UESTC_FinalFantasy

代码库

目录

1 Data Structure

1.1 DLX

```
1 namespace DLX{
2         const int maxn = 1005;
3         const int maxnode = 1000005;
```

```
4         struct DL{
5                 int n,sz;
6                 int S[maxn];
7
8                 int row[maxnode], col[maxnode];
9                 int L[maxnode], R[maxnode], U[maxnode], D[maxnode];
10
11                int ansd, ans[maxn];
12
13                void init(int n){ // 这里的n表示的是列数
14                        this -> n = n;
15                        for(int i = 0; i <= n; i++){
16                                L[i] = i-1; R[i] = i+1;
17                                D[i] = i; U[i] = i;
18                        }
19                        L[0] = n; R[n] = 0;
20                        sz = n + 1;
21                        memset(S,0,sizeof(S));
22                }
23
24                void addRow(int r, vector<int> columns){ // 这里的r其实并没有什么卵用，因为这个函数只能在最后加一行
25                        int first = sz;
26                        for(int i = 0; i < columns.size(); i++){
27                                int c = columns[i];
28                                L[sz] = sz - 1; R[sz] = sz + 1; U[sz] = U[c]; D[sz] = c;
29                                D[U[c]] = sz; U[c] = sz;
30                                row[sz] = r; col[sz] = c;
31                                S[c]++; sz++;
32                        }
33                        L[first] = sz-1; R[sz-1] = first;
34                }
35
36 #define FOR(i,A,s) for(int i = A[s]; i != s; i = A[i])
37                void remove(int c){ // 移除一列
38                        L[R[c]] = L[c]; R[L[c]] = R[c];
39                        FOR(i,D,c) FOR(j,R,i){
40                                U[D[j]] = U[j]; D[U[j]] = D[j]; S[col[j]]--;
41                        }
42                }
43
44                void restore(int c){ // 恢复一列
45                        FOR(i,U,c) FOR(j,L,i){
46                                S[col[j]]++; D[U[j]] = j; U[D[j]] = j;
47                        }
48                        L[R[c]] = c; R[L[c]] = c;
49                }
50                bool dfs(int d){
51                        if(R[0] == 0){
52                                ansd = d;
53                                return true;
54                        }
55
56                        int c = R[0];
57                        FOR(i,R,0) if(S[i] < S[c]) c = i;
58
59                        remove(c);
60                        FOR(i,D,c){
61                                ans[d] = row[i];
62                                FOR(j,R,i) remove(col[j]);
63                                if(dfs(d+1)) return true;
64                                FOR(j,L,i) restore(col[j]);
65                        }
66                        restore(c);
67
68                        return false;
```

```
                }
#undef FOR
                bool solve(vector<int> &v){
                        v.clear();
                        if(!dfs(0)) return false;
                        for(int i = 0; i < ansd; i++) v.push_back(ans[i]);
                        return true;
                }
        }table;
}
```

1.2 RMQ

```
/*
 * St RMQ 模板 v1.0 ——不稳定版
 *
 */
#include <iostream>
#include <cstring>
#include <cstdio>
using namespace std;
namespace ST_TABLE{
        const int maxn = 100005; // 设置总长度
        const int max_ex = 25; // 设置指数的最大值
        struct ST{
                int l,r;
                int w[max_ex + 5][maxn];
                void build(int *num,int l_,int r_){
                        memset(w[0],0x3f,sizeof(w[0]));
                        l = l_; r = r_;
                        for(int i = l; i <= r; i++){
                                w[0][i] = num[i];
                        }
                        for(int i = 1; i <= max_ex; i++){
                                for(int j = l; j <= r; j++){
                                        w[i][j] = min(w[i-1][j],w[i-1][min(max(l,r - (1<<(i-1)) + 1),j+(1<<(i-1)) )]);
                                }
                        }
                }
                int query(int l,int r){
                        int del = r - l + 1;
                        int now = 0;
                        while((1 << (now+1)) <= del) now++;
                        return min(w[now][l],w[now][max(l,r-(1<<now)+1)]);
                }
        };
}
/* 建议使用流程:
 * build()
 * query()
 */
```

1.3 Splay

```
/*
 * splay 模板 v1.0
 * 使用时记得在一头一尾加上虚点。
```

```
 */
#include <cstring>
#include <cstdio>
using namespace std;
namespace SPLAY{
        const int max_node = 200005;
        const int oo = 0x3f3f3f3f;
#define lson(x) son[(x)][0]
#define rson(x) son[(x)][1]
#define g(x) pre[pre[(x)]]
        struct Splay{
                int root,tot;
                int pre[max_node];
                int son[max_node][2];
                int size[max_node]; // the size of the subtree

                int new_node(){
                        tot++;
                        pre[tot] = 0;
                        son[tot][0] = son[tot][1] = 0;
                        return tot;
                }
                void init(){
                        tot = -1;
                        root = new_node();
                }
                void push_down(int x){
                        if(x == 0) return;
                        /*
                         *
                         */
                }
                void push_up(int x){
                        if(x == 0) return;
                        size[x] = size[lson(x)] + size[rson(x)] + 1;
                        /*
                         *
                         */
                }
                void rotate(int x){ // 0 left, 1 right
                        int y = pre[x], z = pre[y], kind = (x == lson(y));
                        push_down(y), push_down(x); // push_down
                        son[y][kind^1] = son[x][kind];
                        son[x][kind] = pre[son[y][kind^1]] = y;
                        pre[y] = x, pre[x] = z;
                        if(z) son[z][y == rson(z)] = x;
                        push_up(y); // push_up
                }
                void splay(int x,int at){
                        push_down(x); // push_down
                        while(pre[x] != at){
                                if(g(x) == at) { rotate(x); break; }
                                int y = pre[x], z = pre[y];
                                if((y == lson(z))^(x == lson(y))) { rotate(x); rotate(x);}
                                else { rotate(y); rotate(x); }
                        }
                        push_up(x); // push_up
                        if(at == 0) root = x;
                }
                int access(int x, int cur){
                        if(x == 0) return cur;
                        while(1){
                                push_down(cur); //
                                if(size[lson(cur)] + 1 == x) return cur;
                                else if(size[lson(cur)] >= x) cur = lson(cur);
```

```
                                else { x -= size[lson(cur)] + 1; cur = rson(cur); }
                        }
                }
                int get_min(int cur){
                        while(lson(cur)) cur = lson(cur);
                        return cur;
                }

                void INSERT(int x,int w){ // left subtree has x node; 用于插入节点
                        x = access(x,root); splay(x,0);
                        root = new_node(); val[root] = w;
                        lson(root) = x; rson(root) = rson(x);
                        pre[lson(root)] = pre[rson(root)] = root;
                        rson(x) = 0;
                        push_up(lson(root)); push_up(rson(root));
                        push_up(root);
                }
                void DELETE(int x){ //用于删除节点
                        int l = access(x-1,root), r = access(x+1,root);
                        splay(l,0), splay(r,l);
                        lson(r) = 0;
                        push_up(r), push_up(l);
                }
        };
#undef lson
#undef rson
#undef g
}
```

1.4 Lca

```
/*
 * 树上倍增模板，仅供参考（用于yy不出来的时候）
 */
void init_table(){
        for(int i = 1; i <= 30; i++)
                for(int j = 1; j <= pp; j++){
                        f[j][i] = f[f[j][i-1]][i-1];
                        minn[j][i] = min(minn[j][i-1],minn[f[j][i-1]][i-1]);
                }
}

int query(int u,int v){
        if(dep[u] < dep[v]) swap(u,v);
        int ans = 0x3f3f3f3f;
        for(int del = dep[u]-dep[v],i = 0; del > 0; del >>= 1,i++){
                if(del & 1){
                        ans = min(ans,minn[u][i]);
                        u = f[u][i];
                }
        }
        if(u == v) return ans;
        for(int i = 30; i >= 0; i--) {
                if(f[u][i] != f[v][i]){
                        ans = min(ans,minn[u][i]);
                        ans = min(ans,minn[v][i]);
                        u = f[u][i];
                        v = f[v][i];
                }
        }
        ans = min(ans,minn[u][0]);
```

```
        ans = min(ans,minn[v][0]);
        return ans;
}
```

1.5 树分治

```
#include <bits/stdc++.h>
#pragma comment(linker,"/STACK:102400000,102400000")
using namespace std;
#define ll long long
#define REPP(I, A, B) for (int I = (A); I <= (B); I++)
#define REP(I, A) for (int I = 0; I < (A); I++)
const ll mod = 1000003;
const int N = 1e5 + 10;
ll n, k, fid[mod+10], a[N];
vector<int> G[N];
int ansA, ansB;
ll pow(ll x, ll n) {
        ll ans = 1;
        while (n) {
                if (n & 1) {
                        ans *= x;
                        ans %= mod;
                }
                x *= x;
                x %= mod;
                n >>= 1;
        }
        return ans;
}
ll inv(ll a) { return pow(a, mod - 2) % mod; }

int T, vis[N], sz[N], vis_num[mod+10], num[mod+10];

void add(ll mul, int x) {
        if(vis_num[mul] != T) {
                vis_num[mul] = T;
                num[mul] = n+1;
        }
        num[mul] = min(num[mul], x);
}

void check_ans(ll mul, int x) {
        ll rv = fid[mul] * k % mod;
        if(vis_num[rv] != T)
                return;
        int y = num[rv];
        if(x > y) swap(x, y);
        if(x < ansA || (x == ansA && y < ansB)) {
                ansA = x;
                ansB = y;
        }
}

void get_size(int x, int fa) {
        sz[x] = 1;
        for (auto v : G[x]) {
                if(vis[v] || v == fa) continue;
                get_size(v, x);
                sz[x] += sz[v];
        }
```

```
}

int BRS, BR;
void get_root(int x, int fa, int subnum) {
        int maxson_size = 0;
        sz[x] = 1;
        for (auto v : G[x]) {
                if (vis[v] || fa == v)
                        continue;
                get_root(v, x, subnum);
                maxson_size = max(maxson_size, sz[v]);
                sz[x] += sz[v];
        }
        maxson_size = max(maxson_size, subnum - sz[x]);
        if (maxson_size < BRS) {
                BRS = maxson_size;
                BR = x;
        }
}

int find_root(int x, int val) {
        BRS = n+1;
        BR = x;
        get_root(x, x, val);
        return BR;
}

void dfs(int x, int fa, ll mul) {
        mul = mul * a[x] % mod;
        check_ans(mul, x);
        for (auto v : G[x]) {
                if(v == fa || vis[v])
                        continue;
                dfs(v, x, mul);
        }
}

void redfs(int x, int fa, ll mul) {
        mul = mul * a[x] % mod;
        add(mul, x);
        for (auto v : G[x]) {
                if(v == fa || vis[v])
                        continue;
                redfs(v, x, mul);
        }
}

void solve(int root) {
        T++;
        vis[root] = 1;
        get_size(root, root);
        int root_val = a[root];
        add(root_val, root);
        for (auto v : G[root]) {
                if(vis[v]) continue;
                dfs(v, root, 1);
                redfs(v, root, root_val);
        }
        for (auto v : G[root]) {
                if(vis[v]) continue;
                solve(find_root(v, sz[v]));
        }
}

int main() {
```

```
        REPP(i, 1, mod) fid[i] = inv(i);
        while(~scanf("%lld%lld", &n, &k)) {
                REPP(i, 1, n) scanf("%lld", &a[i]);
                REP(i, n+1) G[i].clear(), vis[i] = 0;
                int u, v;
                REP(i, n-1) {
                        scanf("%d%d", &u, &v);
                        G[u].push_back(v);
                        G[v].push_back(u);
                }
                ansA = n+1;
                solve(1);
                if(ansA == n+1) puts("No solution");
                else printf("%d %d\n", ansA, ansB);
        }
        return 0;
}
```

1.6 主席树

```
struct Node {
        int count;
        Node *left, *right;

        Node(int count, Node *left, Node *right)
                : count(count), left(left), right(right) {}

        Node *insert(int l, int r, int k);
};

Node *null;

Node *Node::insert(int l, int r, int k) {
        if (k < l || r <= k) {
                return this;
        }
        if (l + 1 == r) {
                return new Node(this->count + 1, null, null);
        }
        int m = (l + r) >> 1;
        return new Node(this->count + 1, this->left->insert(l, m, k),
                        this->right->insert(m, r, k));
}

int main() {
        // initialize
        null = new Node(0, NULL, NULL);
        null->left = null->right = null;
}
```

1.7 LCT

```
#include <bits/stdc++.h>
using namespace std;
#define FILE "read"
#define MAXN 30010
#define INF 1000000000
```

```
#define up(i,j,n) for(int i=j;i<=n;++i)
#define dn(i,j,n) for(int i=j;i>=n;--i)
namespace INIT{
	char buf[1<<15],*fs,*ft;
	inline char getc(){return (fs==ft&&(ft=(fs=buf)+fread(buf,1,1<<15,stdin),fs==ft))?0:*fs++;}
	inline int read(){
		int x=0,f=1;  char ch=getchar();
		while(!isdigit(ch))  {if(ch=='-')  f=-1;  ch=getchar();}
		while(isdigit(ch))  {x=x*10+ch-'0';  ch=getchar();}
		return x*f;
	}
}using namespace INIT;
int n,m,v[MAXN],X[MAXN],Y[MAXN];
namespace Link_Cut_Tree{
	int maxx[MAXN],sum[MAXN],f[MAXN],q[MAXN],vis[MAXN],son[MAXN][2];
	bool get(int x){return son[f[x]][1]==x;}
	bool isroot(int x){return (f[x]==0||(son[f[x]][0]!=x&&son[f[x]][1]!=x));}
	void updata(int x){
		maxx[x]=max(maxx[son[x][0]],max(maxx[son[x][1]],v[x]));
		sum[x]=sum[son[x][0]]+sum[son[x][1]]+v[x];
	}
	void pushdown(int x){
		if(vis[x]){
			swap(son[x][0],son[x][1]);
			vis[son[x][0]]^=1;  vis[son[x][1]]^=1;
			vis[x]=0;
		}
	}
	void rotate(int x){
		int y=f[x],z=f[y],which=get(x);
		if(!isroot(y))  son[z][son[z][1]==y]=x;
		son[y][which]=son[x][which^1];  f[son[y][which]]=y;
		son[x][which^1]=y;  f[y]=x;  f[x]=z;
		updata(y);  updata(x);
	}
	void splay(int x){
		int top(0);  q[++top]=x;
		for(int i=x;!isroot(i);i=f[i])  q[++top]=f[i];
		dn(i,top,1)  pushdown(q[i]);
		for(int y=f[x];!isroot(x);rotate(x),y=f[x])
			if(!isroot(y))  rotate(get(x)==get(y)?y:x);
	}
	void access(int x){for(int temp(0);x;temp=x,x=f[x])splay(x),son[x][1]=temp,updata(x);}
	void reverse(int x){access(x);splay(x);vis[x]^=1;}
	void linkk(int x,int y){reverse(x);f[x]=y;}
	void split(int x,int y){reverse(x);access(y);splay(y);}
	void makeroot(int x) {access(x);splay(x);vis[x] ^= 1;}
}
int main(){
	freopen(FILE".in","r",stdin);
	freopen(FILE".out","w",stdout);
	using namespace Link_Cut_Tree;
	n=read();  maxx[0]=-INF;
	up(i,1,n-1)  X[i]=read(),Y[i]=read();
	up(i,1,n)  v[i]=read(),sum[i]=maxx[i]=v[i];
	up(i,1,n-1)  linkk(X[i],Y[i]);
	m=read();
	up(i,1,m){
		char ch[10];  scanf("%s",ch); int x=read(),y=read();
		if(ch[1]=='H')  splay(x),v[x]=y,updata(x);
		else if(ch[1]=='M')  split(x,y),printf("%d\n",maxx[y]);
		else split(x,y),printf("%d\n",sum[y]);
	}
```

```
}
```

1.8 树链剖分

```
// my QTREE

#include <cstdio>
#include <cstring>
#include <iostream>

#define lc (u << 1)
#define rc (u << 1 | 1)

using namespace std;

int read() {
        int sign = 1, n = 0;
        char c = getchar();
        while (c < '0' || c > '9') {
                if (c == '-')
                        sign = -1;
                c = getchar();
        }
        while (c >= '0' && c <= '9') {
                n = n * 10 + c - '0';
                c = getchar();
        }
        return sign * n;
}

const int Nmax = 10005;

int T, N;
int root, tot;
int dep[Nmax], size[Nmax], son[Nmax], fa[Nmax];
int top[Nmax], w[Nmax];

struct ed {
        int v, w, next;
} e[2 * Nmax];
int k = 1, head[Nmax];
int eed[Nmax][3];

inline void adde(int u, int v, int w) {
        e[k] = (ed){v, w, head[u]};
        head[u] = k++;
}

struct BIT {
        int mmax[Nmax * 3];

        inline void init() { memset(mmax, 0, sizeof(mmax)); }

        void update(int u, int l, int r, int pos, int w) {}

        int query(int u, int l, int r, int L, int R) {}
} t;

void dfs(int u) {
        size[u] = 1, son[u] = 0;
        for (int i = head[u]; i; i = e[i].next) {
```

```
58                 int v = e[i].v;
59                 if (v == fa[u])
60                         continue;
61                 fa[v] = u;
62                 dep[v] = dep[u] + 1;
63                 dfs(v);
64                 size[u] += size[v];
65                 if (size[v] > size[son[u]])
66                         son[u] = v;
67         }
68 }
69
70 void build_tree(int u, int tp) {
71         w[u] = ++tot;
72         top[u] = tp;
73         if (son[u])
74                 build_tree(son[u], tp);
75         for (int i = head[u]; i; i = e[i].next) {
76                 int v = e[i].v;
77                 if (v == son[u] || v == fa[u])
78                         continue;
79                 build_tree(v, v);
80         }
81 }
82
83 int find(int l, int r) {
84         int f1 = top[l], f2 = top[r], res = 0;
85         while (f1 != f2) {
86                 if (dep[f1] > dep[f2]) {
87                         swap(f1, f2);
88                         swap(l, r);
89                 }
90                 res = max(res, t.query(1, 1, tot, w[f2], w[r]));
91                 r = fa[f2];
92                 f2 = top[r];
93         }
94         if (l == r)
95                 return res;
96         if (dep[l] > dep[r])
97                 swap(l, r);
98         return max(res, t.query(1, 1, tot, w[son[l]], w[r]));
99 }
100
101 int main() {
102         for (T = read(); T--;) {
103                 N = read();
104                 root = (1 + N) >> 1;
105                 fa[root] = tot = dep[root] = 0;
106                 memset(head, 0, sizeof(head));
107                 k = 1;
108                 for (int i = 1; i < N; ++i) {
109                         eed[i][0] = read(), eed[i][1] = read(), eed[i][2] = read();
110                         adde(eed[i][0], eed[i][1], eed[i][2]);
111                         adde(eed[i][1], eed[i][0], eed[i][2]);
112                 }
113                 dfs(root);
114                 build_tree(root, root);
115
116                 t.init();
117                 for (int i = 1; i < N; ++i) {
118                         if (dep[eed[i][0]] > dep[eed[i][1]])
119                                 swap(eed[i][0], eed[i][1]);
120                         t.update(1, 1, tot, w[eed[i][1]], eed[i][2]);
121                 }
122
```

```
123                 char c[10];
124                 int a, b;
125                 while (scanf("%s", c) && c[0] != 'D') {
126                         a = read();
127                         b = read();
128                         if (c[0] == 'Q')
129                                 printf("%d\n", find(a, b));
130                         else
131                                 t.update(1, 1, tot, w[eed[a][1]], b);
132                 }
133         }
134         return 0;
135 }
```

2 Graph Theory Template

2.1 SCC

```
1  /* SCC模板 v1.0 */
2  /* maxn表示的是点的数量*/
3  #include <cstring>
4  #include <cstdio>
5  #include <cstdio>
6  #include <stack>
7  using namespace std;
8
9  int pre[maxn], lowpt[maxn], time_flag, cnt_scc;
10 stack <int> st;
11 bool in_stack[maxn];
12 void scc(int v){
13         lowpt[v] = pre[v] = ++time_flag;
14         st.push(v); in_stack[v] = 1;
15         for(int k = head[v]; k; k = data[k].next){
16                 int w = data[k].tov;
17                 if(!pre[w]){
18                         scc(w);
19                         lowpt[v] = min(lowpt[w], lowpt[v]);
20                 }else if(pre[w] < pre[v]){
21                         if(in_stack[w]){
22                                 lowpt[v] = min(lowpt[v], pre[w]);
23                         }
24                 }
25         }
26         if(lowpt[v] == pre[v]){
27                 cnt_scc++;
28                 while(!st.empty() && pre[st.top()] >= pre[v]){
29                         //belong[st.top()] = cnt_scc;
30                         //num[cnt_scc]++;
31                         /* add what you what to do here . */
32                         in_stack[st.top()] = 0;
33                         st.pop();
34                 }
35         }
36 }
```

2.2 Max Flow

```cpp
/*
 * 最大流模板 v1.0
 */
#include <iostream>
#include <cstring>
#include <cstdio>
#include <queue>
#include <utility>
#include <vector>
using namespace std;
namespace flow{
        const int oo = 0x3f3f3f3f;
        const int max_node = 1005;
        struct Edge{
                int from,to,cap,flow;
                Edge(int from = 0, int to = 0, int cap = 0, int flow = 0)
                        :from(from), to(to), cap(cap), flow(flow){}
        };
        struct Dinic{
                int s,t;
                vector <int> G[max_node];
                vector <Edge> edges;
                void init(int n){
                        edges.clear();
                        for(int i = 1; i <= n; i++) G[i].clear();
                }
                void insert(int from, int to, int cap){
                        edges.push_back(Edge(from,to,cap,0));
                        edges.push_back(Edge(to,from,0,0));
                        int m = edges.size();
                        G[from].push_back(m-2);
                        G[to].push_back(m-1);
                }
                int dis[max_node];
                int bfs(){
                        memset(dis,0x3f,sizeof(dis));
                        queue <int> q; dis[s] = 0;
                        q.push(s);
                        while(!q.empty()){
                                int tt = q.front(); q.pop();
                                for(int i = 0; i < G[tt].size(); i++){
                                        Edge &e = edges[G[tt][i]];
                                        if(e.cap > e.flow){
                                                if(dis[e.to] > dis[tt] + 1){
                                                        dis[e.to] = dis[tt] + 1;
                                                        q.push(e.to);
                                                }
                                        }
                                }
                        }
                        return dis[t] != oo;
                }
                int cur[max_node];
                int dfs(int x,int a){
                        if(x == t || a == 0) return a;
                        int flow = 0, f;
                        for(int &i = cur[x]; i < G[x].size(); i++){
                                Edge &e = edges[G[x][i]];
                                if(dis[x] + 1 == dis[e.to] && (f = dfs(e.to,min(a,e.cap - e.flow))) >0 ){
                                        e.flow += f;
                                        a -= f;
                                        edges[G[x][i]^1].flow -= f;
                                        flow += f;
                                        if(a == 0) break;
                                }
```

```
                        }
                        return flow;
                }
                int max_flow(int s,int t){
                        this->s = s; this->t = t;
                        int flow = 0;
                        while(bfs()){
                                memset(cur,0,sizeof(cur));
                                flow += dfs(s,oo);
                        }
                        return flow;
                }
        }dinic;

}
```

2.3 Price flow

```
/*用法：设定s,t,调用price_flow()*/
/*费用流模板 v1.0*/
#include <iostream>
#include <cstdio>
#include <cstring>
#include <queue>
using namespace std;
int n;
namespace PRICE_FLOW{
        const int oo = 0x3f3f3f3f;
        const int  max_node = 15005;
        struct Edge{
                int from,to,flow,cap,cost;
                Edge(int from,int to,int flow,int cap,int cost)
                        :from(from),to(to),flow(flow),cap(cap),cost(cost){}
        };
        vector <Edge> edges;
        vector <int> graph[max_node];
        void insert(int from,int to,int cap,int cost){
                edges.push_back(Edge(from,to,0,cap,cost));
                edges.push_back(Edge(to,from,0,0,-cost));
                int m = edges.size();
                graph[from].push_back(m-2);
                graph[to].push_back(m-1);
        }
        int s,t;

        void init(int node_num){
                edges.clear();
                for(int i = 1; i <= node_num; i++) graph[i].clear();
        }

        int p[max_node];
        int a[max_node];
        int d[max_node];
        int inq[max_node];
        int bfs(int &flow,int &cost){
                memset(p,0,sizeof(p));
                memset(a,0x3f,sizeof(a));
                memset(d,0x3f,sizeof(d));
                memset(inq,0,sizeof(inq));
                queue <int> q;
                q.push(s);inq[s] = 1;d[s] = 0;
```

```
                while(!q.empty()){
                        int tt = q.front(); q.pop(); inq[tt] = 0;
                        int size = graph[tt].size();
                        for(int i = 0; i < size; i++) {
                                Edge &e = edges[graph[tt][i]];
                                if(e.cap > e.flow && d[e.to] > d[tt] + e.cost){
                                        d[e.to] = d[tt] + e.cost;
                                        p[e.to] = graph[tt][i];
                                        a[e.to] = min(a[tt],e.cap - e.flow);
                                        if(!inq[e.to]) {q.push(e.to);inq[e.to] = 1;}
                                }
                        }
                }
                if(d[t] == INF) return 0;
                flow += a[t];
                cost += d[t] * a[t];
                int tmp = t;
                while(tmp != s){
                        edges[p[tmp]].flow += a[t];
                        edges[p[tmp]^1].flow -= a[t];
                        tmp = edges[p[tmp]].from;
                }
                return 1;
        }
        int price_flow(){
                int flow = 0;
                int cost = 0;
                while(bfs(flow,cost)){}
                return cost;
        }
}
```

3 Math Template

3.1 FFT

```
/*
 * FFT 模板v1.0, 注意调用时, 先需要保证 n = 2^k, 并且搞好初始值。
 * 调用DFT()进行FFT, 调用DFT_h()进行逆运算（插值）。
 * A数组: [0,n)
 */
#include <iostream>
#include <cstring>
#include <cstdio>
#include <complex>
#include <cmath>
using namespace std;
namespace FFT{
        const double pi = acos(-1);
        int inv_flag = 1;
        complex<double> omega(int n){
                double u = 2 * pi/ n * inv_flag;
                return complex<double>(cos(u),sin(u));
        }
        inline void BitReverseCopy(complex<double> A[], int n, int k){
                for(int i = 0; i < n; i++){
                        int t = 0;
                        for(int j = 0; j < k; j++) if(i & (1<<j)){ // 二进制位[0,k)
                                t |= 1 << (k - j - 1);
                        }
```

```
                        if(t > i) swap(A[i],A[t]);
                }
        }
        inline void DFT(complex<double> A[], int n){ // 保证n = 2^k之后再调用。
                int k = 0; while((1<<k) < n) k++;
                BitReverseCopy(A,n,k);
                complex<double> wm,w,t,u;
                for(int s = 1; s <= k; s++){
                        int m = 1 << s;
                        wm = omega(m);
                        for(int k = 0; k < n; k += m){
                                w = 1;
                                for(int j = 0, md = m/2; j < md; j++){
                                        t = w * A[k+j+md], u = A[k+j];
                                        A[k+j] = u + t, A[k+j+md] = u - t;
                                        w *= wm;
                                }
                        }
                }
        }
        inline void DFT_h(complex<double> A[], int n){
                inv_flag = -1;
                DFT(A,n);
                inv_flag = 1;
                for(int i = 0; i < n; i++) A[i] /= n;
        }
}
```

3.2 Liner Base

```
/* Liner Base 模板v1.0 */

#include <iostream>
#include <cstring>
#include <cstdio>
using namespace std;
typedef long long LL;

namespace LINER_BASE{
        const int MAX_BASE = 63;
        struct LinerBase{
                LL b[70];
                inline void add(LL a){
                        for(int i = MAX_BASE; i >= 0; i--) if((a >> i) & 1){
                                if(b[i]) a ^= b[i];
                                else {
                                        b[i] = a;
                                        for(int j = i-1; j >= 0; j--) if(b[j] && (b[i] >> j) & 1) b[i] ^= b[j];
                                        for(int j = i+1; j <= MAX_BASE; j++) if((b[j] >> i) & 1) b[j] ^= b[i];
                                        break;
                                }
                        }
                }
                void init(){
                        memset(b, 0, sizeof(b));
                }
        };
};

/*Useage:
  通过调用add(long long)函数来在线性基中增加数字。
```

```
 */
```

3.3 simpson

```
//用法：设置F(x)，调用asr(double a,double b,dobule eps)
/*
 * simpson自适应积分模板 v1.0
 */
#include <iostream>
#include <cstring>
#include <cstring>
#include <cstdio>
#include <cmath>
using namespace std;
namespace SIMPSON{
        double F(double x){
                /*
                 *
                 */
        }
        double simpson(double a, double b){
                double c = a + (b-a)/2.0;
                return (F(a) +4*F(c) + F(b)) * (b-a) / 6.0;
        }
        double asr(double a, double b, double eps, double A){
                double c = a + (b-a) / 2.0;
                double L = simpson(a, c), R = simpson(c, b);
                if(fabs(L+R-A) <= 15*eps) return L+R+(L+R-A)/15.0;
                return asr(a, c, eps/2.0, L) + asr(c, b, eps/2.0, R);
        }
        double asr(double a, double b, double eps){
                return asr(a, b, eps, simpson(a,b));
        }
}
```

3.4 Millar-rabin

```
typedef long long LL;
bool test(LL n, LL b) {
        LL m = n - 1;
        LL counter = 0; while (~m & 1) {
                m >>= 1;
                counter ++;
        }
        LL ret = pow_mod(b, m, n);
        if (ret == 1 || ret == n - 1) {
                return true;
        }
        counter --;
        while (counter >= 0) {
                ret = multiply_mod(ret, ret, n);
                if (ret == n - 1) {
                        return true;
                }
                counter --;
        }
        return false;
```

```
}
const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
bool is_prime(LL n) {
        if (n < 2) {
                return false;
        }
        if (n < 4) {
                return true;
        }
        if (n == 3215031751LL) {
                return false;
        }
        for (int i = 0; i < 12 && BASE[i] < n; ++ i) {
                if (!test(n, BASE[i])) {
                        return false;
                }
        }
        return ture;
}
```

3.5 Polar Rho

```
typedef long long LL;
LL pollard_rho(LL n, LL seed) {
        LL x, y, head = 1, tail = 2; x = y = rand() % (n - 1) + 1;
        while (true) {
                x = multiply_mod(x, x, n);
                x = add_mod(x, seed, n);
                if (x == y) {
                        return n;
                }
                LL d = gcd(abs(x - y), n);
                if (1 < d && d < n) {
                        return d;
                }
                head ++;
                if (head == tail) {
                        y = x;
                        tail <<= 1;
                }
        }
}
vector <LL> divisors;
void factorize(LL n) {
        if (n > 1) {
                if (is_prime(n)) {
                        divisors.push_back(n);
                } else {
                        LL d = n;
                        while (d >= n) {
                                d = pollard_rho(n, rand() % (n - 1) + 1);
                        }
                        factorize(n / d);
                        factorize(d);
                }
        }
}
```

4 String Template

4.1 AC automaton (dfa)

```
/*  AC自动机模板v2.0
 *  1. root的to函数比较特别，将原本为-1的转移全部变成了0的转移，减少了大量的特判.
 *  2. 直接将ac自动机建立成一个dfn，加速50%.
 */
#include <iostream>
#include <cstring>
#include <cstdio>
#include <queue>
#include <utility>
using namespace std;
namespace AC_AUTOMATA{
        const int max_letter_size = 2; //字符集大小的上界
        const int max_ac_auto_node = 125; //ac自动机内部结点个数的上界
        struct Node_AC{
                int to[max_letter_size];
                int fail;
                int flag; // set
        };
        struct AC_AUTO{
                int root;
                int tot;
                int letter_num;
                Node_AC node[max_ac_auto_node];
                int new_node(){
                        tot++;
                        for(int i = 0; i < letter_num; i++) node[tot].to[i] = -1;
                        node[tot].fail = -1;
                        node[tot].flag = 0;
                        return tot;
                }
                void init(int letter_num_){
                        letter_num = letter_num_;
                        tot = -1;
                        root = new_node(); // == 0
                        node[root].fail = 0;
                }
                //#define trans(x) ((x) - '0') 宏加速
                inline int trans(char x){
                        return x - '0';
                }
                void add_word(char *s,int x){
                        int state = root;
                        int len = strlen(s);
                        for(int i = 0; i < len; i++){
                                int at = trans(s[i]);
                                if(node[state].to[at] == -1)
                                        node[state].to[at] = new_node();
                                state = node[state].to[at];
                        }
                        //node[state].flag |= (1 << x); //状态保存的是集合中的元素
                }
                void build_fail(){
                        for(int i = 0; i < letter_num; i++) if(node[root].to[i] == -1)
                                node[root].to[i] = 0;
                        queue <int> q; q.push(root);
                        while(!q.empty()){
                                int tt = q.front(); q.pop();
                                //node[tt].flag |= node[node[tt].fail].flag; // 合并答案
                                for(int i = 0; i < letter_num; i++){
```

```
                                        int state = node[tt].fail;
                                        if(node[tt].to[i] > 0){
                                                int nex = node[tt].to[i];
                                                q.push(nex);
                                                if(tt == 0){node[nex].fail = 0; continue;}
                                                node[nex].fail = node[state].to[i];
                                        }else {
                                                if(tt == root) continue;
                                                node[tt].to[i] = node[state].to[i];
                                        }
                                }
                        }
                }
                //#define next(state,x) node[state].to[trans(x)] 宏加速
                inline int next(int state, char x){
                        int at = trans(x);
                        return node[state].to[at];
                }
                inline pair<int,int> query(char *s){
                        int state = root;
                        int ans = 0;
                        for(int i = 0; s[i]; i++){
                                state = next(state,s[i]);
                                //ans |= node[state].flag;
                        }
                        return make_pair(state,ans);
                }
        };
        //建议调用顺序
        //ac.init(26);
        //ac.add_word(s);
        //ac.build_fail();
        //int ans = ac.query(s);
}
```

4.2 Ac automaton（失配指针）

```
/*  AC自动机模板v1.0
 *  describe: root的to函数比较特别，将原本为-1的转移全部变成了0的转移，减少了大量的特判
 */
#include <iostream>
#include <cstring>
#include <cstdio>
#include <queue>
using namespace std;
namespace AC_AUTOMATA{
        const int max_letter_size = 30; //字符集大小的上界
        const int max_ac_auto_node = 500005; //ac自动机内部结点个数的上界
        struct Node_AC{
                int to[max_letter_size];
                int fail;
                int flag; // num
        };
        struct AC_AUTO{
                int root;
                int tot;
                int letter_num;
                Node_AC node[max_ac_auto_node];
                int new_node(){
                        tot++;
                        for(int i = 0; i < letter_num; i++) node[tot].to[i] = -1;
```

```
                node[tot].fail = -1;
                node[tot].flag = 0;
                return tot;
        }
        void init(int letter_num_){
                letter_num = letter_num_;
                tot = -1;
                root = new_node(); // == 0
                node[root].fail = 0;
        }
        int trans(char x){
                return x - 'a';
        }
        void add_word(char *s){
                int state = root;
                int len = strlen(s);
                for(int i = 0; i < len; i++){
                        int at = trans(s[i]);
                        if(node[state].to[at] == -1)
                                node[state].to[at] = new_node();
                        state = node[state].to[at];
                }
                node[state].flag++;
        }
        void build_fail(){
                for(int i = 0; i < letter_num; i++) if(node[root].to[i] == -1)
                        node[root].to[i] = 0;
                queue <int> q; q.push(root);
                while(!q.empty()){
                        int tt = q.front(); q.pop();
                        for(int i = 0; i < letter_num; i++)if(node[tt].to[i] > 0){
                                int nex = node[tt].to[i]; q.push(nex);
                                if(tt == 0){node[nex].fail = 0; continue;}
                                int state = node[tt].fail;
                                while(node[state].to[i] == -1) state = node[state].fail;
                                node[nex].fail = node[state].to[i];
                                //这里可以加入对答案的合并
                        }
                }
        }
        int query(char *s){
                int ans = 0;
                int len = strlen(s);
                int state = root;
                for(int i = 0; i < len; i++){
                        int at = trans(s[i]);
                        while(node[state].to[at] == -1) state = node[state].fail;
                        state = node[state].to[at];
                        //这里可以加入对答案的计算，上述代码只是负责转移到此处。
                }
                return ans;
        }
};
//建议调用顺序
//ac.init(26);
//ac.add_word(s);
//ac.build_fail();
//int ans = ac.query(s);
}
```

4.3 Suffix Array

```cpp
/*
 * suffix_array 模板 v1.2
 */
//s : [0,length) 原串
//sa : 对原串后缀进行排序后的顺序
//rank : 原串的某个后缀在sa中的位置
//height : height[i] = sa[i]和sa[i-1]的lcp长度
//character : [0,range)
//height : [1,n)
#include <iostream>
#include <cstring>
#include <cstdio>
using namespace std;
namespace SUFFIX_ARRAY{
	const int maxn = 20005; // 用于设置字符串的长度
	struct SuffixArray{
		int n,range;
		int sa[maxn];
		int w[maxn];
		int tmp[maxn*2];
		int rank[maxn*2];
		int height[maxn];
		void init(){ //如果有多组数据，则一定记得调用
			n = range = 0;
			memset(sa,0,sizeof(sa));
			memset(w,0,sizeof(w));
			memset(tmp,-1,sizeof(tmp)); //设为-1简化比较
			memset(rank,-1,sizeof(rank)); //设为-1简化比较
			memset(height,0,sizeof(height));
		}
		int cmp(int i,int j, int *r, int step){ //内部函数,注意数组tmp和rank要开到2n，不然这里会越界访问
			return r[i] == r[j] && r[i+step] == r[j+step];
		}
		void build(int *s,int len_,int range_){ // s表示原串，len_表示原串长度，range_表示原串中字符的范围，建议：range很大时，先离散化，模板不具备离散化的功能
			n = len_; range = max(n,range_);
			int i;
			for(i = 0; i < n; i++) w[rank[i] = s[i]]++;
			for(i = 1; i < range; i++) w[i] += w[i-1];
			for(i = n-1; i >= 0; i--) sa[--w[rank[i]]] = i;
			for(int step = 1; step < n; step <<= 1){
				int p = 0;
				for(i = n - step; i < n; i++) tmp[p++] = i;
				for(i = 0; i < n; i++) if(sa[i] >= step) tmp[p++] = sa[i] - step;
				for(int i = 0; i < range; i++) w[i] = 0;
				for(i = 0; i < n; i++) w[rank[tmp[i]]]++;
				for(i = 1; i < range; i++) w[i] += w[i-1];
				for(i = n-1; i >= 0; i--) sa[--w[rank[tmp[i]]]] = tmp[i];
				for(swap(tmp,rank), p = 0, i = 0; i < n; i++)
					rank[sa[i]] = (i == 0 || cmp(sa[i-1],sa[i],tmp,step)) ? p : ++p;
			}
		}
		void get_real_rank(){
			for(int i = 0; i < n; i++) rank[sa[i]] = i;
		}
		void get_height(int *s){ //[1,n)
			s[n] = -1; // 不然循环可能不会终止
			for(int i = 0,h = 0; i < n; height[rank[i++]] = h){
				if(rank[i]-1 < 0) { h = 0; continue;}
				for(h?--h:0; s[sa[rank[i]-1]+h] == s[i+h]; h++);
			}
		}
	}suffix_array;
}

```

```
/*建议调用顺序
  suffix_array.init();
  suffix_array.build(...);
  suffix_array.get_real_rank();
  suffix_array.get_height(...);
 */
```

4.4 Suffix automaton

```
/*
 *  后缀自动机模板 v1.0
 *  INFO:
 *  1. root = 0;
 *  2. slink[root] = -1;
 *  3. L,R 表示当前状态中的串的长度区间
 */
#include <iostream>
#include <cstring>
#include <cstdio>
#include <string>
using namespace std;
namespace SUFFIX_AUTO{
        const int maxn = 200005;
        const int letter_num = 30;

        struct Suffix_Auto{
                int L[maxn * 2], R[maxn * 2];
                int slink[maxn * 2];
                int trans[maxn * 2][letter_num];

                int tot_sam;
                int last;
                int root;
                void init(){
                        tot_sam = -1;
                        root = last = new_node(0,0,NULL,-1);
                }
                int new_node(int l,int r,int *tran, int s){
                        tot_sam++;
                        L[tot_sam] = l; R[tot_sam] = r;
                        slink[tot_sam] = s;
                        if(tran == NULL) memset(trans[tot_sam],-1,sizeof(trans[tot_sam]));
                        else memcpy(trans[tot_sam], tran, sizeof(trans[tot_sam]));

                        return tot_sam;
                }

                inline void update(int x){
                        L[x] = R[slink[x]] + 1;
                }

                void add(char s){
                        int now = s - 'a';
                        int y = new_node(-1,R[last] + 1, NULL, -1);
                        int p = last; last = y;

                        while(p != -1 && trans[p][now] == -1){
                                trans[p][now] = y;
                                p = slink[p];
                        }

```

```
                        if(p == -1){
                                slink[y] = 0;
                                update(y);
                                return;
                        }

                        int q = trans[p][now];
                        if(R[q] == R[p] + 1){
                                slink[y] = q;
                                update(y);
                                return;
                        }

                        int nq = new_node(L[q],R[p]+1,trans[q],slink[q]);
                        slink[q] = slink[y] = nq;
                        update(q); update(y);
                        while(p != - 1 && trans[p][now] == q){
                                trans[p][now] = nq;
                                p = slink[p];
                        }
                        return;
                }
                inline int next(int state, char ch){
                        return trans[state][ch - 'a'];
                }
        };
}
//SUFFIX_AUTO::Suffix_Auto sa;

/* Useage
   1. sa.init();
   2. sa.add(char s);
   3. sa.next(stats, ch);
 */
```

4.5 马拉车

```
void manacher(char *text, int n) {
        palindrome[0] = 1;
        for (int i = 1, j = 0; i < n; ++ i) {
                if (j + palindrome[j] <= i) {
                        palindrome[i] = 0;
                } else {
                        palindrome[i] = min(palindrome[(j << 1) - i], j + palindrome[j] - i);
                }
                while (i - palindrome[i] >= 0 && i + palindrome[i] < n
                                && text[i - palindrome[i]] == text[i + palindrome[i]]) {
                        palindrome[i] ++;
                }
                if (i + palindrome[i] > j + palindrome[j]) {
                        j = i;
                }
        }
}
```