badcw's template

WUST

So Like Coding? You Baldy

May 30, 2019

Contents

0 2019New

0.1 2019New

0.1.1 mint

```
const int mod = 998244353;

struct mint {
  int n;
  mint(int n_ = 0) : n(n_) {}
};

mint operator+(mint a, mint b) { return (a.n += b.n) >= mod ? a.n - mod : a.n; }
mint operator-(mint a, mint b) { return (a.n -= b.n) < 0 ? a.n + mod : a.n; }
mint operator*(mint a, mint b) { return 1LL * a.n * b.n % mod; }
mint &operator+=(mint &a, mint b) { return a = a + b; }
mint &operator-=(mint &a, mint b) { return a = a - b; }
mint &operator*=(mint &a, mint b) { return a = a * b; }
ostream &operator<<(ostream &o, mint a) { return o << a.n; }

mint dp[20][2][2][2][1 << 10];
mint dp2[20][2][2][2][1 << 10];
```

0.1.2 不重叠区间贪心

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;

const int maxn = 5e5+5;
pair<int, int> a[maxn];
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i].second >> a[i].first;
    }
    sort(a + 1, a + 1 + n);
    int res = 1;
    int tmp = a[1].first;
//    printf("%d %d\n", a[1].second, a[1].first);
    for (int i = 2; i <= n; ++i) {
        if (a[i].second > tmp) {
            res ++;
//            printf("%d %d\n", a[i].second, a[i].first);
            tmp = a[i].first;
        }
    }
    printf("%d\n", res);
    return 0;
}
```

0.1.3 主席树区间 k 大

```
/*************************************************************************
    > File Name: a.cpp
    > Author: badcw
```

```
4       > Mail: 952223482@qq.com
5       > Created Time: 2018年07月21日 星期六 08时47分54秒
6    ************************************************************************/
7
8   #include <bits/stdc++.h>
9   #define ll long long
10  using namespace std;
11
12  const int maxn = 100005;
13  int n, m;
14  int a[maxn];
15  int root[maxn];
16  int cnt = 0;
17  vector<int> b;
18  struct node {
19      int l, r, val;
20  }p[maxn * 40];
21
22  void update(int l, int r, int pre, int &now, int pos) {
23      now = ++cnt;
24      p[now] = p[pre];
25      p[now].val++;
26      if (l == r) {
27          return;
28      }
29      int mid = l + r >> 1;
30      if (pos <= mid) update(l, mid, p[pre].l, p[now].l, pos);
31      else update(mid + 1, r, p[pre].r, p[now].r, pos);
32  }
33
34  int query(int l, int r, int x, int y, int k) {
35      if (l == r) return b[l - 1];
36      int mid = l + r >> 1;
37      int temp = p[p[y].l].val - p[p[x].l].val;
38      if (k <= temp) return query(l, mid, p[x].l, p[y].l, k);
39      return query(mid + 1, r, p[x].r, p[y].r, k - temp);
40  }
41
42  int main(int argc,char *argv[])
43  {
44      while (scanf("%d%d", &n, &m) != EOF) {
45          b.clear();
46          cnt = 0;
47          for (int i = 1; i <= n; ++i) scanf("%d", &a[i]), b.push_back(a[i]);
48          sort(b.begin(), b.end());
49          b.erase(unique(b.begin(), b.end()), b.end());
50          for (int i = 1; i <= n; ++i) {
51              update(1, b.size(), root[i - 1], root[i], lower_bound(b.begin(), b.end(), a[i]) - b.
        begin() + 1);
52          }
53          int L, R, k;
54          while (m--) {
55              scanf("%d%d%d", &L, &R, &k);
56              printf("%d\n", query(1, b.size(), root[L - 1], root[R], k));
57          }
58      }
59      return 0;
60  }
```

0.1.4 莫队

```cpp
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn = 200005;

struct MO {
    int l, r, id;
}q[maxn];

int n, m, col[maxn], block, belong[maxn];
ll vis[maxn * 10], ans;
ll res[maxn];
bool cmp(const MO& a, const MO& b) { return belong[a.l] == belong[b.l] ? a.r < b.r : a.l < b.l; }
void add(ll x) {
    vis[x] ++;
    ans += x * (vis[x] * vis[x] - (vis[x] - 1) * (vis[x] - 1));
}

void del(ll x) {
    vis[x] --;
    ans -= x * ((vis[x] + 1) * (vis[x] + 1) - vis[x] * vis[x]);
}

int main() {
    scanf("%d%d", &n, &m);
    block = sqrt(n);
    for (int i = 1; i <= n; ++i) {
        scanf("%d", &col[i]);
        belong[i] = i / block + 1;
    }
    for (int i = 1; i <= m; ++i) {
        scanf("%d%d", &q[i].l, &q[i].r);
        q[i].id = i;
    }
    sort(q + 1, q + 1 + m, cmp);
    int l = 1, r = 0;
    for (int i = 1; i <= m; ++i) {
        while(r < q[i].r) add(col[++r]);
        while(r > q[i].r) del(col[r--]);
        while(l < q[i].l) del(col[l++]);
        while(l > q[i].l) add(col[--l]);
        res[q[i].id] = ans;
    }
    for (int i = 1; i <= m; ++i) printf("%lld\n", res[i]);
    return 0;
}
```

0.1.5 数位 dp 计和

```cpp
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int mod = 998244353;
pair<ll, ll> dp[20][1<<10];
bool vis[20][1<<10];
int k;
```

```
int t[20];
ll base[20];

pair<ll, ll> dfs(int pos, int state, bool limit, bool lead) {
    if (pos == -1) return __builtin_popcount(state) <= k ? make_pair(1, 0) : make_pair(0, 0);
    if (!limit && !lead && vis[pos][state]) return dp[pos][state];
    int up = limit ? t[pos] : 9;
    pair<ll, ll> res = {0, 0};
    for (int i = 0; i <= up; ++i) {
        int n_s = state;
        if (lead && i == 0) n_s = 0;
        else n_s = state | (1 << i);
        auto tmp = dfs(pos - 1, n_s, limit && i == t[pos], lead && i == 0);
        ll pre = 1ll * i * base[pos] % mod;
        (res.first += tmp.first) %= mod;
        (res.second += tmp.second + pre * tmp.first) %= mod;
    }
    if (!limit && !lead) dp[pos][state] = res, vis[pos][state] = 1;
    return res;
}

ll solve(ll x) {
    int pos = 0;
    do {
        t[pos ++] = x % 10;
    } while (x /= 10);
    return dfs(pos - 1, 0, true, true).second;
}

int main(int argc,char *argv[])
{
    base[0] = 1;
    for (int i = 1; i < 20; ++i) base[i] = base[i - 1] * 10;
    ll l, r;
    scanf("%lld%lld%d", &l, &r, &k);
    printf("%lld\n", (solve(r) - solve(l - 1) + mod) % mod);
    return 0;
}
```

0.1.6 相交回文串对数

```
#include<bits/stdc++.h>

#define ll long long
using namespace std;

const int maxn = 2e6+6;
const int N = 26;
const int mod = 51123987;

struct Palindromic_Tree {
    vector<pair<int, int> > next[maxn];
//    int next[maxn][N];//next指针，next指针和字典树类似，指向的串为当前串两端加上同一个字符构成
    int fail[maxn]{};//fail指针，失配后跳转到fail指针指向的节点
    int cnt[maxn]{}; //表示节点i表示的本质不同的串的个数（建树时求出的不是完全的，最后count()函数跑一遍以后才是正确的）
    int num[maxn]{}; //表示以节点i表示的最长回文串的最右端点为回文串结尾的回文串个数
    int len[maxn]{};//len[i]表示节点i表示的回文串的长度（一个节点表示一个回文串）
```

```
17      int S[maxn]{};//存放添加的字符
18      int last{};//指向新添加一个字母后所形成的最长回文串表示的节点。
19      int n{};//表示添加的字符个数。
20      int p{};//表示添加的节点个数。
21
22      int newnode(int l) {//新建节点
23          next[p].clear();
24  //        for (int i = 0; i < N; ++i) next[p][i] = 0;
25  //        cnt[p] = 0;
26  //        num[p] = 0;
27          len[p] = l;
28          return p++;
29      }
30
31      void init() {//初始化
32          n = last = p = 0;
33          newnode(0);
34          newnode(-1);
35          S[n] = -1;//开头放一个字符集中没有的字符，减少特判
36          fail[0] = 1;
37      }
38
39      int get_fail(int x) {//和KMP一样，失配后找一个尽量最长的
40          while (S[n - len[x] - 1] != S[n]) x = fail[x];
41          return x;
42      }
43
44      int find(int u, int c) {
45          vector<pair<int, int> > & x = next[u];
46          int sz = x.size();
47          for(int i = 0; i < sz; ++i) {
48              if(x[i].first == c) return x[i].second;
49          }
50          return 0;
51      }
52
53      int add(int c) {
54          S[++n] = c;
55          int cur = get_fail(last);//通过上一个回文串找这个回文串的匹配位置
56          int x = find(cur, c);
57          if (!x) {
58  //          if (!next[cur][c]) {//如果这个回文串没有出现过，说明出现了一个新的本质不同的回文串
59              int now = newnode(len[cur] + 2);//新建节点
60              x = now;
61              fail[now] = find(get_fail(fail[cur]), c);
62              next[cur].emplace_back(make_pair(c, now));
63  //            fail[now] = next[get_fail(fail[cur])][c];//和AC自动机一样建立fail指针，以便失配后跳转
64  //            next[cur][c] = now;
65              num[now] = num[fail[now]] + 1;
66          }
67          last = x;
68  //        last = next[cur][c];
69  //        cnt[last]++;
70          return num[last];
71      }
72
73      void count() {
74          for (int i = p - 1; i >= 0; --i) cnt[fail[i]] += cnt[i];
75          //父亲累加儿子的cnt，因为如果fail[v]=u，则u一定是v的子回文串!
```

```
76      }
77  } solve;
78
79  char s[maxn];
80
81  ll a[maxn], b[maxn];
82  int main() {
83      solve.init();
84      int n;
85      scanf("%d", &n);
86      scanf("%s", s);
87      for (int i = 0; i < n; ++i) {
88          a[i] = solve.add(s[i] - 'a');
89      }
90      solve.init();
91      for (int i = n - 1; i >= 0; --i) {
92          b[i] = (b[i + 1] + solve.add(s[i] - 'a')) % mod;
93      }
94      ll res = (b[0] * (b[0] - 1) / 2) % mod;
95      for (int i = 0; i < n; ++i) {
96          res = ((res - (a[i] * b[i + 1]) + mod) % mod) % mod;
97      }
98      printf("%lld\n", res);
99      return 0;
100 }
```

1 秃子的模板

1.1 秃子的模板

1.1.1 System of Difference Constraints

```cpp
#include <bits/stdc++.h>

using namespace std;

const int maxn=1000+10;
const int inf=0x3f3f3f3f;

struct Edge {int nex,to,w; } edge[10*maxn];

int head[maxn],cnt,dis[maxn],n;
bool vis[maxn];

void init()
{
    cnt=0;
    memset(head,0xff,sizeof head);
}

void add(int u,int v,int w)
{
    edge[cnt].nex=head[u];
    edge[cnt].to=v;
    edge[cnt].w=w;
    head[u]=++cnt;
}

void spfa(int u)
{
    int u,v,w;
    for(int i=1;i<=n;i++) dis[i]=inf,vis[i]=false;
    dis[u]=0;
    queue<int> que;
    que.push(u);
    vis[u]=true;
    while(!que.empty())
    {
        u=que.front();
        que.pop();
        vis[u]=false;
        for(int i=head[u];~i;i=edge[i].nex)
        {
            v=edge[i].v,w=edge[i].w;
            if(dis[u]+w<dis[v])
            {
                dis[v]=dis[u]+w;
                if(!vis[v])
                {
                    que.push(v);
                    vis[v]=true;
                }
            }
        }
    }
```

```
}
```

1.1.2 Aho-Corasick automation

```
const int maxn=5e5+10;

class AC_automatiion
{
public:
    int trie[maxn][26],cnt;
    int tag[maxn];
    int fail[maxn];

    void init()
    {
        memset(trie,0,sizeof trie);
        memset(tag,0,sizeof tag);
        memset(fail,0,sizeof fail);
        cnt=0;
    }

    void insert(char *str)
    {
        int root=0;
        for(int i=0;str[i];i++)
        {
            int id=str[i]-'a';
            if(!trie[root][id]) trie[root][id]=++cnt;
            root=trie[root][id];
        }
        tag[root]++;
    }

    void build()
    {
        queue<int> que;
        for(int i=0;i<26;i++) if(trie[0][i]) que.push(trie[0][i]);
        while(!que.empty())
        {
            int k=que.front(); que.pop();
            for(int i=0;i<26;i++)
            {
                if(trie[k][i]) {
                    fail[trie[k][i]]=trie[fail[k]][i];
                    que.push(trie[k][i]);
                }
                else trie[k][i]=trie[fail[k]][i];
            }
        }
    }

    int query(char *str)
    {
        int p=0,res=0;
        for(int i=0;str[i];i++)
        {
            p=trie[p][str[i]-'a'];
            for(int j=p;j&&~tag[j];j=fail[j]) res+=tag[j],tag[j]=-1;
```

```
        }
        return res;
    }
}AC;
```

1.1.3 Persistence Trie

```
#include <bits/stdc++.h>

using namespace std;

const int maxn = 6e5 + 10;

int trie[maxn*24][2], latest[maxn*24];
int s[maxn], root[maxn], n, m, tot;

void insert(int i, int k, int p, int q)
{
    if(k < 0) {
        latest[q] = i;
         return;
    }
    int c= s[i] >> k & 1;
    if(p) trie[q][c^1] = trie[p][c^1];
    trie[q][c] = ++tot;
    insert(i, k - 1, trie[p][c], trie[q][c]);
    latest[q] = max(latest[trie[q][0]], latest[trie[q][1]]);
}

int ask(int now, int val, int k, int limit)
{
    if(k < 0) return s[latest[now]] ^ val;
    int c = val >> k & 1;
    if(latest[trie[now][c^1]] >= limit) return ask(trie[now][c ^ 1], val, k - 1, limit);
    else return ask(trie[now][c], val, k - 1, limit);
}

int main()
{
    int n, m;
    scanf("%d%d", &n, &m);
    latest[0] = -1;
    root[0] = ++tot;
    insert(0, 23, 0, root[0]);
    for(int i = 1, x; i <= n; i++)
    {
        scanf("%d", &x);
        s[i] = s[i - 1] ^ x;
        root[i] = ++tot;
        insert(i, 23, root[i - 1], root[i]);
    }
    for(int i = 1, x, l, r; i <= m; i++)
    {
        char op[2];
        scanf("%s", op);
        if(op[0] == 'A')
        {
            scanf("%d", &x);
```

```
            root[++n] = ++tot;
            s[n] = s[n - 1] ^ x;
            insert(n, 23, root[n - 1], root[n]);
        }
        else
        {
            scanf("%d%d%d", &l, &r, &x);
            printf("%d\n", ask(root[r - 1], x ^ s[n], 23, l - 1));
        }
    }
    return 0;
}
```

1.1.4 Linear Basis

```
typedef long long ll;

const int MAX_BASE=63;
const int maxn=1e5+10;

int n;
ll a[maxn],b[MAX_BASE+5];

void cal()
{
    for(int i=0;i<n;i++)
    {
        for(int j=MAX_BASE;j>=0;j--)
        {
            if(a[i]>>j&1)
            {
                if(b[j]) a[i]^=b[j];
                else
                {
                    b[j]=a[i];
                    for(int k=j-1;k>=0;k--) if(b[k]&&(b[j]>>k&1)) b[j]^=b[k];
                    for(int k=j+1;k<=MAX_BASE;k++) if(b[k]>>j&1) b[k]^=b[j];
                    break;
                }
            }
        }
    }
}
```

1.1.5 MincostMaxflow

```
#include <vector>
#include <queue>

using namespace std;

const int maxn=10000+10;
const int inf=0x3f3f3f3f;

struct Edge { int from, to, cap, flow, cost; };

struct MCMF
{
```

```
13      int n,m;
14      vector<Edge> edges;
15      vector<int> G[maxn];
16      bool inq[maxn];
17      int dis[maxn], path[maxn], a[maxn];
18
19      void init(int n)
20      {
21          this->n=n;
22          for(int i=0;i<=n;i++)
23              G[i].clear();
24          edges.clear();
25      }
26
27      void addEdge(int from, int to, int cap, int cost)
28      {
29          edges.push_back(Edge{from,to,cap,0,cost});
30          edges.push_back(Edge{to,from,0,0,-cost});
31          m=edges.size();
32          G[from].push_back(m-2);
33          G[to].push_back(m-1);
34      }
35
36      bool Bellman_Ford(int s, int t, int& flow, int& cost)
37      {
38          for(int i=0; i<=n; i++) dis[i]=inf;
39          memset(inq, 0, sizeof inq);
40          dis[s]=0, inq[s]=true, path[s]=0, a[s]=inf;
41          queue<int> Q;
42          Q.push(s);
43          while(!Q.empty())
44          {
45              int u=Q.front(); Q.pop();
46              inq[u]=false;
47              for(int i=0;i<G[u].size();i++)
48              {
49                  Edge& e=edges[G[u][i]];
50                  if(e.cap>e.flow&&dis[e.to]>dis[u]+e.cost)
51                  {
52                      dis[e.to]=dis[u]+e.cost;
53                      path[e.to]=G[u][i];
54                      a[e.to]=min(a[u],e.cap-e.flow);
55                      if(!inq[e.to])
56                      {
57                          Q.push(e.to);
58                          inq[e.to]=true;
59                      }
60                  }
61              }
62          }
63          if(dis[t]==inf) return false;
64          flow+=a[t];
65          cost+=dis[t]*a[t];
66          for(int u=t;u!=s;u=edges[path[u]].from)
67          {
68              edges[path[u]].flow+=a[t];
69              edges[path[u]^1].flow-=a[t];
70          }
71          return true;
```

```
72      }
73
74      int mincostMaxFlow(int s, int t, int& cost)
75      {
76          int flow=0;
77          cost=0;
78          while(Bellman_Ford(s,t,flow,cost));
79          return flow;
80      }
81  };
```

1.1.6 Suffix Array

```
1   const int maxn=1e5+10;
2
3   char s[maxn];
4   int sa[maxn],t[maxn],t2[maxn],c[maxn],n;
5   int ra[maxn],height[maxn];
6
7   void build_sa(int m)
8   {
9       int *x=t,*y=t2;
10      for(int i=0;i<m;i++) c[i]=0;
11      for(int i=0;i<n;i++) c[x[i]=s[i]]++;
12      for(int i=1;i<m;i++) c[i]+=c[i-1];
13      for(int i=n-1;i>=0;i--) sa[--c[x[i]]]=i;
14      for(int k=1;k<=n;k<<=1)
15      {
16          int p=0;
17          for(int i=n-k;i<n;i++) y[p++]=i;
18          for(int i=0;i<n;i++) if(sa[i]>=k) y[p++]=sa[i]-k;
19          for(int i=0;i<m;i++) c[i]=0;
20          for(int i=0;i<n;i++) c[x[y[i]]]++;
21          for(int i=0;i<m;i++) c[i]+=c[i-1];
22          for(int i=n-1;i>=0;i--) sa[--c[x[y[i]]]]=y[i];
23          swap(x,y);
24          p=1;x[sa[0]]=0;
25          for(int i=1;i<n;i++)
26              x[sa[i]]=y[sa[i-1]]==y[sa[i]]&&y[sa[i-1]+k]==y[sa[i]+k]?p-1:p++;
27          if(p>=n) break;
28          m=p;
29      }
30  }
31
32  void getHeight()
33  {
34      for(int i=0;i<n;i++) ra[sa[i]]=i;
35      for(int i=0,k=0;i<n;i++)
36      {
37          if(k) k--;
38          int j=sa[ra[i]-1];
39          while(i+k<n&&j+k<n&&s[i+k]==s[j+k]) k++;
40          height[ra[i]]=k;
41      }
42  }
```

2 hld and more

2.1 hld

```
/*
sz[]数组，以x为根的子树节点个数
top[]数组，当前节点的所在链的顶端节点
son[]数组，重儿子
deep[]数组，当前节点的深度
fa[]数组，当前节点的父亲
idx[]数组，树中每个节点剖分后的新编号
rnk[]数组，idx的逆，表示线段上中当前位置表示哪个节点
*/
struct HLD {
#define type int

    struct edge {
        int a, b;
        type v;

        edge(int _a, int _b, type _v = 0) : a(_a), b(_b), v(_v) {}
    };

    struct node {
        int to;
        type w;

        node() {}

        node(int _to, type _w) : to(_to), w(_w) {}
    };

    vector<int> mp[maxn];
    vector<edge> e;

    void init(int _n) {
        n = _n;
        for (int i = 0; i <= n; i++) mp[i].clear();
        e.clear();
        e.push_back(edge(0, 0));
    }

    void add_edge(int a, int b, type v = 0) {
//        e.push_back(edge(a,b,v));
        mp[a].push_back(b);
        mp[b].push_back(a);
    }

    void dfs1(int x, int pre, int h) {
        int i, to;
        deep[x] = h;
        fa[x] = pre;
        sz[x] = 1;
        for (i = 0; i < (int) (mp[x].size()); i++) {
            to = mp[x][i];
            if (to == pre) continue;
            dfs1(to, x, h + 1);
            sz[x] += sz[to];
            if (son[x] == -1 || sz[to] > sz[son[x]]) son[x] = to;
```

```
56          }
57      }
58
59      void dfs2(int x, int tp) {
60          int i, to;
61          top[x] = tp;
62          idx[x] = ++tot;
63          rnk[idx[x]] = x;
64          if (son[x] == -1) return;
65          dfs2(son[x], tp);
66          for (i = 0; i < (int) (mp[x].size()); i++) {
67              to = mp[x][i];
68              if (to != son[x] && to != fa[x]) dfs2(to, to);
69          }
70      }
71
72      void work(int _rt = 1) {
73          memset(son, -1, sizeof son);
74          tot = 0;
75          dfs1(_rt, 0, 0);
76          dfs2(_rt, _rt);
77      }
78
79      int LCA(int x, int y) {
80          while (top[x] != top[y]) {
81              if (deep[top[x]] < deep[top[y]]) swap(x, y);
82              x = fa[top[x]];
83          }
84          if (deep[x] > deep[y]) swap(x, y);
85          return x;
86      }
87
88      void modify_node(int x, int y, type val) {
89          while (top[x] != top[y]) {
90              if (deep[top[x]] < deep[top[y]]) swap(x, y);
91              le = idx[top[x]], re = idx[x];
92              k = val;
93              update(1, 1, n);
94              x = fa[top[x]];
95          }
96          if (deep[x] > deep[y]) swap(x, y);
97          le = idx[x], re = idx[y];
98          k = val;
99          update(1, 1, n);
100     }
101
102     type query_node(int x, int y) {
103         type res = 0;
104         while (top[x] != top[y]) {
105             if (deep[top[x]] < deep[top[y]]) swap(x, y);
106             le = idx[top[x]], re = idx[x];
107             res += query(1, 1, n);
108             x = fa[top[x]];
109         }
110         if (deep[x] > deep[y]) swap(x, y);
111         le = idx[x], re = idx[y];
112         res += query(1, 1, n);
113         return res;
114     }
```

```
115
116     //path
117 //    void init_path()
118 //    {
119 //        v[idx[rt]]=0;
120 //        for(int i=1;i<n;i++)
121 //        {
122 //            if(deep[e[i].a]<deep[e[i].b]) swap(e[i].a,e[i].b);
123 //            a[idx[e[i].a]]=e[i].v;
124 //        }
125 //        build(n);
126 //    }
127     void modify_edge(int id, type val) {
128         if (deep[e[id].a] > deep[e[id].b]) {
129             le = idx[e[id].a], re = idx[e[id].a];
130             k = val;
131             update(1, 1, n);
132         } else {
133             le = idx[e[id].b], re = idx[e[id].b];
134             k = val;
135             update(1, 1, n);
136         }
137     }
138
139     void modify_path(int x, int y, type val) {
140         while (top[x] != top[y]) {
141             if (deep[top[x]] < deep[top[y]]) swap(x, y);
142             le = idx[top[x]], re = idx[x];
143             k = val;
144             update(1, 1, n);
145             x = fa[top[x]];
146         }
147         if (deep[x] > deep[y]) swap(x, y);
148         if (x != y) {
149             le = idx[x] + 1, re = idx[y];
150             k = val;
151             update(1, 1, n);
152         }
153     }
154
155     type query_path(int x, int y) {
156         type res = 0;
157         while (top[x] != top[y]) {
158             if (deep[top[x]] < deep[top[y]]) swap(x, y);
159             le = idx[top[x]], re = idx[x];
160             res += query(1, 1, n);
161             x = fa[top[x]];
162         }
163         if (deep[x] > deep[y]) swap(x, y);
164         if (x != y) {
165             le = idx[x] + 1, re = idx[y];
166             res += query(1, 1, n);
167         }
168         return res;
169     }
170
171 #undef type
172 } hld;
```

2.2 dijkstra

```
const int maxn=1e5+10;
const int inf=2147483647;

int head[maxn], dis[maxn], cnt, n;

struct Edge { int nex,to,w; }edge[20*maxn];

void add(int u,int v,int w)
{
    edge[++cnt].nex=head[u];
    edge[cnt].w=w;
    edge[cnt].to=v;
    head[u]=cnt;
}

void dijkstra(int s)
{
    priority_queue<pair<int, int> vector<pair<int, int> >, greater<pair<int, int> > > que;
    memset(dis, 0x3f, sizeof dis);
    que.push({0, s}); dis[s] = 0;
    while(!que.empty())
    {
        aoto f = que.top(); que.pop();
        int u = f.second, d = f.first;
        if(d != dis[u]) continue;
        for(int i = head[u]; ~i; i = edge[i].nex)
        {
            int v = edge[i].to, w = edge[i].w;
            if(dis[u] + w < dis[v])
            {
                dis[v] = dis[u] + w;
                que.push({dis[v], v});
            }
        }
    }
}
```