LiYue University
Genshin Impact Science

Higher Elemental Theory

Author
Chen Free

September 4, 2022

目录

dinic

```
inline void addage(int x, int y, int z)
{
	to[++cnt] = y;
	Next[cnt] = from[x];
	from[x] = cnt;
	val[cnt] = z;
}
inline void add(int x, int y, int z)
{
	addage(x, y, z);
	addage(y, x, 0);
}
int bfs()
{
	queue<int> q;
	memset(d, 0, sizeof(d));
	d[S] = 1;
	cur[S] = from[S];
	q.push(S);
	while (q.size())
	{
		int x = q.front();
		q.pop();
		for (int i = from[x]; i; i = Next[i])
		{
			int y = to[i];
			if (val[i] && !d[y])
			{
				d[y] = d[x] + 1;
				cur[y] = from[y];
				q.push(y);
				if (y == T)
					return 1;
			}
		}
	}
	return 0;
}
int dinic(int x, int flow)
{
	if (x == T)
		return flow;
	int k, rest = flow;
	for (int i = from[x]; i && rest; i = Next[i])
	{
		int y = to[i];
		if (val[i] && d[y] == d[x] + 1)
```

```
            {
                  k = dinic(y, min(rest, val[i]));
                  if (!k)
                        d[y] = 0;
                  val[i] -= k;
                  val[i ^ 1] += k;
                  rest -= k;
            }
      }
      return flow - rest;
}
int dinic()
{
      int res = 0;
      while (bfs())
      {
            int tep;
            while (tep = dinic(S, INF))
                  res += tep;
      }
      return res;
}
void work()
{
      int n = read(), m = read();
      S = read(), T = read();
      for (int i = 1; i <= m; i++)
      {
            int x = read(), y = read(), z = read();
            add(x, y, z);
      }
      int maxflow = dinic();
      printf("%d\n", maxflow);
      return;
}
```

KMP

```
void KMP()
{
      Next[0] = -1;
      int k = -1;
      for (int q = 1; q < len2; q++)
      {
            while (k != -1 && st2[k + 1] != st2[q])
                  k = Next[k];
            if (st2[k + 1] == st2[q])
                  k++;
            Next[q] = k;
      }
```

```
	k = -1;
	for (int i = 0; i < len1; i++)
	{
		while (k != -1 && st2[k + 1] != st1[i])
			k = Next[k];
		if (st2[k + 1] == st1[i]) k++;
		if (k == len2 - 1)
		{
			i = i - len2 + 1;
			k = -1;
			cout << i + 1 << endl;
		}
	}
}

int main()
{
	cin >> st1;
	cin >> st2;
	len1 = st1.length();
	len2 = st2.length();
	KMP();
	for (int i = 0; i < len2; i++)
		cout << Next[i] + 1 << " ";
	return 0;
}
```

kruskal 重构树求最小瓶颈路

```
int fa[N], f[N][20], val[N], lg[N], dep[N], vis[N];
int n, m, cnt;
vector<int>a[N];
pair<int, pii>e[N];
inline int read()
{
	int X = 0, w = 0;
	char ch = 0;
	while (!isdigit(ch))
	{
		w |= ch == '-';
		ch = getchar();
	}
	while (isdigit(ch)) X = (X << 3) + (X << 1) + (ch ^ 48), ch = get
char();
	return w ? -X : X;
}
int find(int x)
{
	if (x != fa[x])
```

```
            fa[x] = find(fa[x]);
      return fa[x];
}
void add(int x, int y)
{
      a[x].push_back(y);
      a[y].push_back(x);
}
void init()
{
      for (int i = 1; i < N; i++)
      {
            lg[i] = lg[i - 1];
            if (i == (1 << lg[i - 1]))lg[i]++;
      }
}
void dfs(int x, int father)
{
      vis[x] = 1;
      dep[x] = dep[father] + 1;
      f[x][0] = father;
      for (int i = 1; (1 << i) <= dep[x]; i++)
            f[x][i] = f[f[x][i - 1]][i - 1];
      for (auto y : a[x])
            if (y != father)
                  dfs(y, x);
}
int LCA(int x, int y)
{
      if (dep[x] < dep[y])
            swap(x, y);
      while (dep[x] != dep[y])
            x = f[x][lg[dep[x] - dep[y]] - 1];
      if (x == y)return x;
      for (int i = lg[dep[x]]; i >= 0; i--)
            if (f[x][i] != f[y][i])
                  x = f[x][i], y = f[y][i];
      return f[x][0];
}
void kruskal()
{
      sort(e + 1, e + m + 1);
      for (int i = 1; i <= n; i++)
            fa[i] = i;
      for (int i = 1; i <= m; i++)
      {
            auto [x, y] = e[i].second;
            int z = e[i].first;
            x = find(x), y = find(y);
```

```
            if (x == y)continue;
            val[++cnt] = z;
            fa[cnt] = fa[x] = fa[y] = cnt;
            add(x, cnt);
            add(y, cnt);
      }
      for (int i = 1; i <= cnt; i++)
            if (!vis[i])
                  dfs(find(i), 0);
}
signed main()
{
#ifndef ONLINE_JUDGE
      freopen("C:\\Users\\FREE\\Desktop\\1.in", "r", stdin);
#endif
      init();
      n = read(), m = read();
      int q = read();
      cnt = n;
      for (int i = 1; i <= m; i++)
      {
            int x = read(), y = read(), z = read();
            e[i] = { z, {x, y} };
      }
      kruskal();
      while (q--)
      {
            int x = read(), y = read();
            if (find(x) != find(y))
                  puts("-1");
            else
                  printf("%d\n", val[LCA(x, y)]);
      }
      return 0;
}
```

lucas 定理

```
int qpow(int x, int k)
{
      int tep = 1ll;
      while (k)
      {
            if (k & 1ll)
                  tep = tep * x % p;
            x = x * x % p;
            k >>= 1ll;
      }
      return tep;
```

```
}
int C(int x, int y)
{
	if (x < y)
		return 0;
	return (jc[x] * qpow(jc[x - y], p - 2) % p * qpow(jc[y], p - 2));
}
int lucas(int x, int y)
{
	if (!y)
		return 1;
	return C(x % p, y % p) * lucas(x / p, y / p) % p;
}
void init()
{
	memset(jc, 0, sizeof(jc));
	jc[0] = 1;
	for (int i = 1; i <= p; i++)
		jc[i] = (jc[i - 1] * i) % p;
}
signed main()
{
	int T = read();
	while (T--)
	{
		int n = read(), m = read();
		p = read();
		init();
		printf("%lld\n", lucas(n + m, n));
	}
	return 0;
}
```

spfa 最小费用最大流

```
int to[N], Next[N], from[N], val[N], fy[N];
int cnt = 1, ans_cost, ans_flow;
int vis[N], dis[N];
int S, T;
inline void addage(int x, int y, int z, int c)
{
	to[++cnt] = y;
	Next[cnt] = from[x];
	from[x] = cnt;
	val[cnt] = z;
	fy[cnt] = c;
}
inline void add(int x, int y, int z, int c)
{
```

```
	addage(x, y, z, c);
	addage(y, x, 0, -c);
}
int spfa()
{
	memset(vis, 0, sizeof(vis));
	memset(dis, 0x3f, sizeof(dis));
	vis[T] = 1;
	dis[T] = 0;
	deque<int> q;
	q.push_back(T);
	while (q.size())
	{
		int x = q.front();
		q.pop_front();
		vis[x] = 0;
		for (int i = from[x]; i; i = Next[i])
		{
			if (val[i ^ 1] <= 0)
				continue;
			int y = to[i];
			if (dis[y] > dis[x] - fy[i])
			{
				dis[y] = dis[x] - fy[i];
				if (vis[y] == 0)
				{
					vis[y] = 1;
					if (q.size() && dis[y] < dis[q.front()])
						q.push_front(y);
					else
						q.push_back(y);
				}
			}
		}
	}
	return dis[S] != dis[N - 1];
}
int dfs(int x, int flow)
{
	if (x == T)
	{
		vis[T] = 1;
		return flow;
	}
	int used = 0;
	vis[x] = 1;
	for (int i = from[x]; i; i = Next[i])
	{
		int y = to[i];
```

```
            if (vis[y] == 0 && val[i] > 0 && dis[x] - fy[i] == dis[y])
            {
                int tep = dfs(y, min(val[i], flow - used));
                if (tep > 0)
                {
                    ans_cost += tep * fy[i];
                    val[i] -= tep;
                    val[i ^ 1] += tep;
                    used += tep;
                }
                if (used == flow)
                    break;
            }
        }
        return used;
}
void costflow()
{
        ans_cost = 0;
        ans_flow = 0;
        while (spfa())
        {
            vis[T] = 1;
            while (vis[T])
            {
                memset(vis, 0, sizeof(vis));
                ans_flow += dfs(S, INF);
            }
        }
}
void work()
{
        int n = read(), m = read();
        S = read(), T = read();
        for (int i = 1; i <= m; i++)
        {
            int x = read(), y = read(), z = read(), c = read();
            add(x, y, z, c);
        }
        costflow();
        printf("%d %d\n", ans_flow, ans_cost);
        // for (int i = 2; i <= cnt; i += 2)
        // {
        //     val[i] += val[i ^ 1];
        //     val[i ^ 1] = 0;
        //     fy[i] = -fy[i];
        //     fy[i ^ 1] = -fy[i ^ 1];
        // }
        // costflow();
```

```
	// printf("%d\n", -ans_cost);
}
```

ST 表

```
void init()
{
	for (int i = 1; i < N; i++)
	{
		lg[i] = lg[i - 1];
		if (1 << lg[i] == i)lg[i]++;
	}
}
void ST()
{
	for (int i = 1; i <= n; i++)
		st[i][0] = read();
	for (int j = 1; j <= 20; j++)
		for (int i = 1; i + (1 << j) - 1 <= n; i++)
			st[i][j] = max(st[i][j - 1], st[i + (1 << (j - 1))][j
 - 1]);
}
int get_max(int L, int R)
{
	int k = lg[R - L + 1] - 1;
	return max(st[L][k], st[R - (1 << k) + 1][k]);
}
```

tarjanLCA

```
int find(int x)
{
	if (fa[x] != x)
		fa[x] = find(fa[x]);
	return fa[x];
}

void tarjan(int x)
{
	vis[x] = 1;
	int X = find(x);
	for (auto y : v[x])
	{
		if (vis[y])continue;
		tarjan(y);
		int Y = find(y);
		if (X != Y)fa[Y] = X;
	}
	for (auto [y, id] : ask[x])
```

```
            if (vis[y])
                  ans[id] = find(y);
}

int main()
{
      n = read(), m = read(), s = read();
      for (int i = 1; i <= n; i++)
            fa[i] = i;
      for (int i = 1; i < n; i++)
      {
            int x = read(), y = read();
            v[x].push_back(y);
            v[y].push_back(x);
      }
      for (int i = 1; i <= m; i++)
      {
            int x = read(), y = read();
            ask[x].push_back({y, i});
            ask[y].push_back({x, i});
      }
      tarjan(s);
      for (int i = 1; i <= m; i++)
            printf("%d\n", ans[i]);
      return 0;
}
```

tarjan 缩点

```
void tarjan(int x)
{
      in[x] = 1;
      s.push(x);
      dfn[x] = low[x] = ++cnt;
      for (auto y : a[x])
      {
            if (!dfn[y])
            {
                  tarjan(y);
                  low[x] = min(low[x], low[y]);
            }
            else if (in[y])
                  low[x] = min(low[x], dfn[y]);
      }
      if (dfn[x] == low[x])
      {
            M++;
            while (1)
            {
```

```
                int X = s.top();
                s.pop();
                scc[X] = M;
                in[X] = 0;
                sum[M] += w[X];
                if (x == X)break;
            }
        }
}
void search(int x)
{
        if (dp[x])return;
        dp[x] = sum[x];
        int maxson = 0;
        for (auto y : v[x])
        {
            if (!dp[y])
                search(y);
            maxson = max(maxson, dp[y]);
        }
        dp[x] += maxson;
}
signed main()
{
        n = read(), m = read();
        for (int i = 1; i <= n; i++)
            w[i] = read();
        for (int i = 1; i <= m; i++)
        {
            int x = read(), y = read();
            a[x].push_back(y);
        }
        for (int i = 1; i <= n; i++)
            if (!dfn[i])
                tarjan(i);
        for (int i = 1; i <= n; i++)
            for (auto y : a[i])
                if (scc[i] != scc[y])
                    v[scc[i]].push_back(scc[y]);
        for (int i = 1; i <= M; i++)
            search(i);
        for (int i = 1; i <= M; i++)
            ans = max(ans, dp[i]);
        printf("%d\n", ans);
        return 0;
}
```

trie

```
struct Node {
	int son[27], ed, vis;
} trie[N];
void insert(string str)
{
	int pos = 1;
	int len = str.size();
	for (int i = 0; i < len; i++)
	{
		int x = str[i] - 'a';
		if (!trie[pos].son[x])
			trie[pos].son[x] = ++tot;
		pos = trie[pos].son[x];
	}
	trie[pos].ed = 1;
}
void work(string str)
{
	int pos = 1;
	int len = str.size();
	for (int i = 0; i < len; i++)
	{
		int x = str[i] - 'a';
		if (!trie[pos].son[x])
		{
			printf("WRONG\n");
			return;
		}
		pos = trie[pos].son[x];
	}
	if (!trie[pos].ed)
	{
		printf("WRONG\n");
		return;
	}
	else if (trie[pos].vis)
	{
		printf("REPEAT\n");
		return;
	}
	else
	{
		printf("OK\n");
		trie[pos].vis = 1;
	}
}
int main()
```

```
{
	int n = read();
	for (int i = 1; i <= n; i++)
	{
		string st;
		cin >> st;
		insert(st);
	}
	int m = read();
	while (m--)
	{
		string st;
		cin >> st;
		work(st);
	}
	return 0;
}
```

倍增 LCA

```
void dfs(int x, int fa)
{
	dep[x] = dep[fa] + 1;
	f[x][0] = fa;
	for (int i = 1; (1 << i) <= dep[x]; i++)
		f[x][i] = f[f[x][i - 1]][i - 1];
	for (auto y : a[x])
		if (y != fa)
			dfs(y, x);
}

int LCA(int x, int y)
{
	if (dep[x] < dep[y])
		swap(x, y);
	while (dep[x] != dep[y])
		x = f[x][lg[dep[x] - dep[y]] - 1];
	if (x == y)return x;
	for (int i = lg[dep[x]]; i >= 0; i--)
		if (f[x][i] != f[y][i])
			x = f[x][i], y = f[y][i];
	return f[x][0];
}
int getdis(int x, int y)
{
	return dep[x] + dep[y] - 2 * dep[LCA(x, y)];
}
void work()
{
```

```
	int n = read(), m = read(), root = read();
	for (int i = 1; i <= n - 1; i++)
	{
		int x = read(), y = read();
		a[x].push_back(y);
		a[y].push_back(x);
	}
	dfs(root, 0);
	while (m--)
	{
		int x = read(), y = read();
		printf("%d\n", LCA(x, y));
	}
}
void init()
{
	for (int i = 1; i < N; i++)
	{
		lg[i] = lg[i - 1];
		if (i == (1 << lg[i - 1]))lg[i]++;
	}
}
```

分块

```
signed main()
{
	int n = read(), m = read();
	int len = sqrt(n);
	int tot = (n + len - 1) / len;
	for (int i = 1; i <= n; i++)
	{
		a[i] = read();
		id[i] = (i - 1) / len + 1;
		sum[id[i]] += a[i];
	}
	for (int i = 1; i <= tot; i++)
	{
		L[i] = (i - 1) * len + 1;
		R[i] = i * len;
	}
	while (m--)
	{
		int p = read();
		if (p == 1)
		{
			int x = read(), y = read(), z = read();
			if (id[x] == id[y])
				for (int i = x; i <= y; i++)
```

```
                        a[i] += z, sum[id[i]] += z;
                else
                {
                        for (int i = x; i <= R[id[x]]; i++)
                                a[i] += z, sum[id[i]] += z;
                        for (int i = L[id[y]]; i <= y; i++)
                                a[i] += z, sum[id[i]] += z;
                        for (int i = id[x] + 1; i <= id[y] - 1; i++)
                                lazy[i] += z;
                }
        }
        else
        {
                int x = read(), y = read();
                int s = 0;
                if (id[x] == id[y])
                        for (int i = x; i <= y; i++)
                                s += a[i] + lazy[id[i]];
                else
                {
                        for (int i = x; i <= R[id[x]]; i++)
                                s += a[i] + lazy[id[i]];
                        for (int i = L[id[y]]; i <= y; i++)
                                s += a[i] + lazy[id[i]];
                        for (int i = id[x] + 1; i <= id[y] - 1; i++)
                                s += sum[i] + lazy[i] * len;
                }
                printf("%lld\n", s);
        }
    }
    return 0;
}
```

割点

```
void tarjan(int x, int fa)
{
    dfn[x] = low[x] = ++cnt;
    int son = 0;
    for (auto y : a[x])
    {
        if (!dfn[y])
        {
            tarjan(y, fa);
            low[x] = min(low[x], low[y]);
            if (low[y] >= dfn[x] && x != fa)
                gd[x] = 1;
            if (x == fa)
                son++;
```

```
            }
            low[x] = min(low[x], dfn[y]);
      }
      if (son >= 2 && x == fa)
            gd[x] = 1;
}
void work()
{
      int n = read(), m = read();
      for (int i = 1; i <= m; i++)
      {
            int x = read(), y = read();
            a[x].push_back(y);
            a[y].push_back(x);
      }
      for (int i = 1; i <= n; i++)
            if (!dfn[i])
                  tarjan(i, i);
      int ans = 0;
      for (int i = 1; i <= n; i++)
            if (gd[i])
                  ans++;
      printf("%d\n", ans);
      for (int i = 1; i <= n; i++)
            if (gd[i])
                  printf("%d ", i);
}
```

匈牙利

```
int find(int x)
{
      for (int i = 0; i < v[x].size(); i++)
      {
            int y = v[x][i];
            if (!vis[y])
            {
                  vis[y] = 1;
                  if (!match[y] || find(match[y])) {
                        match[y] = x;
                        return 1;
                  }
            }
      }
      return 0;
}
int main()
{
      scanf("%d%d%d", &n, &m, &e);
```

```
	for (int i = 1; i <= e; i++)
	{
		scanf("%d%d", &x, &y);
		if (x <= n && y <= m)
			v[x].push_back(y);
	}
	for (int i = 1; i <= n; i++)
	{
		memset(vis, 0, sizeof(vis));
		if (find(i))ans++;
	}
	cout << ans;
}
```

单调栈

```
void work()
{
	int n = read(), m = read();
	for (int i = 1; i <= n; i++)
		for (int j = 1; j <= m; j++)
		{
			a[i][j] = read();
			c[i][j] = 0;
		}
	for (int j = 1; j <= m; j++)
		for (int i = 1; i <= n; i++)
		{
			if (a[i][j] >= a[i - 1][j])
				c[i][j] = c[i - 1][j] + 1;
			else
				c[i][j] = 1;
		}
	int ans = 0;
	stack<int>s;
	for (int i = 1; i <= n; i++)
	{
		vector<int>val(m + 2), L(m + 2), R(m + 2);
		for (int j = 1; j <= m; j++)
			val[j] = c[i][j];
		val[0] = val[m + 1] = 0;
		while (s.size())s.pop();
		for (int j = 0; j <= m + 1; j++)
		{
			while (s.size() && val[s.top()] >= val[j])
				s.pop();
			if (s.size())
				L[j] = s.top();
			s.push(j);
```

```
            }
            while (s.size())s.pop();
            for (int j = m + 1; j >= 0; j--)
            {
                  while (s.size() && val[s.top()] >= val[j])
                        s.pop();
                  if (s.size())
                        R[j] = s.top();
                  s.push(j);
            }
            for (int j = 1; j <= m; j++)
                  ans = max(ans, val[j] * (R[j] - L[j] - 1));
      }
      printf("%d\n", ans);
}
```

可持久化 KMP

```
void add(char ch)
{
      int j = last;
      while (j && s[Next[j] + 1] != ch)
            j = pre[j];
      s[++last] = ch, j = Next[j] + 1;
      if (last == 1)
            Next[1] = pre[1] = 0;
      else if (s[j] == ch)
      {
            Next[last] = j;
            if (s[Next[j] + 1] == s[j + 1])
                  pre[last] = pre[j];
            else
                  pre[last] = j;
      }
      else
            Next[last] = pre[last] = 0;
}
void del(int len)
{
      last -= len;
}
void work()
{
      last = 0;
      string st;
      cin >> st;
      for (auto ch : st)
            add(ch);
      int n;
```

```
	cin >> n;
	while (n--)
	{
		string str;
		cin >> str;
		int len = str.size();
		for (auto ch : str)
		{
			add(ch);
			cout << Next[last] << " ";
		}
		cout << endl;
		del(len);
	}
}
```

启发式合并

```
int find(int x)
{
	if (fa[x] != x)
		fa[x] = find(fa[x]);
	return fa[x];
}
void merge(int x, int y)
{
	x = find(x), y = find(y);
	if (x == y)return;
	if (cb[x].size() > cb[y].size())
	{
		swap(x, y);
		swap(rb[x], rb[y]);
	}
	fa[x] = y;
	sz[y] += sz[x];
	for (auto tep : cb[x])
	{
		int v = find(tep);
		if (v == y)continue;
		rb[v].erase(x);
		rb[v].insert(y);
		cb[y].insert(v);
		if (rb[v].size() == 1)
			q.push({v, *rb[v].begin()});
	}
	cb[y].erase(x);
}
void work(int num)
{
```

```
	int n = read();
	for (int i = 1; i <= n; i++)
	{
		rb[i].clear(), cb[i].clear();
		fa[i] = i;
		sz[i] = 1;
	}
	for (int i = 1; i <= n; i++)
	{
		int m = read();
		while (m--)
		{
			int x = read();
			rb[i].insert(x);
			cb[x].insert(i);
		}
	}
	for (int i = 1; i <= n; i++)
		if (rb[i].size() == 1)
			q.push({i, *rb[i].begin()});
	while (q.size())
	{
		auto tep = q.front();
		q.pop();
		merge(tep.first, tep.second);
	}
	int ans = 0;
	for (int i = 1; i <= n; i++)
		ans = max(ans, sz[find(i)]);
	printf("Case #%d: %d\n", num, ans);
}
```

堆优化 DIJ

```
signed main()
{
	int n = read(), m = read(), s = read();
	for (int i = 1; i <= m; i++)
	{
		int x = read(), y = read(), z = read();
		a[x].emplace_back(y, z);
	}
	memset(d, 0x3f, sizeof(d));
	d[s] = 0;
	priority_queue<pii> q;
	q.push({0, s});
	while (!q.empty())
	{
		auto [val, x] = q.top();
```

```
            val = -val;
            q.pop();
            if (vis[x] || val > d[x])
                  continue;
            vis[x] = 1;
            for (auto [y, z] : a[x])
                  if (val + z < d[y])
                  {
                        d[y] = val + z;
                        q.push({ -d[y], y});
                  }
      }
      for (int i = 1; i <= n; i++)
            printf("%d ", d[i]);
      return 0;
}
```

堆优化 prim

```
signed main()
{
      int n = read(), m = read();
      vector<int>vis(n + 1);
      for (int i = 1; i <= m; i++)
      {
            int x = read(), y = read(), z = read();
            a[x].emplace_back(y, z);
            a[y].emplace_back(x, z);
      }
      memset(d, 0x7f, sizeof(d));
      d[1] = 0;
      int cnt = 0;
      int ans = 0;
      priority_queue<pair<int, int> >q;
      q.push({0, 1});
      while (q.size() && cnt < n)
      {
            auto [val, x] = q.top();
            q.pop();
            if (vis[x])continue;
            vis[x] = 1;
            cnt++;
            ans += -val;
            for (auto [y, z] : a[x])
                  if (d[y] > z)
                  {
                        d[y] = z;
                        if (!vis[y])
                              q.push({ -d[y], y});
```

```
                }
        }
        if (cnt == n)
                printf("%lld\n", ans);
        else
                puts("orz");
        return 0;
}
```

并查集

```
namespace UF
{
int fa[N], rank[N];
inline void init(int n)
{
        for (int i = 1; i <= n; ++i)
        {
                fa[i] = i;
                rank[i] = 1;
        }
}
int find(int x)
{
        return x == fa[x] ? x : (fa[x] = find(fa[x]));
}
inline void merge(int i, int j)
{
        int x = find(i), y = find(j);
        if (rank[x] <= rank[y])
                fa[x] = y;
        else
                fa[y] = x;
        if (rank[x] == rank[y] && x != y)
                rank[y]++;
}
inline bool same(int i, int j)
{
        return find(i) == find(j);
}
} // namespace UF
```

无汇源上下界可行流

```
struct Node
{
        int from, to, min, max;
} e[N];
```

```
inline int read()
{
	int X = 0, w = 0;
	char ch = 0;
	while (!isdigit(ch))
	{
		w |= ch == '-';
		ch = getchar();
	}
	while (isdigit(ch))
		X = (X << 3) + (X << 1) + (ch ^ 48), ch = getchar();
	return w ? -X : X;
}
inline void addage(int x, int y, int z)
{
	to[++cnt] = y;
	Next[cnt] = from[x];
	from[x] = cnt;
	val[cnt] = z;
}
inline void add(int x, int y, int mini, int maxn)
{
	addage(x, y, maxn - mini);
	addage(y, x, 0);
}
int bfs()
{
	memset(d, 0, sizeof(d));
	queue<int> q;
	q.push(S);
	d[S] = 1;
	while (q.size())
	{
		int x = q.front();
		q.pop();
		for (int i = from[x]; i; i = Next[i])
		{
			int y = to[i];
			if (val[i] && !d[y])
			{
				q.push(y);
				d[y] = d[x] + 1;
				if (y == T)
					return 1;
			}
		}
	}
	return 0;
}
```

```
int dinic(int x, int flow)
{
	if (x == T)
		return flow;
	int k, rest = flow;
	for (int i = from[x]; i && rest; i = Next[i])
	{
		int y = to[i];
		if (val[i] && d[y] == d[x] + 1)
		{
			k = dinic(y, min(rest, val[i]));
			if (!k)
				d[y] = 0;
			val[i] -= k;
			val[i ^ 1] += k;
			rest -= k;
		}
	}
	return flow - rest;
}
int dinic()
{
	int res = 0;
	while (bfs())
	{
		int tep = 0;
		while (tep = dinic(S, INF))
			res += tep;
	}
	return res;
}
void work()
{
	int n = read(), m = read();
	vector<int> delta(n + 1);
	S = 0, T = n + 1;
	for (int i = 1; i <= m; i++)
	{
		int a = read(), b = read(), c = read(), d = read();
		add(a, b, c, d);
		e[i] = {a, b, c, d};
		delta[a] -= c;
		delta[b] += c;
	}
	int sum = 0;
	for (int i = 1; i <= n; i++)
		if (delta[i] > 0)
		{
			sum += delta[i];
```

```
                add(S, i, 0, delta[i]);
            }
            else if (delta[i] < 0)
                add(i, T, 0, -delta[i]);
        int maxflow = dinic();
        if (maxflow != sum)
        {
            puts("NO");
            return;
        }
        puts("YES");
        for (int i = 1; i <= m; i++)
            printf("%d\n", e[i].min + val[i << 1 | 1]);
}
```

普通平衡树

```
struct Node {
        int v, rnd, son[2], size;
} tree[MAXN];

void PushUp(int k)
{
        tree[k].size = tree[tree[k].son[0]].size + tree[tree[k].son[1]].s
ize + 1;
}

int build(int v)
{
        ++cnt;
        tree[cnt].size = 1;
        tree[cnt].v = v;
        tree[cnt].rnd = rand();
        return cnt;
}

int merge(int x, int y)
{
        if (!x || !y)return x + y;
        if (tree[x].rnd < tree[y].rnd)
        {
            tree[x].son[1] = merge(tree[x].son[1], y);
            PushUp(x);
            return x;
        }
        else
        {
            tree[y].son[0] = merge(x, tree[y].son[0]);
            PushUp(y);
```

```
            return y;
        }
}

void split(int now, int k, int &x, int &y)
{
        if (!now)x = y = 0;
        else
        {
            if (tree[now].v <= k)
            {
                x = now;
                split(tree[now].son[1], k, tree[now].son[1], y);
            }
            else
            {
                y = now;
                split(tree[now].son[0], k, x, tree[now].son[0]);
            }
            PushUp(now);
        }
}

int kth(int now, int k)
{
        while (1)
        {
            if (k <= tree[tree[now].son[0]].size)
                now = tree[now].son[0];
            else
            {
                if (k == tree[tree[now].son[0]].size + 1)
                    return now;
                else
                {
                    k -= tree[tree[now].son[0]].size + 1;
                    now = tree[now].son[1];
                }
            }
        }
}

int main()
{
        srand(time(NULL));
        n = read();
        int x, y, z;
        while (n--)
        {
```

```
		int opt = read(), a = read();
		if (opt == 1)
		{
			split(root, a, x, y);
			root = merge(merge(x, build(a)), y);
		}//插入 x
		if (opt == 2)
		{
			split(root, a, x, z);
			split(x, a - 1, x, y);
			y = merge(tree[y].son[0], tree[y].son[1]);
			root = merge(merge(x, y), z);
		}//删除 x
		if (opt == 3)
		{
			split(root, a - 1, x, y);
			printf("%d\n", tree[x].size + 1);
			root = merge(x, y);
		}//查询 x 的排名
		if (opt == 4)
			printf("%d\n", tree[kth(root, a)].v);//查询排名为 x 的
数
		if (opt == 5)
		{
			split(root, a - 1, x, y);
			printf("%d\n", tree[kth(x, tree[x].size)].v);
			root = merge(x, y);
		}//求 x 前驱
		if (opt == 6)
		{
			split(root, a, x, y);
			printf("%d\n", tree[kth(y, 1)].v);
			root = merge(x, y);
		}//求 x 后缀
	}
	return 0;
}
```

最大权闭合子图

```
inline void addage(int x, int y, int z)
{
	to[++cnt] = y;
	Next[cnt] = from[x];
	from[x] = cnt;
	val[cnt] = z;
}
inline void add(int x, int y, int z)
{
```

```
	addage(x, y, z);
	addage(y, x, 0);
}
int bfs()
{
	queue<int> q;
	memset(d, 0, sizeof(d));
	d[S] = 1;
	cur[S] = from[S];
	q.push(S);
	while (q.size())
	{
		int x = q.front();
		q.pop();
		for (int i = from[x]; i; i = Next[i])
		{
			int y = to[i];
			if (val[i] && !d[y])
			{
				d[y] = d[x] + 1;
				cur[y] = from[y];
				q.push(y);
				if (y == T)
					return 1;
			}
		}
	}
	return 0;
}
int dinic(int x, int flow)
{
	if (x == T)
		return flow;
	int k, rest = flow;
	for (int i = from[x]; i && rest; i = Next[i])
	{
		int y = to[i];
		if (val[i] && d[y] == d[x] + 1)
		{
			k = dinic(y, min(rest, val[i]));
			if (!k)
				d[y] = 0;
			val[i] -= k;
			val[i ^ 1] += k;
			rest -= k;
		}
	}
	return flow - rest;
}
```

```
int dinic()
{
	int res = 0;
	while (bfs())
	{
		int tep;
		while (tep = dinic(S, INF))
			res += tep;
	}
	return res;
}
void work()
{
	int n = read(), m = read();
	S = 0, T = n + m + 1;
	int ans = 0;
	for (int i = 1; i <= n; i++)
	{
		int x = read();
		add(i + m, T, x); // 负权点到T
	}
	for (int i = 1; i <= m; i++)
	{
		int a = read(), b = read(), c = read();
		add(S, i, c);           // S 到正权点
		add(i, m + a, INF); // i 需要的依赖
		add(i, m + b, INF); // i 需要的依赖
		ans += c;               // 计算所有正权点的和
	}
	printf("%lld\n", ans - dinic()); //所有正权点的和-最小割
	return;
}
```

最小点权覆盖

```
inline void addage(int x, int y, int z)
{
	to[++cnt] = y;
	Next[cnt] = from[x];
	from[x] = cnt;
	val[cnt] = z;
}
inline void add(int x, int y, int z)
{
	addage(x, y, z);
	addage(y, x, 0);
}
int bfs()
{
```

```
	queue<int> q;
	memset(d, 0, sizeof(d));
	d[S] = 1;
	cur[S] = from[S];
	q.push(S);
	while (q.size())
	{
		int x = q.front();
		q.pop();
		for (int i = from[x]; i; i = Next[i])
		{
			int y = to[i];
			if (val[i] && !d[y])
			{
				d[y] = d[x] + 1;
				cur[y] = from[y];
				q.push(y);
				if (y == T)
					return 1;
			}
		}
	}
	return 0;
}
int dinic(int x, int flow)
{
	if (x == T)
		return flow;
	int k, rest = flow;
	for (int i = from[x]; i && rest; i = Next[i])
	{
		int y = to[i];
		if (val[i] && d[y] == d[x] + 1)
		{
			k = dinic(y, min(rest, val[i]));
			if (!k)
				d[y] = 0;
			val[i] -= k;
			val[i ^ 1] += k;
			rest -= k;
		}
	}
	return flow - rest;
}
int dinic()
{
	int res = 0;
	while (bfs())
	{
```

```
            int tep;
            while (tep = dinic(S, INF))
                res += tep;
        }
        return res;
}
void dfs(int x)
{
        vis[x] = 1;
        for (int i = from[x]; i; i = Next[i])
        {
            if (!val[i]) //在残量网络上 dfs
                continue;
            int y = to[i];
            if (!vis[y])
                dfs(y);
        }
}
void work()
{
        int n = read(), m = read();
        S = 0, T = 2 * n + 1;
        for (int i = 1; i <= n; i++)
        {
            int x = read();
            add(S, i, x); // S 到每个左点，边权为点权
        }
        for (int i = 1; i <= n; i++)
        {
            int x = read();
            add(i + n, T, x); //右点到 T
        }
        for (int i = 1; i <= m; i++)
        {
            int x = read(), y = read();
            add(y, x + n, INF); //左点到右点
        }
        printf("%d\n", dinic());
        dfs(S);
        int res = 0;
        for (int i = 2; i <= cnt; i += 2)
        {
            int y = to[i], x = to[i ^ 1];
            if (vis[x] && !vis[y]) //删点的数量
                res++;
        }
        printf("%d\n", res);
        for (int i = 2; i <= cnt; i += 2)
        {
```

```
            int y = to[i], x = to[i ^ 1];
            if (vis[x] && !vis[y])
            {
                  if (x == S)
                        printf("%d +\n", y);
                  if (y == T)
                        printf("%d -\n", x - n);
            }
      }
      return;
}
```

有汇源上下界最大流

```
inline void addage(int x, int y, int z)
{
      to[++cnt] = y;
      Next[cnt] = from[x];
      from[x] = cnt;
      val[cnt] = z;
}
inline void add(int x, int y, int mini, int maxn)
{
      addage(x, y, maxn - mini);
      addage(y, x, 0);
}
int bfs()
{
      memset(d, 0, sizeof(d));
      queue<int> q;
      q.push(S);
      d[S] = 1;
      while (q.size())
      {
            int x = q.front();
            q.pop();
            for (int i = from[x]; i; i = Next[i])
            {
                  int y = to[i];
                  if (val[i] && !d[y])
                  {
                        q.push(y);
                        d[y] = d[x] + 1;
                        if (y == T)
                              return 1;
                  }
            }
      }
      return 0;
```

```
}
int dinic(int x, int flow)
{
	if (x == T)
		return flow;
	int k, rest = flow;
	for (int i = from[x]; i && rest; i = Next[i])
	{
		int y = to[i];
		if (val[i] && d[y] == d[x] + 1)
		{
			k = dinic(y, min(rest, val[i]));
			if (!k)
				d[y] = 0;
			val[i] -= k;
			val[i ^ 1] += k;
			rest -= k;
		}
	}
	return flow - rest;
}
int dinic()
{
	int res = 0;
	while (bfs())
	{
		int tep = 0;
		while (tep = dinic(S, INF))
			res += tep;
	}
	return res;
}
void work()
{
	int n = read(), m = read(), s = read(), t = read();
	vector<int> delta(n + 1);
	S = 0, T = n + 1;
	for (int i = 1; i <= m; i++)
	{
		int a = read(), b = read(), c = read(), d = read();
		add(a, b, c, d);
		e[i] = {a, b, c, d};
		delta[a] -= c;
		delta[b] += c;
	}
	int sum = 0;
	for (int i = 1; i <= n; i++)
		if (delta[i] > 0)
		{
```

```
                sum += delta[i];
                add(S, i, 0, delta[i]);
            }
            else if (delta[i] < 0)
                add(i, T, 0, -delta[i]);
        add(t, s, 0, INF);
        int maxflow = dinic();
        if (maxflow != sum)
        {
            puts("No Solution");
            return;
        }
        int res = val[cnt];
        S = s, T = t;
        val[cnt] = val[cnt - 1] = 0;
        printf("%lld\n", res + dinic());
}
```

有汇源上下界最小流

```
inline void addage(int x, int y, int z)
{
    to[++cnt] = y;
    Next[cnt] = from[x];
    from[x] = cnt;
    val[cnt] = z;
}
inline void add(int x, int y, int mini, int maxn)
{
    addage(x, y, maxn - mini);
    addage(y, x, 0);
}
int bfs()
{
    queue<int> q;
    memset(d, 0, sizeof(d));
    d[S] = 1;
    cur[S] = from[S];
    q.push(S);
    while (q.size())
    {
        int x = q.front();
        q.pop();
        for (int i = from[x]; i; i = Next[i])
        {
            int y = to[i];
            if (val[i] && !d[y])
            {
                d[y] = d[x] + 1;
```

```
                        cur[y] = from[y];
                        q.push(y);
                        if (y == T)
                              return 1;
                  }
            }
      }
      return 0;
}
int dinic(int x, int flow)
{
      if (x == T)
            return flow;
      int k, rest = flow;
      for (int i = cur[x]; i && rest; i = Next[i])
      {
            cur[x] = i;
            int y = to[i];
            if (val[i] && d[y] == d[x] + 1)
            {
                  k = dinic(y, min(val[i], rest));
                  if (!k)
                        d[y] = 0;
                  rest -= k;
                  val[i] -= k;
                  val[i ^ 1] += k;
            }
      }
      return flow - rest;
}
int dinic()
{
      int res = 0;
      while (bfs())
      {
            int tep = 0;
            while (tep = dinic(S, INF))
                  res += tep;
      }
      return res;
}
void work()
{
      int n = read(), m = read(), s = read(), t = read();
      vector<int> delta(n + 1);
      S = 0, T = n + 1;
      for (int i = 1; i <= m; i++)
      {
            int a = read(), b = read(), c = read(), d = read();
```

```
            add(a, b, c, d);
            e[i] = {a, b, c, d};
            delta[a] -= c;
            delta[b] += c;
      }
      int sum = 0;
      for (int i = 1; i <= n; i++)
            if (delta[i] > 0)
            {
                  sum += delta[i];
                  add(S, i, 0, delta[i]);
            }
            else if (delta[i] < 0)
                  add(i, T, 0, -delta[i]);
      add(t, s, 0, INF);
      int maxflow = dinic();
      if (maxflow < sum)
      {
            puts("No Solution");
            return;
      }
      int res = val[cnt];
      S = t, T = s;
      val[cnt] = val[cnt - 1] = 0;
      printf("%d\n", res - dinic());
}
```

树上启发式合并

```
void dfs1(int x, int fa)
{
      sz[x] = 1;
      dep[x] = dep[fa] + 1;
      for (auto y : a[x])
      {
            if (y == fa)continue;
            dfs1(y, x);
            sz[x] += sz[y];
            if (sz[y] > sz[son[x]])
                  son[x] = y;
      }
}
void cal(int x, int fa, int opt)
{
      cnt[dep[x]][st[x] - 'a'] += opt;
      for (auto y : a[x])
            if (y != fa && y != flag)
                  cal(y, x, opt);
}
```

```
int check(int k)
{
	int p = 0;
	for (int i = 0; i < 26; i++)
		p += (cnt[k][i] % 2);
	return p <= 1;
}
void dfs2(int x, int fa, int opt)
{
	for (auto y : a[x])
		if (y != fa && y != son[x])
			dfs2(y, x, 0);
	if (son[x])
	{
		dfs2(son[x], x, 1);
		flag = son[x];
	}
	cal(x, fa, 1);
	flag = 0;
	for (auto [k, id] : ask[x])
		ans[id] = check(k);
	if (!opt)
		cal(x, fa, -1);
}
void work()
{
	int n = read(), m = read();
	for (int i = 2; i <= n; i++)
	{
		int x = read();
		a[x].push_back(i);
		a[i].push_back(x);
	}
	scanf("%s", st + 1);
	dfs1(1, 0);
	for (int i = 1; i <= m; i++)
	{
		int x = read(), y = read();
		ask[x].push_back({ y, i });
	}
	dfs2(1, 0, 0);
	for (int i = 1; i <= m; i++)
		if (ans[i])
			puts("Yes");
		else
			puts("No");
}
```

树状数组

```
inline int lowbit(int x)
{
	return x & -x;
}
void add(int x, int k)
{
	while (x <= n)
	{
		tree[x] += k;
		x += lowbit(x);
	}
}
int sum(int x)
{
	int s = 0;
	while (x)
	{
		s += tree[x];
		x -= lowbit(x);
	}
	return s;
}
```

树链剖分

```
void PushUp(int k)
{
	tree[k].w = (tree[k << 1].w + tree[k << 1 | 1].w) % P;
}

void PushDown(int k)
{
	if (!tree[k].f)return;
	int x = tree[k].f;
	tree[k].f = 0;
	tree[k << 1].f += x;
	tree[k << 1 | 1].f += x;
	tree[k << 1].w += x * (tree[k << 1].r - tree[k << 1].l + 1);
	tree[k << 1 | 1].w += x * (tree[k << 1 | 1].r - tree[k << 1 | 1].
l + 1);
	tree[k << 1].w %= P;
	tree[k << 1 | 1].w %= P;
}

void build(int k, int L, int R)
{
	tree[k].l = L;
```

```
	tree[k].r = R;
	if (L == R)
	{
		tree[k].w = a[L];
		return;
	}
	int mid = (L + R) >> 1;
	build(k << 1, L, mid);
	build(k << 1 | 1, mid + 1, R);
	PushUp(k);
}

void change(int k, int L, int R, int d)
{
	if (tree[k].l >= L && tree[k].r <= R)
	{
		tree[k].f += d;
		tree[k].w += d * (tree[k].r - tree[k].l + 1);
		tree[k].w %= P;
		return;
	}
	PushDown(k);
	int mid = (tree[k].l + tree[k].r) >> 1;
	if (mid >= L)change(k << 1, L, R, d);
	if (R > mid)change(k << 1 | 1, L, R, d);
	PushUp(k);
}

int search(int k, int L, int R)
{
	int ans = 0;
	if (tree[k].l >= L && tree[k].r <= R)
		return tree[k].w % P;
	PushDown(k);
	int mid = (tree[k].l + tree[k].r) >> 1;
	if (mid >= L)ans += search(k << 1, L, R);
	if (R > mid)ans += search(k << 1 | 1, L, R);
	return ans % P;
}

void dfs1(int x, int fa)
{
	dep[x] = dep[fa] + 1;
	f[x] = fa;
	sz[x] = 1;
	for (auto y : v[x])
	{
		if (y == f[x])continue;
		dfs1(y, x);
```

```
            sz[x] += sz[y];
            if (sz[y] > sz[son[x]])
                son[x] = y;
        }
}

void dfs2(int x, int topf)
{
        top[x] = topf;
        id[x] = ++cnt;
        a[cnt] = b[x];
        if (!son[x])return;
        dfs2(son[x], topf);
        for (auto y : v[x])
            if (!id[y])
                dfs2(y, y);
}

void path_change(int x, int y, int z)
{
        while (top[x] != top[y])
        {
            if (dep[top[x]] < dep[top[y]])swap(x, y);
            change(1, id[top[x]], id[x], z);
            x = f[top[x]];
        }
        if (dep[x] > dep[y])swap(x, y);
        change(1, id[x], id[y], z);
}

void path_search(int x, int y)
{
        int ans = 0;
        while (top[x] != top[y])
        {
            if (dep[top[x]] < dep[top[y]])swap(x, y);
            ans += search(1, id[top[x]], id[x]);
            ans %= P;
            x = f[top[x]];
        }
        if (dep[x] > dep[y])swap(x, y);
        ans += search(1, id[x], id[y]);
        printf("%lld\n", ans % P);
}

void tree_change(int x, int y)
{
        change(1, id[x], id[x] + sz[x] - 1, y);
}
```

```
void tree_search(int x)
{
	printf("%lld\n", search(1, id[x], id[x] + sz[x] - 1));
}

signed main()
{
	n = read(), m = read(), root = read(), P = read();
	for (int i = 1; i <= n; i++)
		b[i] = read();
	for (int i = 1; i < n; i++)
	{
		int x = read(), y = read();
		v[x].push_back(y);
		v[y].push_back(x);
	}
	dfs1(root, 0);
	dfs2(root, root);
	build(1, 1, n);
	for (int i = 1; i <= m; i++)
	{
		int opt = read();
		if (opt == 1)
		{
			int x = read(), y = read(), z = read();
			path_change(x, y, z);
		}
		else if (opt == 2)
		{
			int x = read(), y = read();
			path_search(x, y);
		}
		else if (opt == 3)
		{
			int x = read(), y = read();
			tree_change(x, y);
		}
		else
		{
			int x = read();
			tree_search(x);
		}
	}
	return 0;
}
```

模拟退火

```
struct Node
{
	int x, y, z;
} a[N];
int n;
double ansx, ansy, ansz, ans;
double down = 0.996;
double cal(double x, double y, double z)
{
	double tep = 0.0;
	for (int i = 1; i <= n; i++)
		tep = max(tep, sqrt((a[i].x - x) * (a[i].x - x) + (a[i].y -
 y) * (a[i].y - y) + (a[i].z - z) * (a[i].z - z)));
	return tep;
}
void SA()
{
	double t = 3000;
	while (t >= 1e-15)
	{
		double x1 = ansx + (rand() * 2 - RAND_MAX) * t;
		double y1 = ansy + (rand() * 2 - RAND_MAX) * t;
		double z1 = ansz + (rand() * 2 - RAND_MAX) * t;
		double ans1 = cal(x1, y1, z1);
		double d = ans1 - ans;
		if (d < 0)
		{
			ansx = x1;
			ansy = y1;
			ansz = z1;
			ans = ans1;
		}
		else if (exp(-d / t) * RAND_MAX > rand())
		{
			ansx = x1;
			ansy = y1;
			ansz = z1;
		}
		t *= down;
	}
}
int main()
{
	n = read();
	for (int i = 1; i <= n; i++)
	{
		a[i].x = read(), a[i].y = read(), a[i].z = read();
```

```
            ansx += a[i].x;
            ansy += a[i].y;
            ansz += a[i].z;
      }
      ansx /= n;
      ansy /= n;
      ansz /= n;
      ans = cal(ansx, ansy, ansz);
      SA(), SA(), SA(), SA(), SA();
      printf("%.10lf\n", ans);
      return 0;
}
```

点双

```
void tarjan(int x, int fa)
{
      dfn[x] = low[x] = ++cnt;
      S.push(x);
      int son = 0;
      for (auto y : a[x])
      {
            if (!dfn[y])
            {
                  son++;
                  tarjan(y, x);
                  low[x] = min(low[x], low[y]);
                  if (low[y] >= dfn[x])
                  {
                        ans[++bcc].push_back(x);
                        while (ans[bcc].back() != y)
                        {
                              ans[bcc].push_back(S.top());
                              S.pop();
                        }
                  }
            }
            else if (y != fa)
                  low[x] = min(low[x], dfn[y]);
      }
      if (fa == 0 && son == 0)
            ans[++bcc].push_back(x);
}
void work()
{
      int n = read(), m = read();
      for (int i = 1; i <= m; i++)
      {
            int x = read(), y = read();
```

```
            a[x].push_back(y);
            a[y].push_back(x);
      }
      for (int i = 1; i <= n; i++)
            if (!dfn[i])
                  tarjan(i, 0);
      printf("%d\n", bcc);
      for (int i = 1; i <= bcc; i++)
      {
            printf("%d ", ans[i].size());
            for (auto x : ans[i])
                  printf("%d ", x);
            puts("");
      }
}
```

矩阵快速幂

```
struct mat
{
      int M[N][N];
} ori, ans;
mat mul(mat a, mat b, int n)
{
      mat tep;
      for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
            {
                  tem.M[i][j] = 0;
                  for (int k = 1; k <= n; k++)
                        tep.M[i][j] = (tep.M[i][j] + (a.M[i][k] * b.M[k]
[j]) % P) % P;
            }
      return tep;
}
void qpow_mat(int n, int k)
{
      for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                  ans.M[i][j] = (i == j);
      while (k)
      {
            if (k & 1)
                  ans = mul(ans, ori, n);
            ori = mul(ori, ori, n);
            k >>= 1;
      }
}
signed main()
```

```
{
	int n = read(), k = read();
	for (int i = 1; i <= n; i++)
		for (int j = 1; j <= n; j++)
			ori.M[i][j] = read();
	qpow_mat(n, k);
	for (int i = 1; i <= n; i++)
	{
		for (int j = 1; j <= n; j++)
			printf("%lld ", ans.M[i][j]);
		printf("\n");
	}
	return 0;
}
```

米勒罗宾

```
int qpow(int a, int n, int p)
{
	int ans = 1;
	while (n)
	{
		if (n & 1)
			ans = (__int128)ans * a % p;
		a = (__int128)a * a % p;
		n >>= 1;
	}
	return ans;
}
int is_prime(int x)
{
	if (x < 3)
		return x == 2;
	if (x % 2 == 0)
		return 0;
	int A[] = {2, 325, 9375, 28178, 450775, 9780504, 1795265022}, d =
 x - 1, r = 0;
	while (d % 2 == 0)
		d /= 2, ++r;
	for (auto a : A)
	{
		int v = qpow(a, d, x);
		if (v <= 1 || v == x - 1)
			continue;
		for (int i = 0; i < r; ++i)
		{
			v = (__int128)v * v % x;
			if (v == x - 1 && i != r - 1)
			{
```

```
                        v = 1;
                        break;
                }
                if (v == 1)
                        return 0;
        }
        if (v != 1)
                return 0;
    }
    return 1;
}
```

线性乘法逆元

```
int main()
{
    int n = read(), p = read();
    inv[1] = 1;
    printf("1\n");
    for (int i = 2; i <= n; i++)
    {
        inv[i] = (LL)(p - p / i) * inv[p % i] % p;
        printf("%d\n", inv[i]);
    }
    return 0;
}
```

线性筛

```
void init(int n)
{
    vis[1] = 1;
    for (int i = 2; i <= n; i++)
    {
        if (!vis[i])prime.push_back(i);
        for (auto x : prime)
        {
            if (x * i > n)break;
            vis[x * i] = 1;
            if (i % x == 0)break;
        }
    }
}
```

线段树

```
void PushUp(int k)
{
    tree[k].w = tree[k << 1].w + tree[k << 1 | 1].w;
```

```
}
void PushDown(int k)
{
	if (!tree[k].f)
		return;
	int x = tree[k].f;
	tree[k].f = 0;
	tree[k << 1].f += x;
	tree[k << 1 | 1].f += x;
	tree[k << 1].w += x * (tree[k << 1].r - tree[k << 1].l + 1);
	tree[k << 1 | 1].w += x * (tree[k << 1 | 1].r - tree[k << 1 | 1].
l + 1);
}
void build(int k, int L, int R)
{
	tree[k].l = L;
	tree[k].r = R;
	if (L == R)
	{
		tree[k].w = read();
		return;
	}
	int mid = (L + R) >> 1;
	build(k << 1, L, mid);
	build(k << 1 | 1, mid + 1, R);
	PushUp(k);
}
void change(int k, int L, int R, int x)
{
	if (tree[k].l >= L && tree[k].r <= R)
	{
		tree[k].f += x;
		tree[k].w += x * (tree[k].r - tree[k].l + 1);
		return;
	}
	PushDown(k);
	int mid = (tree[k].l + tree[k].r) >> 1;
	if (mid >= L)
		change(k << 1, L, R, x);
	if (mid < R)
		change(k << 1 | 1, L, R, x);
	PushUp(k);
}
LL query(int k, int L, int R)
{
	if (tree[k].l >= L && tree[k].r <= R)
		return tree[k].w;
	PushDown(k);
	LL sum = 0;
```

```
	int mid = (tree[k].l + tree[k].r) >> 1;
	if (mid >= L)
		sum += query(k << 1, L, R);
	if (mid < R)
		sum += query(k << 1 | 1, L, R);
	return sum;
}
void work()
{
	int n = read(), m = read();
	build(1, 1, n);
	while (m--)
	{
		int opt = read();
		if (opt == 1)
		{
			int x = read(), y = read(), z = read();
			change(1, x, y, z);
		}
		else
		{
			int x = read(), y = read();
			printf("%lld\n", query(1, x, y));
		}
	}
}
```

组合数

```
int qpow(int x, int y)
{
	int res = 1ll;
	x %= P;
	while (y)
	{
		if (y & 1ll)
			res = res * x % P;
		x = x * x % P;
		y >>= 1ll;
	}
	return res;
}
void init()
{
	jc[0] = 1;
	for (int i = 1; i < N; i++)
		jc[i] = (jc[i - 1] * i) % P;
	jc_inv[N - 1] = qpow(jc[N - 1], P - 2, P);
	for (int i = N - 1; i >= 1; i--)
```

```
            jc_inv[i - 1] = (jc_inv[i] * i) % P;
}
int C(int x, int y)
{
      if (x < y || y < 0)
            return 0;
      if (y == 0)
            return 1;
      return      (jc[x] * jc_inv[y] % P) * jc_inv[x - y] % P;
}
```

莫比乌斯反演

```
void init(int n)
{
      vis[1] = 1;
      mu[1] = 1;
      for (int i = 2; i <= n; i++)
      {
            if (!vis[i])
                  prime.push_back(i), mu[i] = -1;
            for (auto x : prime)
            {
                  if (x * i > n)
                        break;
                  vis[x * i] = 1;
                  if (i % x == 0)
                        break;
                  mu[x * i] = -mu[i];
            }
      }
      for (int i = 1; i <= n; i++)
            sum[i] = sum[i - 1] + mu[i];
}
int get(int a, int b)
{
      int res = 0;
      for (int L = 1, R; L <= min(a, b); L = R + 1)
      {
            R = min(a / (a / L), b / (b / L));
            res += (sum[R] - sum[L - 1]) * ((a / k) / L) * ((b / k) /
L);
      }
      return res;
}
void work()
{
      a = read(), b = read(), c = read(), d = read(), k = read();
      printf("%d\n", get(b, d) - get(a - 1, d) - get(b, c - 1) + get(a
```

```
- 1, c - 1));
}
```

莫队

```
inline int cmp(Node x, Node y)
{
	if (x.pos != y.pos)return x.pos < y.pos;
	if (x.pos % 2)return x.r < y.r;
	else return x.r > y.r;
}
void add(int pos)
{
	ans_tep += 2 * cnt[a[pos]] + 1;
	cnt[a[pos]]++;
}
void remove(int pos)
{
	ans_tep += 1 - cnt[a[pos]] * 2;
	cnt[a[pos]]--;
}
int main()
{
	n = read(), m = read(), k = read();
	len = (int)sqrt(n);
	for (int i = 1; i <= n; i++)
		a[i] = read();
	for (int i = 1; i <= m; i++)
	{
		ask[i].l = read(), ask[i].r = read();
		ask[i].id = i;
		ask[i].pos = (ask[i].l - 1) / len + 1;
	}
	sort(ask + 1, ask + m + 1, cmp);
	int L = 0, R = 0;
	for (int i = 1; i <= m; i++)
	{
		while (L < ask[i].l)remove(L++);
		while (L > ask[i].l)add(--L);
		while (R > ask[i].r)remove(R--);
		while (R < ask[i].r)add(++R);
		ans[ask[i].id] = ans_tep;
	}
	for (int i = 1; i <= m; i++)
		printf("%lld\n", ans[i] - 1);
	return 0;
}
```

边双求割边

```
void tarjan(int x, int fa)
{
	dfn[x] = low[x] = ++cnt;
	for (auto [y, z] : a[x])
	{
		if (!dfn[y])
		{
			tarjan(y, x);
			low[x] = min(low[x], low[y]);
			if (low[y] > dfn[x])
				gb[z] = 1;
		}
		else if (y != fa)
			low[x] = min(low[x], dfn[y]);
	}
}
void dfs(int x)
{
	id[x] = dcc;
	for (auto [y, z] : a[x])
		if (!gb[z] && !id[y])
			dfs(y);
}
void work()
{
	int n = read(), m = read();
	for (int i = 1; i <= m; i++)
	{
		int x = read(), y = read();
		a[x].push_back({y, i});
		a[y].push_back({x, i});
	}
	for (int i = 1; i <= n; i++)
		if (!dfn[i])
			tarjan(i, 0);
	for (int i = 1; i <= n; i++)
		if (!id[i])
		{
			dcc++;
			dfs(i);
		}
	for (int i = 1; i <= n; i++)
		ans[id[i]].push_back(i);
	printf("%d\n", dcc);
	for (int i = 1; i <= dcc; i++)
	{
		printf("%d ", ans[i].size());
```

```
            for (auto x : ans[i])
                  printf("%d ", x);
            puts("");
      }
}
```

树状数组求逆序对

```
int low(int x)
{
      return x & -x;
}
void add(int x)
{
      for (; x <= cnt; x += low(x))
            tree[x]++;
}
int sum(int x)
{
      int s = 0;
      for (; x != 0; x -= low(x))
            s += tree[x];
      return s;
}
signed main()
{
      int n = read();
      vector<int>v;
      for (int i = 1; i <= n; i++)
      {
            a[i] = read();
            v.push_back(a[i]);
      }
      sort(v.begin(), v.end());
      map<int, int>M;
      for (auto x : v)
            if (M.find(x) == M.end())
                  M[x] = ++cnt;
      int ans = 0;
      for (int i = 1; i <= n; i++)
      {
            add(M[a[i]]);
            ans += i - sum(M[a[i]]);
      }
      printf("%lld\n", ans);
      return 0;
}
```

LiYue University
Genshin Impact Science

Higher Elemental Theory

Author
Chen Free

September 4, 2022