树链剖分

```cpp
// 从 1 开始
#include <algorithm>
#include <cmath>
#include <cstdio>
#include <cstring>
#include <iostream>
#define ll long long
using namespace std;
ll p;
int n, m, root, k, x, y, cnt, z, s;
int w[2000010], head[2000010], dep[2000010], f[2000010],
tot[2000010],
    son[2000010];
int id[2000010], ww[2000010], top[2000010];

struct c {  // 链式前向星节点
  int to, next; // 这条边终点为 to, 同起点的下一条边索引为 next
} a[2000100];
struct cc {  // 线段树节点
  int x, y, l, add, size, w;
} t[2000100];

void add(int x, int y) {
  a[++k].to = y;
  a[k].next = head[x];
  head[x] = k; // head 一直指向最后输入的边
}

void dfs1(int x, int fa) {
  dep[x] = dep[fa] + 1;  // 深度加一
  f[x] = fa;             // 找到 x 的父节点
  tot[x] = 1;            // x 子树现在只有自己一个节点
  int maxn = -1;         // 临时的重链大小
  for (int i = head[x]; i; i = a[i].next) {
    int y = a[i].to;  // x 到 y 有一条边
    if (y == fa) continue;
```

```
    dfs1(y, x);
    tot[x] += tot[y];  // 统计子树大小
    if (tot[y] > maxn) {
      maxn = tot[y];
      son[x] = y;  // 更新重儿子
    }
  }
}
void dfs2(int x, int topf) {  // 当前在 x, 当前链顶端 topf
  id[x] = ++cnt;              // 重新映射的节点编号
  ww[cnt] = w[x];             // 节点初始值用来建树
  top[x] = topf;              // 节点链的顶端
  if (!son[x]) return;        // 到头
  dfs2(son[x], topf);         // 先走重儿子
  for (int i = head[x]; i; i = a[i].next) {
    int y = a[i].to;
    if (!id[y]) dfs2(y, y);  // 轻儿子自己作为链顶端
  }
}

void up(int k) { t[k].w = (t[k * 2].w + t[k * 2 + 1].w) % p; }
void pushdown(int k) {
  if (!t[k].add) return;
  t[k * 2].add = (t[k * 2].add + t[k].add) % p;
  t[k * 2 + 1].add = (t[k * 2 + 1].add + t[k].add) % p;
  t[k * 2].w = (t[k * 2].w + t[k * 2].size * t[k].add) % p;
  t[k * 2 + 1].w = (t[k * 2 + 1].w + t[k * 2 + 1].size * t[k].add)
% p;
  t[k].add = 0;
}
void build(int k, int l, int r) {
  t[k].x = l, t[k].y = r, t[k].size = r - l + 1;
  if (l == r) {
    t[k].w = ww[l];
    return;
  }
  int mid = (l + r) >> 1;
  build(k * 2, l, mid);
  build(k * 2 + 1, mid + 1, r);
  up(k);
}
void pushadd(int k, int l, int r, int v) {
```

```
  if (t[k].x >= l && t[k].y <= r) {
    t[k].w += t[k].size * v;
    t[k].add += v;
    return;
  }
  int mid = (t[k].x + t[k].y) / 2;
  pushdown(k);
  if (l <= mid) pushadd(k * 2, l, r, v);
  if (r > mid) pushadd(k * 2 + 1, l, r, v);
  up(k);
}
void Treeadd(int x, int y, int v) {              // x 到 y 路径加 v
  while (top[x] != top[y]) {                     // 当不在同一链上时
    if (dep[top[x]] < dep[top[y]]) swap(x, y);  // 使 x 为更深点
    pushadd(1, id[top[x]], id[x], v);  // 更新更深节点到链顶端的值
    x = f[top[x]];                     // x 跳到顶端
  }
  if (dep[x] > dep[y]) swap(x, y);  // 在同一链上, 使 x 为更深
  pushadd(1, id[x], id[y], v);      // 更新 xy 之间的值
}
int pushsum(int k, int l, int r) {
  int ans = 0;
  if (t[k].x >= l && t[k].y <= r) return t[k].w;
  int mid = (t[k].x + t[k].y) / 2;
  pushdown(k);
  if (l <= mid) ans = (ans + pushsum(k * 2, l, r)) % p;
  if (r > mid) ans = (ans + pushsum(k * 2 + 1, l, r)) % p;
  return ans;
}
int Treesum(int x, int y) {  // x 到 y 路径和
  int ans = 0;
  while (top[x] != top[y]) {                     // 当不在同一链上
    if (dep[top[x]] < dep[top[y]]) swap(x, y);  // 使 x 更深
    ans =
        (ans + pushsum(1, id[top[x]], id[x])) % p;  // 加上更深节点到
顶端的区间
    x = f[top[x]];                                  // 跳到顶端
  }
  if (dep[x] > dep[y]) swap(x, y);
  ans = (ans + pushsum(1, id[x], id[y])) % p;  // 加上区间和
  return ans;
}
```

```
int main() {
  // n 个节点，m次 操作，根节点 root，结果取模 p
  scanf("%d%d%d%d", &n, &m, &root, &p);
  for (int i = 1; i <= n; i++) scanf("%d", &w[i]);  // 节点初始值
  for (int i = 1; i <= n - 1; i++) {  // 链式前向星
    scanf("%d%d", &x, &y); add(x, y); add(y, x);
  }
  dfs1(root, 0); dfs2(root, root); build(1, 1, n);
  for (int i = 1; i <= m; i++) {
    scanf("%d", &s);
    if (s == 1) {  // x 到 y 的最短路径都加 z
      scanf("%d%d%d", &x, &y, &z);
      Treeadd(x, y, z % p);
    }
    if (s == 2) {  // x 到 y 的最短路径权值和
      scanf("%d%d", &x, &y);
      printf("%d\n", Treesum(x, y));
    }
    if (s == 3) {  // 根节点 x 的子树都加 z
      scanf("%d%d", &x, &z);
      pushadd(1, id[x], id[x] + tot[x] - 1, z % p);
    }
    if (s == 4) {  // 根节点 x 的子树权值和
      scanf("%d", &x);
      printf("%d\n", pushsum(1, id[x], id[x] + tot[x] - 1));
    }
  }
  return 1;
}
/*
8 10 2 448348
458 718 447 857 633 264 238 944
1 2
2 3
3 4
6 2
1 5
5 7
8 6
3 7 611
4 6
```

```
3 1 267
3 2 111
1 6 3 153
3 7 673
4 8
2 6 1
4 7
3 4 228

1208
1055
2346
1900
*/
```