modal

目录

线段树合并

```cpp
#include <bits/stdc++.h>
using i64 = long long;
struct data {
  int u, cnt;
  friend bool operator < (data a, data b) {
    return (a.cnt == b.cnt) ? a.u > b.u : a.cnt < b.cnt;
  }
  friend data operator + (data a, data b) {
    return (data){a.u, a.cnt + b.cnt};
  }
};
const int N = 1e5;
void solve() {
  int n, m;
  std::cin >> n >> m;
  std::vector<std::vector<int> > e(n + 1);
  for (int i = 1; i < n; i ++) {
    int u, v;
    std::cin >> u >> v;
    e[u].push_back(v);
    e[v].push_back(u);
  }
  std::vector<std::vector<data> > on(n + 1);
  int cnt = n; // 已经有 n 个节点
  std::vector<std::array<int, 2> > c(n * 128 + 1);
  std::vector<data> t(n * 128 + 1);
  auto insert = [&](auto self, int u, int l, int r, data x) -> void {
    if (l == r) { t[u] = x + t[u]; return ; }
    int mid = l + r >> 1;
    if (x.u <= mid) self(self, c[u][0] = c[u][0] ? c[u][0] : ++ cnt, l, mid, x);
    else self(self, c[u][1] = c[u][1] ? c[u][1] : ++ cnt, mid + 1, r, x);
    t[u] = std::max(t[c[u][0]], t[c[u][1]]);
  };
  auto merge = [&](auto self, int u1, int u2, int l, int r) -> void {
    if (l == r) { t[u1] = t[u1] + t[u2]; return ; }
    int mid = l + r >> 1;
    if (c[u1][0] && c[u2][0]) { self(self, c[u1][0], c[u2][0], l, mid); }
    else if (c[u2][0]) { c[u1][0] = c[u2][0]; }
    if (c[u1][1] && c[u2][1]) { self(self, c[u1][1], c[u2][1], mid + 1, r); }
    else if (c[u2][1]) { c[u1][1] = c[u2][1]; }
    t[u1] = std::max(t[c[u1][0]], t[c[u1][1]]);
  };

  std::vector<std::vector<int> > fa(19, std::vector<int>(n + 1));
```

```cpp
std::vector<int> dep(n + 1);
auto dfs1 = [&](auto self, int u, int p) -> void {
   fa[0][u] = p;
   dep[u] = dep[p] + 1;
   for (auto v : e[u]) {
      if (v == p) continue;
      self(self, v, u);
   }
};
dfs1(dfs1, 1, 0);
for (int j = 1; j <= 18; j ++) {
   for (int i = 1; i <= n; i ++) {
      fa[j][i] = fa[j - 1][fa[j - 1][i]];
   }
}
auto lca = [&](int u, int v) -> int {
   if (dep[u] < dep[v]) std::swap(u, v);
   for (int d = dep[u] - dep[v], i = 18; i >= 0; i --) {
      if (d >> i & 1) u = fa[i][u];
   }
   if (u == v) return u;
   for (int i = 18; i >= 0; i --) {
      if (fa[i][u] != fa[i][v]) {
         u = fa[i][u], v = fa[i][v];
      }
   }
   return fa[0][u];
};
for (int i = 1; i <= m; i ++) {
   int u, v, w;
   std::cin >> u >> v >> w;
   int o = lca(u, v);
   on[u].push_back((data){w, 1});
   on[v].push_back((data){w, 1});
   on[o].push_back((data){w, -1});
   on[fa[0][o]].push_back((data){w, -1});
}
std::vector<int> ans(n + 1);
auto dfs2 = [&](auto self, int u) -> void {
   for (auto v : e[u]) {
      if (v == fa[0][u]) continue;
      self(self, v);
      merge(merge, u, v, 0, N);
   }
   for (auto it : on[u]) {
      insert(insert, u, 0, N, it);
   }
```

```
    ans[u] = t[u].u;
  };
  dfs2(dfs2, 1);
  for (int i = 1; i <= n; i ++) {
    std::cout << ans[i] << '\n';
  }
}
int main() {
  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);
  int t = 1;
  //std::cin >> t;
  while (t--) {
    solve();
  }
}
```

线段树分裂

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int MAXN=200010;
int n,m,tot,cnt,seq=1,op,x,y,z,bac[MAXN<<5],ch[MAXN<<5][2],rt[MAXN];
ll val[MAXN<<5];
int newnod () {return (cnt?bac[cnt--]:++tot);}
void del (int p) {
  bac[++cnt]=p,ch[p][0]=ch[p][1]=val[p]=0;
  return;
}
void modify (int &p,int l,int r,int pos,int v) {
  if (!p) {p=newnod();}
  val[p]+=v;
  if (l==r) {return;}
  int mid=(l+r)>>1;
  if (pos<=mid) {modify(ch[p][0],l,mid,pos,v);}
  else {modify(ch[p][1],mid+1,r,pos,v);}
  return;
}
ll query (int p,int l,int r,int xl,int xr) {
  if (xr<l||r<xl) {return 0;}
  if (xl<=l&&r<=xr) {return val[p];}
  int mid=(l+r)>>1;
  return query(ch[p][0],l,mid,xl,xr)+query(ch[p][1],mid+1,r,xl,xr);
}
```

```
int kth (int p,int l,int r,int k) {
   if (l==r) {return l;}
   int mid=(l+r)>>1;
   if (val[ch[p][0]]>=k) {return kth(ch[p][0],l,mid,k);}
   else {return kth(ch[p][1],mid+1,r,k-val[ch[p][0]]);}
}
int merge (int x,int y) {
   if (!x||!y) {return x+y;}
   val[x]+=val[y];
   ch[x][0]=merge(ch[x][0],ch[y][0]);
   ch[x][1]=merge(ch[x][1],ch[y][1]);
   del(y);
   return x;
}
void split (int x,int &y,ll k) { // 分裂出 x 为原 x 的前 k 个
   if (x==0) {return;}
   y=newnod();
   ll v=val[ch[x][0]];
   if (k>v) {split(ch[x][1],ch[y][1],k-v);}
   else {swap(ch[x][1],ch[y][1]);}
   if (k<v) {split(ch[x][0],ch[y][0],k);}
   val[y]=val[x]-k;
   val[x]=k;
   return;
}
int main () {
   scanf("%d%d",&n,&m);
   for (int i=1;i<=n;i++) {
      scanf("%d",&x);
      modify(rt[1],1,n,i,x);
   }
   for (int i=1;i<=m;i++) {
      scanf("%d",&op);
      if (op==0) {
         scanf("%d%d%d",&x,&y,&z);
         ll k1=query(rt[x],1,n,1,z),k2=query(rt[x],1,n,y,z);
         int tmp=0;
         split(rt[x],rt[++seq],k1-k2);
         split(rt[seq],tmp,k2);
         rt[x]=merge(rt[x],tmp);
      } else if (op==1) {
         scanf("%d%d",&x,&y);
         rt[x]=merge(rt[x],rt[y]);
      } else if (op==2) {
         scanf("%d%d%d",&x,&y,&z);
         modify(rt[x],1,n,z,y);
      } else if (op==3) {
         scanf("%d%d%d",&x,&y,&z);
         printf("%lld\n",query(rt[x],1,n,y,z));
      } else if (op==4) {
         scanf("%d%d",&x,&y);
         if (val[rt[x]]<y) {printf("-1\n");continue;}
         printf("%d\n",kth(rt[x],1,n,y));
      }
   }
   return 0;
}
```

全子序列最小值之和

```
#include <bits/stdc++.h>
using i64 = long long;
const int inf = 1e9 + 7;
void solve() {
  int n, m;
  std::cin >> n >> m;
  std::vector<int> a(n + 2, inf);
  std::vector<std::vector<int> > f(17, std::vector<int>(n + 1));
  for (int i = 1; i <= n; i ++) {
    std::cin >> a[i];
    f[0][i] = i;
  }
  auto cmp = [&](int x, int y) -> int {
    return a[x] > a[y] ? y : x;
  };
  for (int j = 1; j <= std::__lg(n); j ++) {
    for (int i = 1; i <= n; i ++) {
      if (i + (1 << j) - 1 > n) break;
      f[j][i] = cmp(f[j - 1][i], f[j - 1][i + (1 << j - 1)]);
    }
  }
  auto rmq = [&](int l, int r) -> int {
    int x = std::__lg(r - l + 1);
    return cmp(f[x][l], f[x][r - (1 << x) + 1]);
  };

  std::vector<int> st, pre(n + 2), suf(n + 2);
  st.push_back(0);
  for (int i = 1; i <= n; i ++) {
    while (st.size() > 1 && a[st.back()] > a[i]) {
      suf[st.back()] = i;
      st.pop_back();
```

```cpp
        }
        pre[i] = st.back();
        st.push_back(i);
    }
    while (st.size() > 1) {
        int u = st.back();
        st.pop_back();
        pre[u] = st.back();
        if (st.size() > 1) suf[st.back()] = n + 1;
    }
    std::vector<i64> fr(n + 2), fl(n + 2), gr(n + 2), gl(n + 2);
    for (int i = 1; i <= n; i ++) {
        fr[i] = fr[pre[i]] + 1ll * a[i] * (i - pre[i]);
        gr[i] = gr[i - 1] + fr[i];
    }
    for (int i = n; i >= 1; i --) {
        fl[i] = fl[suf[i]] + 1ll * a[i] * (suf[i] - i);
        gl[i] = gl[i + 1] + fl[i];
    }
    for (int i = 1; i <= m; i ++) {
        int l, r;
        std::cin >> l >> r;
        int p = rmq(l, r);
        i64 ans = 1ll * a[p] * (p - l + 1) * (r - p + 1) +
        gr[r] - gr[p] - fr[p] * (r - p) +
        gl[l] - gl[p] - fl[p] * (p - l);
        std::cout << ans << '\n';
    }
}
int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int t = 1;
    //std::cin >> t;
    while (t--) {
        solve();
    }
}
```