
17    {
18        e[etot] = {v, head[u], f}; head[u] = etot++;
19        e[etot] = {u, head[v], 0}; head[v] = etot++;
20    }
21    bool bfs()
22    {
23        for(int i = 1 ; i <= vtot ; i++ )
24        {
25            dis[i] = 0;
26            cur[i] = head[i];
27        }
28        queue<int> q;
29        q.push(s); dis[s] = 1;
30        while(!q.empty())
31        {
32            int u = q.front(); q.pop();
33            for(int i = head[u] ; ~i ; i = e[i].nxt)
34            {
35                if(e[i].f && !dis[e[i].v])
36                {
37                    int v = e[i].v;
38                    dis[v] = dis[u] + 1;
39                    if(v == t) return true;
40                    q.push(v);
41                }
42            }
43        }
44        return false;
45    }
46    T dfs(int u, T m)
47    {
48        if(u == t) return m;
49        T flow = 0;
50        for(int i = cur[u] ; ~i ; cur[u] = i = e[i].nxt)
51        {
52            if(e[i].f && dis[e[i].v] == dis[u] + 1)
53            {
54                T f = dfs(e[i].v, min(m, e[i].f));
55                e[i].f -= f;
56                e[i ^ 1].f += f;
57                m -= f;
58                flow += f;
59                if(!m) break;
60            }
61        }
62        if(!flow) dis[u] = -1;

```

```

63         return flow;
64     }
65     T dinic()
66     {
67         T flow = 0;
68         while(bfs()) flow += dfs(s, numeric_limits<T>::max());
69         return flow;
70     }
71     void init(int s_, int t_, int vtot_)
72     {
73         s = s_;
74         t = t_;
75         vtot = vtot_;
76         etot = 0;
77         for(int i = 1 ; i <= vtot ; i++)
78         {
79             head[i] = -1;
80         }
81     }
82 };
83
84 MaxFlow<ll> g;
85 /**记得每次init,
86

```

## 二分图最大匹配

```

1  int a[N];
2  int v[N], n1, n2;
3  int to[N], b[N];
4  int n;
5  vector<int> e[N];
6  //n1为左边点数量, n2为右边点数量, v为右边的点连向左边哪条边
7  bool find(int x)
8  {
9      b[x] = true;
10     for(auto y : e[x])
11     {
12         if(!v[y] || (!b[v[y]] && find(v[y])))
13         {
14             v[y] = x;
15             return true;
16         }
17     }
18     return false;
19 }
20
21 int match()
22 {
23     int ans = 0;
24     memset(v, 0, sizeof(v));
25     for(int i = 1 ; i <= n1 ; i++)
26     {
27         memset(b, 0, sizeof(b));
28         if(find(i))

```

```

29     {
30         ++ans;
31     }
32 }
33 return ans;
34 }

```

## 2—SAT—Tarjan

```

1  vector<int> e[N];
2  int dfn[N], ins[N], low[N], bel[N], idx, cnt;
3  stack<int> st;
4  vector<vector<int> > scc;
5
6
7  void dfs(int u)
8  {
9      dfn[u] = low[u] = ++idx;
10     ins[u] = true;
11     st.push(u);
12     for(auto v : e[u])
13     {
14         if(!dfn[v])
15         {
16             dfs(v);
17             low[u] = min(low[u], low[v]);
18         }else{
19             if(ins[v]) low[u] = min(low[u], dfn[v]);
20         }
21     }
22     if(dfn[u] == low[u])
23     {
24         vector<int> c;
25         ++cnt;
26         while(true)
27         {
28             int v = st.top();
29             c.push_back(v);
30             ins[v] = false;
31             bel[v] = cnt;
32             st.pop();
33             if(v == u) break;
34         }
35         sort(c.begin(), c.end());
36         scc.push_back(c);
37     }
38 }
39
40 int main()
41 {
42     fastio
43     //freopen("1.in", "r", stdin);
44     int n, m;
45     cin >> n >> m;
46     for(int i = 1 ; i <= m ; i++ )

```



```

47     {
48         int u, ch1, v, ch2;
49         cin >> u >> ch1 >> v >> ch2;
50         u = 2 * u + (ch1 == 0);
51         v = 2 * v + (ch2 == 0);
52         e[u ^ 1].push_back(v);
53         e[v ^ 1].push_back(u);
54     }
55     for(int i = 1 ; i <= 2 * n ; i++ )
56     {
57         if(!dfn[i]) dfs(i);
58     }
59     for(int i = 1 ; i <= n ; i++ )
60     {
61         if(bel[2 * i] == bel[2 * i + 1])
62         {
63             cout << "IMPOSSIBLE\n";
64             return 0;
65         }
66     }
67     cout << "POSSIBLE\n";
68     for(int i = 1 ; i <= n ; i++ )
69     {
70         cout << (bel[2 * i] < bel[2 * i + 1]) << " ";
71     }
72     cout << endl;
73     return 0;
74 }

```

## SCC hosoraju

```

1  int vis[N], n, m;
2  vector<int> out, c, e[N], erev[N];
3  int sz[N];
4  int bel[N], cnt;
5  vector<vector<int> >scc;
6
7  void dfs1(int u)
8  {
9      vis[u] = 1;
10     for(auto v : e[u])
11     {
12         if(!vis[v]) dfs1(v);
13     }
14     out.push_back(u);
15 }
16
17 void dfs2(int u, int cnt)
18 {
19
20     vis[u] = 1;
21     for(auto v : erev[u])
22     {
23         if(!vis[v]) dfs2(v, cnt);
24     }

```

```

25     bel[u] = cnt;
26     sz[cnt]++;
27     c.push_back(u);
28 }
29
30 int main()
31 {
32     fastio
33     //freopen("1.in", "r", stdin);
34     int n, m, x, y;
35     cin >> n >> m;
36     for(int i = 1 ; i <= m ; i++ )
37     {
38         cin >> x >> y;
39         e[x].push_back(y);
40         erev[y].push_back(x);
41     }
42     memset(vis, 0, sizeof(vis));
43     for(int i = 1 ; i <= n ; i++ )
44     {
45         if(!vis[i])
46         {
47             dfs1(i);
48         }
49     }
50     reverse(out.begin(), out.end());
51     memset(vis, 0, sizeof(vis));
52     for(auto u : out)
53     {
54         if(!vis[u])
55         {
56             c.clear();
57             dfs2(u, ++cnt);
58             sort(c.begin(), c.end());
59             scc.push_back(c);
60         }
61     }
62     sort(scc.begin(), scc.end());
63     for(auto c : scc)
64     {
65         for(auto x : c)
66         {
67             cout << x << " ";
68         }
69         cout << "\n";
70     }
71     return 0;
72 }
73 }

```

## SCC Tarjan

```

1  vector<int> e[N];
2  int dfn[N], ins[N], low[N], bel[N], idx, cnt;
3  stack<int> st;

```

```

4  vector<vector<int> > scc;
5
6
7  void dfs(int u)
8  {
9      dfn[u] = low[u] = ++idx;
10     ins[u] = true;
11     st.push(u);
12     for(auto v : e[u])
13     {
14         if(!dfn[v])
15         {
16             dfs(v);
17             low[u] = min(low[u], low[v]);
18         }else{
19             if(ins[v]) low[u] = min(low[u], dfn[v]);
20         }
21     }
22     if(dfn[u] == low[u])
23     {
24         vector<int> c;
25         ++cnt;
26         while(true)
27         {
28             int v = st.top();
29             c.push_back(v);
30             ins[v] = false;
31             bel[v] = cnt;
32             st.pop();
33             //cout << v << " ";
34             if(v == u) break;
35         }
36         //cout << endl;
37         sort(c.begin(), c.end());
38         scc.push_back(c);
39     }
40
41 }
42
43 int main()
44 {
45     fastio
46     //freopen("1.in", "r", stdin);
47     int n, m;
48     cin >> n >> m;
49     for(int i = 1 ; i <= m ; i++ )
50     {
51         int x, y;
52         cin >> x >> y;
53         e[x].push_back(y);
54     }
55     for(int i = 1 ; i <= n ; i++ )
56     {
57         if(!dfn[i])
58         {
59             dfs(i);

```

```

60     }
61 }
62 sort(scc.begin(), scc.end());
63 for(auto c : scc)
64 {
65     for(auto x : c)
66     {
67         cout << x << " ";
68     }
69     cout << "\n";
70 }
71 return 0;
72 }

```

## 边双连通分量

```

1  int head[N], e[N], nxt[N], idx = 1, n, m;
2  int dfn[M], low[M], cnt, b[N], bel[N], anscnt[M];
3  vector<vector<int> > dcc;
4  void add(int x, int y)
5  {
6      nxt[++idx] = head[x];
7      head[x] = idx;
8      e[idx] = y;
9  }
10 void tarjan(int x, int e_in)
11 {
12     dfn[x] = low[x] = ++cnt;
13     for(int i = head[x] ; i ; i = nxt[i])
14     {
15         int y = e[i];
16         if(!dfn[y])
17         {
18             tarjan(y, i);
19             if(dfn[x] < low[y])
20             {
21                 b[i] = b[i ^ 1] = 1;
22             }
23             low[x] = min(low[x], low[y]);
24         } else if (i != (e_in ^ 1))
25         {
26             low[x] = min(low[x], dfn[y]);
27         }
28     }
29 }
30
31 vector<int> v;
32
33 void dfs(int x, int cnt)
34 {
35     bel[x] = cnt;
36     v.push_back(x);
37     anscnt[cnt]++;
38     for(int i = head[x] ; i ; i = nxt[i])
39     {

```

```

40     int y = e[i];
41     if(bel[y] || b[i]) continue;
42     dfs(y, cnt);
43 }
44
45 }
46 signed main()
47 {
48     fastio
49     //freopen("1.in", "r", stdin);
50     cin >> n >> m;
51     int x, y;
52     for(int i = 1 ; i <= m ; i++ )
53     {
54         cin >> x >> y;
55         if(x == y) continue;
56         add(x, y);
57         add(y, x);
58     }
59     for(int i = 1 ; i <= n ; i++ )
60     {
61         if(!dfn[i]) tarjan(i, 0);
62     }
63     int ans = 0;
64     for(int i = 1 ; i <= n ; i++ )
65     {
66         if(!bel[i])
67         {
68             v.clear();
69             dfs(i, ++ans);
70             dcc.push_back(v);
71         }
72     }
73     int sz = dcc.size();
74     cout << dcc.size() << "\n";
75     for(int i = 0 ; i < sz ; i++ )
76     {
77         auto v = dcc[i];
78         cout << ansct[i + 1] << " ";
79         for(auto x : v)
80         {
81             cout << x << " ";
82         }
83         cout << "\n";
84     }
85     return 0;
86 }
87 }

```

## 割点

```

1  int n, m;
2  int dfn[N], idx, low[N];
3  bool vis[N], cut[N];
4  vector<int> e[N];

```

```

5  int cnt;
6
7  void dfs(int u, int root)
8  {
9      vis[u] = 1;
10     dfn[u] = ++idx;
11     low[u] = idx;
12     int child = 0;
13     for(auto v : e[u])
14     {
15         if(!vis[v])
16         {
17             dfs(v, root);
18             low[u] = min(low[u], low[v]);
19             if(low[v] >= dfn[u] && u != root)
20             {
21                 cut[u] = 1;
22             }
23             if(u == root)
24             {
25                 child++;
26             }
27         }
28         low[u] = min(low[u], dfn[v]);
29     }
30     if(child >= 2 && u == root)
31     {
32         cut[u] = 1;
33     }
34 }
35
36 int main()
37 {
38     fastio
39     //freopen("1.in","r",stdin);
40     cin >> n >> m;
41     rep(i, 1, m + 1)
42     {
43         int x, y;
44         cin >> x >> y;
45         e[x].push_back(y);
46         e[y].push_back(x);
47     }
48     rep(i, 1, n + 1)
49     {
50         if(!vis[i])
51         {
52             dfs(i, i);
53         }
54     }
55     cout << accumulate(cut + 1, cut + n + 1, 0ll) << "\n";
56     rep(i, 1, n + 1)
57     {
58         if(cut[i])
59         {
60             cout << i << " ";

```

```

61     }
62 }
63 return 0;
64 }

```

## 无向图欧拉图

```

1  vector<pair<int ,int > > e[N];
2  int d[N], n, m;
3  int f[N], b[N], sz[N], ans[N], idxans;
4
5  void dfs(int x)
6  {
7      //cout << "dfs = " << x << endl;
8      for(; f[x] < sz[x] ; )
9      {
10         int y = e[x][f[x]].first, id = e[x][f[x]].second;
11         if(!b[id])
12         {
13             b[id] = 1;
14             f[x]++;
15             dfs(y);
16             ans[++idxans] = y;
17         }else{
18             f[x]++;
19         }
20     }
21 }
22
23 void Euler()
24 {
25     memset(f, 0, sizeof(f));
26     memset(b, 0 ,sizeof(b));
27     int cnt = 0, x = 0;
28     for(int i = 1 ; i <= n ; i++ )
29     {
30         if(d[i] & 1)
31         {
32             cnt++;
33             x = i;
34         }
35     }
36     if(!(cnt == 0 || cnt == 2))
37     {
38         cout << "No\n";
39         return;
40     }
41     for(int i = 1 ; i <= n ; i++ )
42     {
43         sz[i] = e[i].size();
44         if(!x)
45             if(d[i])
46             {
47                 x = i;
48             }

```

```

49     }
50     dfs(x);
51     ans[++idxans] = x;
52     if(idxans == m + 1)
53     {
54         cout << "Yes\n";
55     }else{
56         cout << "No\n";
57     }
58 }
59 int main()
60 {
61     fastio
62     //freopen("1.in","r",stdin);
63     cin >> n >> m;
64     int idx = 0;
65     for(int i = 1 ; i <= m ; i++ )
66     {
67         int x, y;
68         cin >> x >> y;
69         ++idx;
70         ++d[x];
71         ++d[y];
72         e[x].push_back({y, idx});
73         e[y].push_back({x, idx});
74     }
75     Euler();
76     return 0;
77 }
78 }

```

## 有向图欧拉图

```

1  int n;
2  vector<int> e[N];
3  int ind[N], outd[N], f[N], sz[N], ans[N], idx = 0;
4
5  void dfs(int x)
6  {
7      for(; f[x] < sz[x] ;)
8      {
9          int y = e[x][f[x]];
10         f[x]++;
11         dfs(y);
12         ans[++idx] = y;
13     }
14 }
15 void Euler()
16 {
17     memset(f, 0, sizeof(f));
18     int cntdiff = 0;
19     int cntin = 0;
20     int x = 0;
21     for(int i = 1 ; i <= n ; i++ )
22     {

```



```

23         if(ind[i] != outd[i])
24         {
25             cntdiff++;
26         }
27         if(ind[i] + 1 == outd[i])
28         {
29             cntin++;
30             x = i;
31         }
32     }
33     if(!(cntdiff == 2 && cntin == 1 || cntdiff == 0))
34     {
35         cout << "No\n";
36         return;
37     }
38     for(int i = 1 ; i <= n ; i++ )
39     {
40         sz[i] = e[i].size();
41         //cout << e[i].size();
42         if(!x)
43         {
44             if(ind[i])
45             {
46                 x = i;
47             }
48         }
49     }
50     dfs(x);
51     ans[++idx]= x;
52     if(idx == n + 1)
53     {
54         cout << "Yes\n";
55     }else{
56         cout << "No\n";
57     }
58     for(int i = idx ; i > 0 ; i--)
59     {
60         cout << ans[i] << " ";
61     }
62 }

```

## 笛卡尔树

```

1 //每个父节点都小于其所有子节点
2
3 int a[N], n, l[N], r[N];
4 int root = 0;
5
6 void build()
7 {
8     stack<int> st;
9     for(int i = 1 ; i <= n ; i++ )
10    {

```

```

11     int last = 0;
12     while(!st.empty() && a[st.top()] > a[i])
13     {
14         last = st.top();
15         st.pop();
16     }
17     if(!st.empty())
18     {
19         r[st.top()] = i;
20     }else{
21         root = i;
22     }
23     l[i] = last;
24     st.push(i);
25 }
26 }

```

## dfs序求lca

```

1  int main()
2  {
3      int idx = 0;
4      vector<int> dfn(n + 5);
5      vector st(__lg(n) + 2, vector<int> (n + 5)); //****不能改成23****
6      function<int(int,int)> get = [&](int x, int y)
7      {
8          return dfn[x] < dfn[y] ? x : y;
9      };
10     function<void(int,int)> dfs = [&](int x, int fa)
11     {
12         st[0][dfn[x] = ++idx] = fa;
13         for(int y : adj[x]) if(y != fa) dfs(y, x);
14     };
15     function<int(int,int)> lca = [&](int u, int v)
16     {
17         if(u == v) return u;
18         if((u = dfn[u]) > (v = dfn[v])) swap(u, v);
19         int d = __lg(v - u++);
20         return get(st[d][u], st[d][v - (1 << d) + 1]);
21     };
22     dfs(s, 0);
23     for(int i = 1 ; i <= __lg(n) ; i++ ) //****不能改成23****
24     {
25         for(int j = 1 ; j + (1 << i - 1) <= n ; j++ ) // ****注意边界****
26         {
27             st[i][j] = get(st[i - 1][j], st[i - 1][j + (1 << i - 1)]);
28         }
29     }
30     /// lca(u, v);
31 }

```

## 点分治

```

1  signed main()

```

```

2 {
3     fastio
4     int n, k, ans = 0;
5     cin >> n >> k;
6     ans = n + 1;
7     vector<vector<pair<int,int>>> adj(n + 1);
8     vector<int> sz(n + 1, 0), maxsz(n + 1, 0), del(n + 1, 0);
9     vector<int> mark(k + 1, 0), c(k + 1, 0);
10    int T = 1;
11    int u, v, w;
12    for(int i = 1 ; i < n ; i++ )
13    {
14        cin >> u >> v >> w;
15        u++;
16        v++;
17        adj[u].emplace_back(v, w);
18        adj[v].emplace_back(u, w);
19    }
20    function<void(int, int)> solve = [&](int x, int s)
21    {
22        T++;
23        int mxs = s + 1, root = -1;
24        function<void(int, int)> dfs1 = [&](int x, int fx)
25        {
26            sz[x] = 1;
27            maxsz[x] = 0;
28            for(auto [y, w] : adj[x])
29            {
30                if(del[y] || y == fx) continue;
31                dfs1(y, x);
32                sz[x] += sz[y];
33                maxsz[x] = max(maxsz[x], sz[y]);
34            }
35            maxsz[x] = max(maxsz[x], s - sz[x]);
36            if(maxsz[x] < mxs)
37            {
38                mxs = maxsz[x], root = x;
39            }
40        };
41        dfs1(x, -1);
42        //////////////////////////////////////
43        mark[0] = T;
44        c[0] = 0;
45        for(auto [y, w] : adj[root])
46        {
47            if(del[y]) continue;
48            vector<pair<int, int>> self;
49            function<void(int, int, int, int)> dfs2 = [&](int x, int fx, int
dis, int dep)
50            {
51                self.emplace_back(dis, dep);
52                for(auto [y, w] : adj[x])
53                {
54                    if(del[y] || y == fx) continue;
55                    dfs2(y, x, dis + w, dep + 1);
56                }

```

```

57         };
58         dfs2(y, root, w, 1);
59         for(auto [dis, dep] : self)
60         {
61             if(k - dis >= 0 && mark[k - dis] == T)
62             {
63                 ans = min(ans, c[k - dis] + dep);
64             }
65         }
66         for(auto [dis, dep] : self)
67         {
68             if(dis > k) continue;
69             if(mark[dis] == T)
70             {
71                 c[dis] = min(c[dis], dep);
72             }else{
73                 c[dis] = dep;
74                 mark[dis] = T;
75             }
76         }
77     }
78     //////////////////////////////////
79     del[root] = 1;
80     for(auto [y, w] : adj[root])
81     {
82         if(del[y]) continue;
83         solve(y, sz[y]);
84     }
85 };
86 solve(1, n);
87 cout << (ans > n ? -1 : ans) << "\n";
88 return 0;
89 }

```

## 四、数论

### exgcd

```

1  int exgcd(int a, int b, int &x, int &y)
2  {
3      if(b == 0)
4      {
5          x = 1;
6          y = 0;
7          return a;
8      }
9      int d = exgcd(b, a % b, y, x);
10     y -= (a / b) * x;
11     return d;
12 }

```

## 整数分块

```
1  for(ll l = 1 ; l <= n ; l++ )
2      {
3          ll d = n / l, r = n / d;
4          cout << l << " : " << r << " = " << d << endl;
5          l = r;
6      }
```

## 欧拉筛（质数）

```
1  const ll MAXN = 1e6 + 5;
2  ll prime[MAXN], idxprime = 0;
3  bool isprime[MAXN];
4
5  void prime_build()
6  {
7      for(int i = 2 ; i < MAXN ; i++ )
8          {
9              if(isprime[i] == 0)
10                 {
11                     prime[++idxprime] = i;
12                 }
13                 for(int j = 1 ; j <= idxprime && i * prime[j] < MAXN ; j++ )
14                     {
15                         isprime[i * prime[j]] = 1;
16                         if(i % prime[j] == 0) break;
17                     }
18             }
19 }
```

## 欧拉筛(约数个数)

```
1  ll prim[50000005], sum[50000005], d[50000005], len;
2  bool vis[50000005];
3
4  inline void sieve(int x) {
5      for(int i = 2; i <= x; i++) {
6          if(! vis[i]) {
7              prim[++len] = i;
8              d[i] = 2;
9              sum[i] = 1;
10          }
11          for(int j = 1; j <= len && i * prim[j] <= x; j++) {
12              vis[i * prim[j]] = 1;
13              if(i % prim[j] == 0) {
14                  sum[i * prim[j]] = sum[i] + 1;
15                  d[i * prim[j]] = d[i] / (sum[i] + 1) * (sum[i] + 2);
16                  break;
17              }
18              sum[i * prim[j]] = 1;
19              d[i * prim[j]] = d[i] * 2;
20          }
21      }
```

```

21     }
22 }
23

```

## 欧拉筛（最小素因子）

```

1  int MAXN = 50;
2  int p[N], pr[N], idx;
3
4  void build()
5  {
6      for(int i = 2 ; i < MAXN ; i++ )
7      {
8          if(!p[i])
9          {
10             p[i] = i;
11             pr[++idx] = i;
12         }
13         for(int j = 1 ; j <= idx && pr[j] * i < MAXN ; j++ )
14         {
15             p[i * pr[j]] = pr[j];
16             if(p[i] == pr[j]) break;
17         }
18     }
19 }

```

## $ax-by=1$ 的解

```

1  ll exgcd(ll a, ll b, ll &x, ll &y)
2  {
3      if(b == 0)
4      {
5          x = 1;
6          y = 0;
7          return a;
8      }
9      int d = exgcd(b, a % b, y, x);
10     y -= (a / b) * x;
11     return d;
12 }
13
14 void solve()
15 {
16     ll a, b;
17     cin >> a >> b;
18     ll x, y;
19     ll d = exgcd(a, b, x, y);
20     y = -y;
21     while(x < 0 || y < 0)
22     {
23         x += b/d;
24         y += a/d;
25     }
26     while(x >= b/d && y >= a/d)

```

```

27     {
28         x -= b/d;
29         y -= a/d;
30     }
31     cout << x << " " << y << "\n";
32 }

```

## pollard\_rho

```

1  using i64 = long long;
2  using i128 = __int128;
3  i64 power(i64 a, i64 b, i64 m) {
4      i64 res = 1;
5      for (; b >= 1, a = i128(a) * a % m) {
6          if (b & 1) {
7              res = i128(res) * a % m;
8          }
9      }
10     return res;
11 }
12
13 bool isprime(i64 p) {
14     if (p < 2) {
15         return 0;
16     }
17     i64 d = p - 1, r = 0;
18     while (!(d & 1)) {
19         r++;
20         d >>= 1;
21     }
22     int prime[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
23     for (auto a : prime) {
24         if (p == a) {
25             return true;
26         }
27         i64 x = power(a, d, p);
28         if (x == 1 || x == p - 1) {
29             continue;
30         }
31         for (int i = 0; i < r - 1; i++) {
32             x = i128(x) * x % p;
33             if (x == p - 1) {
34                 break;
35             }
36         }
37         if (x != p - 1) {
38             return false;
39         }
40     }
41     return true;
42 }
43
44 mt19937 rng((unsigned int)
45 chrono::steady_clock::now().time_since_epoch().count());

```

```

46 i64 pollard_rho(i64 x) {
47     i64 s = 0, t = 0;
48     i64 c = i64(rng()) % (x - 1) + 1;
49     i64 val = 1;
50     for (int goal = 1; ; goal <= 1, s = t, val = 1) {
51         for (int step = 1; step <= goal; step++) {
52             t = (i128(t) * t + c) % x;
53             val = i128(val) * abs(t - s) % x;
54             if (step % 127 == 0) {
55                 i64 g = gcd(val, x);
56                 if (g > 1) {
57                     return g;
58                 }
59             }
60         }
61         i64 g = gcd(val, x);
62         if (g > 1) {
63             return g;
64         }
65     }
66 }
67
68 unordered_map<i64, int> getprimes(i64 x) {
69     unordered_map<i64, int> p;
70     function<void(i64)> get = [&](i64 x) {
71         if (x < 2) {
72             return;
73         }
74         if (isprime(x)) {
75             p[x]++;
76             return;
77         }
78         i64 mx = pollard_rho(x);
79         get(x / mx);
80         get(mx);
81     };
82     get(x);
83     return p;
84 }
85

```

## 五、数据结构

### ST表

```

1  for(int i = 1 ; i <= n ; i++ )
2  {
3      a[i] = read();
4      f[0][i] = a[i];
5  }
6  for(int i = 1 ; i <= 22 ; i++ )
7  {

```



```

8     for(int j = 1 ; j + (1 << i) - 1 <= n ; j++ )
9     {
10         f[i][j] = max(f[i-1][j], f[i-1][j + (1 << i - 1)]);
11     }
12 }
13 for(int i = 1 ; i <= m ; i++ )
14 {
15     int l = read(), r = read();
16     int len = __lg(r - l + 1);
17     printf("%d\n", max(f[len][l], f[len][r - (1 << len) + 1]));
18 }

```

## 树状数组

```

1  template<class T>
2  struct BIT{
3      T c[N];
4      void change(int x, T y)
5      {
6          for(; x < N ; x += x & (-x))
7          {
8              c[x] += y;
9          }
10     }
11     T query(int x)
12     {
13         T s = 0;
14         for(; x ; x -= x & (-x))
15         {
16             s += c[x];
17         }
18         return s;
19     }
20 };

```

## 并查集

```

1  struct DSU {
2      std::vector<int> f, siz;
3      DSU(int n) : f(n), siz(n, 1) { std::iota(f.begin(), f.end(), 0); }
4      int leader(int x) {
5          while (x != f[x]) x = f[x] = f[f[x]];
6          return x;
7      }
8      bool same(int x, int y) { return leader(x) == leader(y); }
9      bool merge(int x, int y) {
10         x = leader(x);
11         y = leader(y);
12         if (x == y) return false;
13         siz[x] += siz[y];
14         f[y] = x;
15         return true;
16     }
17     int size(int x) { return siz[leader(x)]; }

```

## 二维树状数组维护区间查询，修改

```

1  ll c1[N][N], c2[N][N], c3[N][N], c4[N][N];
2
3  int n, m, k, q;
4
5  int lowbit(int x)
6  {
7      return x & (-x);
8  }
9
10 void add(ll x, ll y, ll d)
11 {
12     for(int i = x ; i <= n ; i += lowbit(i))
13     {
14         for(int j = y ; j <= m ; j += lowbit(j))
15         {
16             //cout << "test" << endl;
17             c1[i][j] += d;
18             c2[i][j] += d * x;
19             c3[i][j] += d * y;
20             c4[i][j] += d * x * y;
21         }
22     }
23 }
24
25 void modify(int x1, int y1, int x2, int y2, int d)
26 {
27     add(x1, y1, d);
28     add(x1, y2 + 1, -d);
29     add(x2 + 1, y1, -d);
30     add(x2 + 1, y2 + 1, d);
31 }
32
33 ll sum(ll x, ll y)
34 {
35     ll ans = 0;
36     for(int i = x ; i ; i -= lowbit(i))
37     {
38         for(int j = y ; j ; j -= lowbit(j))
39         {
40             ans += (x + 1) * (y + 1) * c1[i][j];
41             ans -= (y + 1) * c2[i][j];
42             ans -= (x + 1) * c3[i][j];
43             ans += c4[i][j];
44         }
45     }
46     return ans;
47 }
48 ll query(int x1, int y1, int x2, int y2)
49 {
50     return (sum(x2, y2) - sum(x1 - 1, y2) - sum(x2, y1 - 1) + sum(x1 - 1, y1
    - 1));

```

```

51 }
52 int h[100005];
53 int main()
54 {
55     fastio
56     //freopen("1.in","r",stdin);
57     cin >> n >> m >> k >> q;
58     for(int i = 1 ; i <= k ; i++ )
59     {
60         cin >> h[i];
61     }
62     for(int i = 1 ; i <= q ; i++ )
63     {
64         int op;
65         cin >> op;
66         if(op == 1)
67         {
68             int a, b, c, d, id;
69             cin >> a >> b >> c >> d >> id;
70             modify(a, b, c, d, h[id]);
71         }else{
72             int a, b, c, d;
73             cin >> a >> b >> c >> d;
74             cout << query(a, b, c, d) << "\n";
75         }
76     }
77     return 0;
78 }
79

```

## 线段树（区间查询最小值，最小值个数）

```

1  struct Node{
2      int minx, cntminx;
3  };
4
5  ll a[N];
6
7  Node tr[4 * N];
8
9  void pushup(int u, int L, int R)
10 {
11     if(tr[u << 1].minx < tr[u << 1 | 1].minx)
12     {
13         tr[u].minx = tr[u << 1].minx;
14         tr[u].cntminx = tr[u << 1].cntminx;
15     }
16     if(tr[u << 1].minx > tr[u << 1 | 1].minx)
17     {
18         tr[u].minx = tr[u << 1 | 1].minx;
19         tr[u].cntminx = tr[u << 1 | 1].cntminx;
20     }
21     if(tr[u << 1].minx == tr[u << 1 | 1].minx)
22     {
23         tr[u].minx = tr[u << 1 | 1].minx;

```

```

24         tr[u].cntminx = tr[u << 1].cntminx + tr[u << 1 | 1].cntminx;
25     }
26 }
27
28
29 void build(int u, int L, int R)
30 {
31     int mid = L + R >> 1;
32     if(L == R)
33     {
34         tr[u].minx = a[L];
35         tr[u].cntminx = 1;
36         return;
37     }
38     build(u << 1, L, mid);
39     build(u << 1 | 1, mid + 1, R);
40     pushup(u, L, R);
41
42 }
43
44 void change(int u, int L, int R, int x, int y)
45 {
46     int mid = L + R >> 1;
47     if(L == R)
48     {
49         tr[u].minx = y;
50         return;
51     }
52     if(x <= mid)
53     {
54         change(u << 1, L, mid, x, y);
55     }
56     if(x > mid)
57     {
58         change(u << 1 | 1, mid + 1, R, x, y);
59     }
60     pushup(u, L, R);
61 }
62
63 pair<int, int> query(int u, int L, int R, int l, int r)
64 {
65     int mid = L + R >> 1;
66     if(l <= L && R <= r)
67     {
68         return {tr[u].minx, tr[u].cntminx};
69     }
70     if(r <= mid)
71     {
72         return query(u << 1, L, mid, l, r);
73     }
74     if(l >= mid + 1)
75     {
76         return query(u << 1 | 1, mid + 1, R, l, r);
77     }
78     auto s1 = query(u << 1, L, mid, l, r);
79     auto s2 = query(u << 1 | 1, mid + 1, R, l, r);

```

```

80     if(s1.first < s2.first)
81     {
82         return s1;
83     }
84     if(s1.first > s2.first)
85     {
86         return s2;
87     }
88     return {s1.first, s1.second + s2.second};
89 }
90
91 int main()
92 {
93     fastio
94     //freopen("1.in","r",stdin);
95     int n, m;
96     cin >> n >> m;
97     for(int i = 1 ; i <= n ; i++ )
98     {
99         cin >> a[i];
100     }
101     build(1, 1, n);
102     for(int i = 1 ; i <= m ; i++ )
103     {
104         int op, x, y;
105         cin >> op >> x >> y;
106         if(op == 1)
107         {
108             change(1, 1, n, x, y);
109         }else{
110             auto [_,__] = query(1, 1, n, x, y);
111             cout << _ << " " << __ << "\n";
112         }
113     }
114     return 0;
115 }

```

## 线段树（区间修改加法，区间查询）

```

1  struct Node{
2      ll sum, lazy, size;
3  };
4  Node tr[N * 4];
5  ll a[N];
6
7  void pushup(int u, int L, int R)
8  {
9      tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
10 }
11
12
13 void build(int u, int L, int R)
14 {
15     int mid = L + R >> 1;
16     tr[u].size = R - L + 1;

```

```

17     tr[u].sum = tr[u].lazy = 0;
18     if(L == R)
19     {
20         tr[u].sum = a[L];
21         return;
22     }
23     build(u << 1, L, mid);
24     build(u << 1 | 1, mid + 1, R);
25     pushup(u, L, R);
26
27 }
28
29 void pushdown(int u)
30 {
31     auto &root = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
32     if(root.lazy)
33     {
34         left.sum += root.lazy * left.size;
35         left.lazy += root.lazy;
36         right.sum += root.lazy * right.size;
37         right.lazy += root.lazy;
38         root.lazy = 0;
39     }
40 }
41
42 void pushup(int u)
43 {
44     tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
45 }
46
47 ll query(int u, int L, int R, int l, int r)
48 {
49     int mid = L + R >> 1;
50     if(l <= L && R <= r)
51     {
52         return tr[u].sum;
53     }
54     ll ans = 0;
55     pushdown(u);
56     if(l <= mid)
57     {
58         ans += query(u << 1, L, mid, l, r);
59     }
60     if(r > mid)
61     {
62         ans += query(u << 1 | 1, mid + 1, R, l, r);
63     }
64     return ans;
65 }
66
67 void modify(int u, int L, int R, int l, int r, int x)
68 {
69     int mid = L + R >> 1;
70     if(l <= L && R <= r)
71     {
72         tr[u].lazy += x;

```

```

73     tr[u].sum += x * tr[u].size;
74     return;
75 }
76 pushdown(u);
77 if(l <= mid)
78 {
79     modify(u << 1, L, mid, l, r, x);
80 }
81 if(r > mid)
82 {
83     modify(u << 1 | 1, mid + 1, R, l, r, x);
84 }
85 pushup(u);
86 }

```

## 线段树（区间修改加与乘，区间查询）

```

1  struct Node{
2      ll sum, mul, add, size;
3  } tr[4 * N];
4  ll a[N];
5
6  void pushup(int u)
7  {
8      tr[u].sum = (tr[u << 1].sum % P + tr[u << 1 | 1].sum % P) % P;
9  }
10
11 void pushdown(int u)
12 {
13     auto &root = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
14     root.mul %= P, root.add %= P;
15     left.sum *= root.mul;          left.sum %= P;
16     left.sum += root.add * left.size; left.sum %= P;
17     right.sum *= root.mul;         right.sum %= P;
18     right.sum += root.add * right.size; right.sum %= P;
19     left.add *= root.mul;          left.add %= P;
20     left.mul *= root.mul;          left.mul %= P;
21     right.add *= root.mul;         right.add %= P;
22     right.mul *= root.mul;         right.mul %= P;
23     left.add += root.add;          left.add %= P;
24     right.add += root.add;         right.add %= P;
25     root.mul = 1;
26     root.add = 0;
27 }
28
29 void build(int u, int L, int R)
30 {
31     int mid = L + R >> 1;
32     tr[u].size = R - L + 1;
33     tr[u].mul = 1;
34     tr[u].add = 0;
35     if(L == R)
36     {
37         tr[u].sum = a[L] % P;
38         return;

```

```

39     }
40     build(u << 1, L, mid);
41     build(u << 1 | 1, mid + 1, R);
42     pushup(u);
43 }
44
45 void modify_add(int u, int L, int R, int l, int r, int x)
46 {
47     int mid = L + R >> 1;
48     if(l <= L && R <= r)
49     {
50         tr[u].sum += tr[u].size * x;    tr[u].sum %= P;
51
52         tr[u].add += x;                  tr[u].add %= P;
53         return;
54     }
55     pushdown(u);
56     if(l <= mid)
57     {
58         modify_add(u << 1, L, mid, l, r, x);
59     }
60     if(r >= mid + 1)
61     {
62         modify_add(u << 1 | 1, mid + 1, R, l, r, x);
63     }
64     pushup(u);
65 }
66
67 void modify_mul(int u, int L, int R, int l, int r, int x)
68 {
69     int mid = L + R >> 1;
70     if(l <= L && R <= r)
71     {
72         tr[u].sum *= x; tr[u].sum %= P;
73         tr[u].add *= x; tr[u].add %= P;
74         tr[u].mul *= x; tr[u].mul %= P;
75         return;
76     }
77     pushdown(u);
78     if(l <= mid)
79     {
80         modify_mul(u << 1, L, mid, l, r, x);
81     }
82     if(r >= mid + 1)
83     {
84         modify_mul(u << 1 | 1, mid + 1, R, l, r, x);
85     }
86     pushup(u);
87 }
88
89 ll query(int u, int L, int R, int l, int r)
90 {
91     if(l <= L && R <= r)
92     {
93         return tr[u].sum % P;
94     }

```



```

95     pushdown(u);
96     ll ans = 0;
97     int mid = L + R >> 1;
98     if(l <= mid)
99     {
100         ans += query(u << 1, L, mid, 1, r);
101         ans %= P;
102     }
103     if(r >= mid + 1)
104     {
105         ans += query(u << 1 | 1, mid + 1, R, 1, r);
106         ans %= P;
107     }
108     pushup(u);
109     return ans % P;
110 }

```

## DynamicSegmentTree

```

1  class SegTree {
2  private:
3      struct Node {
4          Node () : left_(nullptr), right_(nullptr), val_(0), lazy_(0) {}
5          int val_;
6          int lazy_;
7          Node* left_;
8          Node* right_;
9      };
10
11 public:
12     Node* root_;
13     SegTree() { root_ = new Node(); }
14     ~SegTree() {}
15
16     // 更新区间值
17     void upDate(Node* curNode, int curLeft, int curRight, int upDateLeft,
18 int upDateRight, int addVal) {
19         if (upDateLeft <= curLeft && upDateRight >= curRight) {
20             // 如果需要更新的区间[upDateLeft, upDateRight] 包含了 当前这个区间
21             [curLeft, curRight]
22             // 那么暂存一下更新的值
23             // 等到什么时候用到孩子结点了，再把更新的值发放给孩子
24             curNode->val_ += addVal * (curRight - curLeft + 1);
25             curNode->lazy_ += addVal;
26             return;
27         }
28
29         // 到这里说明要用到左右孩子了
30         // 因此，要用pushDown函数把懒标签的值传递下去
31         int mid = (curLeft + curRight) / 2;
32         pushDown(curNode, mid - curLeft + 1, curRight - mid);
33
34         // 说明在[curLeft, curRight]中，
35         if (upDateLeft <= mid) {

```

```

34         upDate(curNode->left_, curLeft, mid, upDateLeft, upDateRight,
addVal);
35     }
36     if (upDateRight > mid) {
37         upDate(curNode->right_, mid + 1, curRight, upDateLeft,
upDateRight, addVal);
38     }
39
40     // 更新了子节点还需要更新现在的结点
41     pushUp(curNode);
42 }
43
44
45 // 把结点curNode的懒标记分发给左右孩子 然后自己的懒标记清零
46 void pushDown(Node* curNode, int leftChildNum, int rightChildNum) {
47     if (curNode->left_ == nullptr) curNode->left_ = new Node;
48     if (curNode->right_ == nullptr) curNode->right_ = new Node;
49
50     if (curNode->lazy_ == 0) return;
51
52     curNode->left_->val_ += curNode->lazy_ * leftChildNum;
53     curNode->left_->lazy_ += curNode->lazy_;
54
55     curNode->right_->val_ += curNode->lazy_ * rightChildNum;
56     curNode->right_->lazy_ += curNode->lazy_;
57
58     curNode->lazy_ = 0;
59
60     // 注意不需要递归再继续下推懒标签
61     // 每次只需要推一层即可
62 }
63
64 // 一般是子节点因为要被用到了，所以需要更新值 因此也要同时更新父节点的值
65 void pushUp(Node* curNode) {
66     curNode->val_ = curNode->left_->val_ + curNode->right_->val_;
67 }
68
69 // 查询
70 int query(Node* curNode, int curLeft, int curRight, int queryLeft, int
queryRight) {
71     if (queryLeft <= curLeft && queryRight >= curRight) {
72         return curNode->val_;
73     }
74     // 用到左右结点力 先下推！
75     int mid = (curLeft + curRight) / 2;
76     pushDown(curNode, mid - curLeft + 1, curRight - mid);
77
78     int curSum = 0;
79     if (queryLeft <= mid) curSum += query(curNode->left_, curLeft, mid,
queryLeft, queryRight);
80     if (queryRight > mid) curSum += query(curNode->right_, mid + 1,
curRight, queryLeft, queryRight);
81
82     return curSum;
83 }
84 };

```

## pdbbs

```
1  #include<ext/pb_ds/tree_policy.hpp>
2  #include<ext/pb_ds/assoc_container.hpp>
3
4  using namespace __gnu_pbds;
5  __gnu_pbds::tree<ll, null_type, less<ll>, rb_tree_tag,
6  tree_order_statistics_node_update> T;
7
8  if(op == 1)
9  {
10     T.insert({x, i});
11 }else if (op == 2)
12 {
13     T.erase(T.lower_bound({x, 0}));
14 }else if (op == 3)
15 {
16     cout << T.order_of_key({x, 0}) + 1 << "\n";
17 }else if (op == 4)
18 {
19     cout << T.find_by_order(x - 1)->first << "\n";
20 }else if (op == 5)
21 {
22     cout << prev(T.lower_bound({x, 0}))->first << "\n";
23 }else if (op == 6)
24 {
25     cout << T.lower_bound({x + 1, 0})->first << "\n";
26 }
```

## 六、简单计算几何

### 点

```
1  using i64 = long long;
2
3  using T = double;
4  struct Point {
5      T x;
6      T y;
7      Point(T x = 0, T y = 0) : x(x), y(y) {}
8
9      Point &operator+=(const Point &p) {
10         x += p.x, y += p.y;
11         return *this;
12     }
13     Point &operator-=(const Point &p) {
14         x -= p.x, y -= p.y;
15         return *this;
16     }
```

```

16     }
17     Point &operator*=(const T &v) {
18         x *= v, y *= v;
19         return *this;
20     }
21     friend Point operator-(const Point &p) {
22         return Point(-p.x, -p.y);
23     }
24     friend Point operator+(Point lhs, const Point &rhs) {
25         return lhs += rhs;
26     }
27     friend Point operator-(Point lhs, const Point &rhs) {
28         return lhs -= rhs;
29     }
30     friend Point operator*(Point lhs, const T &rhs) {
31         return lhs *= rhs;
32     }
33 };
34
35 T dot(const Point &a, const Point &b) {
36     return a.x * b.x + a.y * b.y;
37 }
38
39 T cross(const Point &a, const Point &b) {
40     return a.x * b.y - a.y * b.x;
41 }

```

## 七、杂项

### 矩阵快速幂

```

1  struct Matrix{
2      int n , m ;
3      vector<vector<ll>> s;
4
5      Matrix(int n , int m):n(n) ,m(m) , s(n , vector<ll>(m ,0)){}
6
7      friend Matrix operator * (Matrix a , Matrix b){
8          assert(a.m == b.n);
9          Matrix res(a.n , b.m);
10         for(int k = 0 ; k < a.m ; k ++ )
11             for(int i = 0 ; i < a.n ; i ++ )
12                 for(int j = 0 ; j < b.m ; j ++ )
13                     res.s[i][j] = (res.s[i][j] + a.s[i][k] * b.s[k][j] %
mod) % mod;
14         return res;
15     }
16
17     Matrix qmi(ll b){
18         assert(n == m);
19         Matrix res(n , n);
20         for(int i = 0 ; i < n ; i ++ )
21             res.s[i][i] = 1;
22         while(b){

```

```

23         if(b & 1)res = ((*this) * res );
24         b >>= 1;
25         *this = (*this) * (*this);
26     }
27     return (*this) = res;
28 };
29
30 };

```

## 组合数

```

1  ll fact[N] = {1}, inv[N] = {1};
2  ll c(ll x, ll y)
3  {
4      return(((fact[x] * inv[y])% MOD * inv[x-y]) % MOD);
5  }
6
7  ll P(ll x, ll y)
8  {
9      return fact[x] * inv[x - y] % MOD;
10 }
11
12 ll ksm(ll x, ll y)
13 {
14     ll ans = 1;
15     x %= MOD;
16     while(y)
17     {
18         if(y&1)
19         {
20             ans = ans * x % MOD;
21         }
22         x = x * x % MOD;
23         y /= 2;
24     }
25     return ans;
26 }
27
28 void build()
29 {
30     for(int i = 1 ; i < N ; i++ )
31     {
32         fact[i] = fact[i-1] * i % MOD;
33     }
34     for(int i = 1 ; i < N ; i++ )
35     {
36         inv[i] = inv[i-1] * ksm(i, MOD-2) % MOD;
37     }
38 }

```

## 八、python

```

1  '''

```

```

2  def main():
3      Do something
4  if __name__ == '__main__':
5      t = int(input())
6      for i in range(t):
7          main()
8      '''
9  for T in range(0,int(input())): #T组数据
10     N=int(input())
11     n,m=map(int,input().split())
12     s=input()
13     s=[int(x) for x in input().split()] #一行输入的数组
14     a[1:]=[int(x) for x in input().split()] #从下标1开始读入一行
15     for i in range(0,len(s)):
16         a,b=map(int,input().split())
17
18 while True: #未知多组数据
19     try:
20         #n,m=map(int,input().split())
21         #print(n+m,end="\n")
22     except EOFError: #捕获到异常
23         break
24     //////////////////////////////////
25     '''多行输入，指定行数'''
26
27 n, m = map(int, input().strip().split())#获取第一行，获取第二行可以再写一句同样的语
    句
28 #需要矩阵承接数据时
29 data = []
30 for i in range(n):
31     tmp = list(map(int, input().split()))
32     data.append(tmp)
33
34 '''多行输入，不指定行数'''
35 try:
36     data = []
37     while True:
38         line = input().strip() #strip去除左右两边的空白符
39         if line == ' ':
40             break
41         tmp = list(map(int, line.split())) #split按空白符拆开
42         data.append(tmp)
43 expect:
44     pass
45

```

## 一些基本数据结构

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

匿名函数使用lambda关键字来定义 `lambda 参数: 表达式`

```

1  #使用中括号[]定义一个列表
2  # l=[23,'wtf',3.14]
3  list.append(obj)#将obj添加到list末尾,0(1)
4  list.insert(index,obj)#将obj插入列表index位置,0(n)

```

```

5 list.pop([index=-1])#移除元素并返回该元素
6 list.sort(key=None,reverse=False)#默认升序排序,o(nlogn)
7 list.reverse()#反转列表元素
8 list.clear()
9 len(list)#列表元素个数,o(1)
10 max(list)#返回列表元素最大值,o(n)
11 del list[2]#删除list中第三个元素
12
13 #用小括号定义一个元组,可以当作不能修改的list
14 # t=(23,'wtf',3.14)
15
16 #用花括号{}定义一个字典
17 d={key1:value1,key2:value2}#通过key访问value
18 print(d[key1])#输出value1
19 if key in dict : #key不存在会报错,要先询问
20     do something #或者使用
21 d.get(key)
22 for key in d: #遍历字典d
23     print(key,':',d.get(key))
24 dMerge=dict(d1,**d2)#将d1和d2合并为dMerge
25
26 #调用set()方法创建集合
27 s=set([1,2,3])#定义
28 s.add(4)#添加
29 s.remove(4)#删除

```

## math库

```

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

```

## 快速幂

```

1 def qmod(a,b,mod):
2     a=a%mod
3     ans=1
4     while b!=0:
5         if b&1:
6             ans=(ans*a)%mod
7             b>>=1
8             a=(a*a)%mod
9     return ans

```

## 并查集

```
1 N,m=map(int,input().split())
2 fa=[int(i) for i in range(N+1)]
3 siz=[1]*(N+1)
4 def findfa(x):
5     if fa[x]!=x:
6         fa[x]=findfa(fa[x])
7     return fa[x]
8 def Merge(x,y):
9     xx,yy=findfa(x),findfa(y)
10    if xx == yy:
11        return False
12    if siz[xx] > siz[yy]: #按秩合并
13        fa[yy]=xx
14        siz[xx]+=siz[yy]
15    else:
16        fa[xx]=yy
17        siz[yy]+=siz[xx]
18    return True
19 for i in range(m):
20     z,x,y=map(int,input().split())
21     if z==1:
22         Merge(x,y)
23     else:
24         print('Y' if findfa(x)==findfa(y) else 'N')
```

## 线段树区间加区间和

```
1 class SegTreeNode(): #python3中所有类默认都是新式类
2     def __init__(self): #类似构造函数,类方法必须包含参数self
3         self.value=0
4         self.lazytag=0
5
6 Data=[0 for i in range(0,100010)]
7
8 class SegTree():
9     def __init__(self):
10         self.SegTree=[SegTreeNode() for i in range(0,400010)]
11
12     def Build_SegTree(self,Root,L,R):
13         if L==R:
14             self.SegTree[Root].value=Data[L]
15             return
16         mid=(L+R)>>1
17         self.Build_SegTree(Root<<1,L,mid)
18         self.Build_SegTree(Root<<1|1,mid+1,R)
19
20         self.SegTree[Root].value=self.SegTree[Root<<1].value+self.SegTree[Root<<1|1].value
21
22         return
23
24     def Push_Down(self,Root,L,R):
25         if self.SegTree[Root].lazytag==0:
```



```

24         return
25         Add=self.SegTree[Root].lazytag
26         self.SegTree[Root].lazytag=0
27         mid=(L+R)>>1
28         self.SegTree[Root<<1].value+=(mid-L+1)*Add
29         self.SegTree[Root<<1|1].value+=(R-mid)*Add
30         self.SegTree[Root<<1].lazytag+=Add
31         self.SegTree[Root<<1|1].lazytag+=Add
32         return
33
34     def Update(self,Root,L,R,QL,QR,Add):
35         if R<QL or QR<L:
36             return
37         if QL<=L and R<=QR:
38             self.SegTree[Root].value+=(R-L+1)*Add
39             self.SegTree[Root].lazytag+=Add
40             return
41         mid=(L+R)>>1
42         self.Push_Down(Root,L,R)
43         self.Update(Root<<1,L,mid,QL,QR,Add)
44         self.Update(Root<<1|1,mid+1,R,QL,QR,Add)
45
46         self.SegTree[Root].value=self.SegTree[Root<<1].value+self.SegTree[Root<<1|1]
47         .value
48         return
49
50     def Query(self,Root,L,R,QL,QR):
51         if R<QL or QR<L:
52             return 0
53         if QL<=L and R<=QR:
54             return self.SegTree[Root].value
55         mid=(L+R)>>1
56         self.Push_Down(Root,L,R)
57         return
58         self.Query(Root<<1,L,mid,QL,QR)+self.Query(Root<<1|1,mid+1,R,QL,QR)
59
60 Tree=SegTree()
61 N,M=map(int,input().split())
62 a=input().split() #初始值
63
64 for i in range(1,N+1):
65     Data[i]=int(a[i-1])
66
67 Tree.Build_SegTree(1,1,N)
68
69 while M:
70     opt,L,R=map(int,input().split())
71     if opt==1:
72         Tree.Update(1,1,N,L,R,int(a[3]))
73     else:
74         print(str(Tree.Query(1,1,N,L,R)))
75     M-=1

```

## 字符串

```

1 ord('a')# 返回单个字符的 unicode:
2 chr(100)# 返回'd'
3
4 #strip和split
5 '   spacious   '.strip()#strip()移除 string 前后的字符串，默认来移除空格
6 '1,2,3'.split(',') #['1', '2', '3'],按照某个字符串来切分，返回一个 list,
7 '1,2,3'.split(',', maxsplit=1)#[ '1', '2,3'],传入一个参数maxsplit来限定分离数
8
9 #将字符串和列表相互转换
10 字符串转换成列表，注意交换字符需要先转换成列表
11 #1.list
12 str1 = '12345'
13 list1 = list(str1)
14 print(list1) #['1', '2', '3', '4', '5']
15 #2.str.split()通过指定分隔符对字符串进行切片
16 str3 = 'this is string example'
17 list3 = str3.split('i', 1)
18 print(list3) #['th', 's is string example']
19
20 列表转换成字符串，join里面的可以是list、set
21 #1.split.join(str),split是指定的分隔符，str是要转换的字符串
22 list1 = ['1', '2', '3', '4', '5']
23 str1 = "".join(list1)#12345
24
25 list3 = ['www', 'baidu', 'com']
26 str3 = ".".join(list3)#www.baidu.com
27
28 #是元音
29 def isVowel(ch:str) -> bool:
30     return ch in "aeiouAEIOU"
31 isVowel(s[i])
32

```

## 二维列表

```

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

```

## list

```

1 #初始化
2 l = [0] * len(array)
3 l=[]
4
5 #从后往前访问
6 l[-1]表示最后一个数
7 for i in range(0, -10, -1)      #0, -1, -2, -3, -4, -5, -6, -7, -8, -9
8 for j in reversed(range(len(nums)-1)) #加一个reverse可以直接颠倒
9
10 #enumerate 枚举
11 l = ["a", "b", "c"]
12 for i, v in enumerate(l):

```

```

13     print(i, v)
14 #0 a
15 #1 b
16 #2 c
17
18 #map
19 #可以将参数一一映射来计算， 比如
20 date = "2019-8-15"
21 Y, M, D = map(int, date.split('-'))    #Y = 2019, M = 8, D = 15
22 #map返回的是迭代对象而不是一个列表，要转成列表要加list
23
24
25 #sort
26 1.调用sort()排序，不会产生新的列表。lst.sort()升序排序
27 降序排序lst.sort(reverse=True) 升序排序lst.sort()
28 2.使用内置函数sorted()排序，会产生新的列表对象
29 lst1=sorted(lst)升序排序    lst2=sorted(lst,reverse=True)降序排序
30 l1 = [(1,2), (0,1), (3,10) ]
31 l2 = sorted(l1, key=lambda x: x[0])#按照 tuple 的第一个元素进行排序key允许传入一个
    自定义参数
32 # l2 = [(0, 1), (1, 2), (3, 10)]
33 #排序默认从小到大。可以用reverse=True倒序
34
35 #列表生成式
36 lst = [i*j for i in range(1,10)]
37 #ZIP
38 x = [1, 2, 3]
39 y = [4, 5, 6]
40 zipped = zip(x, y)
41 list(zipped)#[(1, 4), (2, 5), (3, 6)]
42 ````keys(), values(), items()
43 这三个方法可以分别获得key, value, {key: value}的数组。
44
45 #max可以代替if来更新更大的数
46 maxnums=max(maxnums, tmp)
47
48 #多维数组
49 res = [[], []]
50 res[0].append()
51
52 #extend一次性添加多个元素
53 lst1.extend(lst2)
54 #insert在i位置添加x
55 lst.insert(i, x)
56

```

## 常用函数

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