**Determining DNA Health(Hackerrank hard level problem)**

#include <bits/stdc++.h>

#define pb push\_back

#define sqr(x) (x)\*(x)

#define sz(a) int(a.size())

#define reset(a,b) memset(a,b,sizeof(a))

#define oo 1000000007

using namespace std;

typedef pair<int,int> pii;

typedef long long ll;

struct node {

    int v, h, size, priority; ll sum;

    node \*left, \*right, \*parent;

} \*nilT;

void setlink(node \*x, node \*y, bool inleft) {

    y->parent = x; if (inleft) x->left = y;

    else x->right = y;

}

node \*new\_node(int v, int h) {

    node \*p = new node; p->v = v; p->h = h; p->size = 1; p->sum = h;

    p->priority = rand(); p->left = p->right = p->parent = nilT;

    return p;

}

void update(node \*x) {

    x->sum = x->left->sum + x->right->sum + x->h;

    x->size = x->left->size + x->right->size + 1;

}

void uptree(node\* &root, node \*x) {

    node \*y = x->parent;

    if (y == nilT) return;

    node \*z = y->parent;

    if (x == y->left) { setlink(y, x->right, 1); setlink(x, y, 0); }

    else { setlink(y, x->left, 0); setlink(x, y, 1); }

    setlink(z, x, y == z->left);

    if (y == root) root = x;

    update(y); update(x);

}

void add(node\* &root, int v, int h) {

    if (root == nilT) {

        root = new\_node(v, h);

        return;

    }

    node \*x = root, \*y = nilT;

    while (x != nilT) { y = x;

        if (v < y->v) x = x->left; else    x = x->right;

    }

    x = new\_node(v, h); setlink(y, x, v < y->v);

    while (y != nilT) { y->size++; y->sum += v; y = y->parent; }

    while (x->parent != nilT && x->parent->priority < x->priority) uptree(root, x);

}

ll getSum(node \*root, int maxv){

    if(root==nilT) return 0;

    if(root->v <= maxv) return root->left->sum + root->h + getSum(root->right, maxv);

    else return getSum(root->left, maxv);

}

struct TrieNode{

    node \*data;

    TrieNode \*next[26];

} \*trieRoot;

TrieNode \*newTrieNode(){

    TrieNode \*p = new TrieNode();

    p->data = nilT;

    for(int i=0; i<26; ++i) p->next[i] = NULL;

    return p;

}

void init() {

    nilT = new node;

    nilT->sum = nilT->size = nilT->h = 0;

    nilT->priority = 0;

    trieRoot = newTrieNode();

}

const int maxn=100007;

int n,s;

string gene[maxn];

int h[maxn];

int main(){

//    freopen("input.txt","r",stdin);

    ios::sync\_with\_stdio(0);

    init();

    cin>>n;

    for(int i=1; i<=n; ++i) cin>>gene[i];

    for(int i=1; i<=n; ++i){

        cin>>h[i];

        TrieNode \*p = trieRoot;

        for(int j=0; j<sz(gene[i]); ++j){

            int v = gene[i][j]-'a';

            if(p->next[v]==NULL) p->next[v] = newTrieNode();

            p=p->next[v];

        }

        add(p->data, i-1, h[i]);

    }

    ll res1 = 1000111000111000111ll;

    ll res2 = -res1;

    cin>>s;

    for(int i=1; i<=s; ++i){

        int l,r;string d;

        cin>>l>>r>>d;

        ll val=0;

        int len=sz(d);

        for(int j=0; j<len; ++j){

            TrieNode \*p = trieRoot;

            int it=j;

            while(it<len && p->next[d[it]-'a']!=NULL){

                p = p->next[d[it]-'a'];

                if(p->data!=nilT)

                    val += getSum(p->data, r) - getSum(p->data, l-1);

                ++it;

            }

        }

        res1=min(res1,val);

        res2=max(res2,val);

    }

    cout<<res1<<' '<<res2<<endl;

}