

省选模板

panda_2134

2018 年 4 月 1 日

目录

1	图论	4
1.1	强连通分量	4
1.2	点双连通分量	4
1.3	边双连通分量	4
2	网络流	5
2.1	最大流	5
2.2	最小费用最大流	6
3	树	7
3.1	倍增 LCA	7
3.2	欧拉序列求 LCA	8
3.3	树链剖分	9
3.4	点分治	9
4	单调数据结构	10
4.1	单调队列	10
4.2	单调栈	10
5	线段树	11
5.1	Lazy-Tag	11
5.2	动态开点线段树	11
5.3	可持久化线段树	11
5.4	二维线段树	11
6	平衡树	12
6.1	Treap	12
6.2	Splay	12
7	动态树	13
7.1	Link-cut Tree	13
8	字符串	14
8.1	KMP 字符串匹配	14
8.2	AC 自动机	14
8.3	后缀数组	16
9	Miscellaneous	18
9.1	ST 表	18
9.2	Fenwick Tree	18
9.3	左偏树	18

10 悬线法	19
10.1 Algorithm 1	19
10.2 Algorithm 2	19
11 莫队	20
11.1 普通莫队	20
11.2 带修改莫队	20
12 分块相关	21
12.1 分块	21
12.2 区间众数	21

1 图论

1.1 强连通分量

1.2 点双连通分量

1.3 边双连通分量

2 网络流

2.1 最大流

MaximumFlow.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Edge {
5     int v, flow, cap, next;
6 };
7
8 const int MAXN = 1e4, MAXM = 1e5, INF = 0x3f3f3f3f;
9 int n, m, s, t, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1];
10 int d[MAXN+10], cur[MAXN+10];
11
12 void AddEdge(int u, int v, int cap) {
13     E[++e_ptr] = (Edge) { v, 0, cap, head[u] }; head[u] = e_ptr;
14     E[++e_ptr] = (Edge) { u, 0, 0, head[v] }; head[v] = e_ptr;
15 }
16
17 bool BFS() {
18     queue<int> Q;
19     memset(d, 0xff, sizeof(d));
20     Q.push(s); d[s] = 0;
21     while(!Q.empty()) {
22         int u = Q.front(); Q.pop();
23         for(int j=head[u]; j; j=E[j].next) {
24             int v = E[j].v, f = E[j].flow, c = E[j].cap;
25             if(f < c && d[v] == -1) {
26                 d[v] = d[u] + 1;
27                 if(v == t) return true;
28                 else Q.push(v);
29             }
30         }
31     }
32     return false;
33 }
34
35 int DFS(int u, int flow) {
36     if(u == t || flow == 0) return flow // !!!!!
37     int res = flow
38     for(int &j=cur[u]; j; j=E[j].next) { // !!!!!
39         int v = E[j].v, f = E[j].flow, c = E[j].cap;
40         if(f < c && d[v] == d[u] + 1) {
41             int aug = DFS(v, min(res, c-f));
42             E[j].flow += aug; E[j^1].flow -= aug;
43             res -= aug;
44             if(res == 0) break; // !!!!!
45         }
46     }
47     return flow - res;
48 }
49
50 int Dinic() {
51     int MaxFlow = 0, CurFlow = 0;
```

```

52     while(BFS()) {
53         memcpy(cur, head, sizeof(head));
54         while((CurFlow = DFS(s, INF)))
55             MaxFlow += CurFlow
56     }
57     return MaxFlow
58 }
59
60 int main() {
61     int u, v, c;
62     scanf("%d%d%d", &n, &m, &s, &t);
63     for(int i = 1; i <= m; i++) {
64         scanf("%d%d%d", &u, &v, &c);
65         AddEdge(u, v, c);
66     }
67     printf("%d", Dinic());
68     return 0;
69 }

```

2.2 最小费用最大流

3 树

3.1 倍增 LCA

DoublingLCA.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Edge { int v, next; };
5
6 const int MAXN = 1e6, LOG = 20;
7 int n, q, s, e_ptr = 1, head[MAXN+10]; Edge E[(MAXN+10)<<1];
8 int dep[MAXN+10], anc[MAXN+10][LOG+1];
9
10 void add_edge(int u, int v) { E[++e_ptr] = (Edge) { v, head[u] }; head[u] = e_ptr; }
11 void add_pair(int u, int v) { add_edge(u, v); add_edge(v, u); }
12
13 void dfs(int u) {
14     for(int i = 1; i <= LOG; i++)
15         anc[u][i] = anc[anc[u][i-1]][i-1];
16     for(int j=head[u]; j; j=E[j].next) {
17         int v = E[j].v;
18         if(v == anc[u][0]) continue;
19         anc[v][0] = u; dep[v] = dep[u] + 1;
20         dfs(v);
21     }
22 }
23
24 int lca(int u, int v) {
25     if(dep[u] < dep[v]) swap(u, v);
26     for(int i = LOG; i >= 0; i--)
27         if(dep[anc[u][i]] >= dep[v])
28             u = anc[u][i];
29     if(u == v) return u;
30     for(int i = LOG; i >= 0; i--)
31         if(anc[u][i] != anc[v][i])
32             u = anc[u][i], v = anc[v][i];
33     u = anc[u][0], v = anc[v][0];
34     return u;
35 }
36
37 inline int readint() {
38     int f=1, r=0; char c=getchar();
39     while(!isdigit(c)) { if(c=='-') f=-1; c=getchar(); }
40     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
41     return f*r;
42 }
43
44 int main() {
45     int u, v;
46     n = readint(); q = readint(); s = readint();
47     for(int i = 1; i <= n-1; i++) {
48         u = readint(); v = readint();
49         add_pair(u, v);
50     }
51     dep[s] = 1; dfs(s);
```

```

52     while(q--) {
53         u = readint(); v = readint();
54         printf("%d\n", lca(u, v));
55     }
56     return 0;
57 }

```

3.2 欧拉序列求 LCA

EulerTourLCA.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int MAXN = 1e6;
5
6  struct Edge {
7      int v, next;
8  };
9
10 int n, q, s, e_ptr = 1, dfs_clock, head[MAXN+10]; Edge E[(MAXN+10)<<1];
11 int dfn[MAXN+10], dfs_seq[MAXN+10], idx[MAXN+10], euler_seq[(MAXN+10)<<1], st[(MAXN+10)<<1][22];
12 /*
13     dfn: dfs-clock of vertex u
14     idx: the index of vertex u in euler-tour sequence
15     dfs_seq: the dfs sequence
16 */
17
18 void add_edge(int u, int v) {
19     E[++e_ptr] = (Edge) { v, head[u] }; head[u] = e_ptr;
20 }
21
22 void add_pair(int u, int v) {
23     add_edge(u, v); add_edge(v, u);
24 }
25
26 inline int readint() {
27     int f=1, r=0; char c=getchar();
28     while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
29     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
30     return f*r;
31 }
32
33 void dfs(int u, int fa) {
34     euler_seq[++euler_seq[0]] = dfn[u] = ++dfs_clock;
35     idx[u] = euler_seq[0]; dfs_seq[dfs_clock] = u;
36     for(int j=head[u]; j; j=E[j].next) {
37         int v = E[j].v;
38         if(v == fa) continue;
39         dfs(v, u);
40         euler_seq[++euler_seq[0]] = dfn[u];
41     }
42 }
43
44 void init_lca() {
45     memset(st, 0x3f, sizeof(st));

```



```

46     for(int i = 1; i <= euler_seq[0]; i++)
47         st[i][0] = euler_seq[i];
48     for(int j = 1; j <= 21; j++)
49         for(int i = 1; i <= euler_seq[0] - (1<<j) + 1; i++) // bounds of sparse-table!
50             st[i][j] = min(st[i][j-1], st[i + (1<<(j-1))][j-1]);
51 }
52
53 int query(int l, int r) {
54     if(l > r) swap(l, r);
55     int j;
56     for(j = 0; (1<<(j+1)) <= (r-l+1); j++);
57     return min(st[l][j], st[r - (1<<j) + 1][j]);
58 }
59
60 int lca(int u, int v) {
61     return dfs_seq[query(dx[u], dx[v])];
62 }
63
64 int main() {
65     int u, v;
66     n = readint(); q = readint(); s = readint();
67     for(int i = 1; i <= n-1; i++) {
68         u = readint(); v = readint();
69         addpair(u, v);
70     }
71     dfs(s, -1); init_lca();
72     while(q--) {
73         u = readint(); v = readint();
74         printf("%d\n", lca(u, v));
75     }
76     return 0;
77 }

```

3.3 树链剖分

3.4 点分治

4 单调数据结构

4.1 单调队列

4.2 单调栈

5 线段树

5.1 Lazy-Tag

5.2 动态开点线段树

5.3 可持久化线段树

5.4 二维线段树

6 平衡树

6.1 Treap

6.2 Splay

7 动态树

7.1 Link-cut Tree

8 字符串

8.1 KMP 字符串匹配

1-indexed

8.2 AC 自动机

0-indexed

ACAutomaton.cpp

```
1 #include <bits/stdc++.h>
2 #define CLEAR(x) memset((x), 0, sizeof(x))
3 using namespace std;
4
5 const int SIGMA = 26, MAX_TEMP_LEN = 70, MAXN = 150,
6 MAX_LEN = 1e6, MAX_NODE = MAXN * MAX_TEMP_LEN;
7
8 int N, sz, ch[MAX_NODE + 10][SIGMA + 2], f[MAX_NODE + 10], last[MAX_NODE+10],
9     val[MAX_NODE + 10], found_cnt[MAX_NODE+10];
10 char str[MAX_LEN+10], tpl[MAXN+10][MAX_TEMP_LEN+10];
11 unordered_map<string, int> ms;
12
13 inline int idx(char c) { return c - 'a' + 1; }
14
15 void insert(char *str) {
16     int u = 0, len = strlen(str);
17     for(int i = 0; i < len; i++) {
18         int c = idx(str[i]);
19         if(!ch[u][c]) ch[u][c] = ++sz;
20         u = ch[u][c];
21     }
22     ms[string(str)] = u;
23     ++val[u];
24 }
25
26 void get_fail() {
27     queue<int> Q;
28     f[0] = 0;
29     for(int c = 1; c <= SIGMA; c++) if(ch[0][c]) {
30         int v = ch[0][c];
31         f[v] = last[v] = 0;
32         Q.push(v);
33     }
34     while(!Q.empty()) {
35         int u = Q.front(); Q.pop();
36         for(int c = 1; c <= SIGMA; c++) {
37             int v = ch[u][c];
38             if(!v) {
39                 ch[u][c] = ch[f[u]][c];
40                 continue;
41             }
42             Q.push(v);
43         }
44     }
```

```

45         int u2 = f[u];
46         while(u2 && !ch[u2][c]) u2 = f[u2];
47         f[v] = ch[u2][c];
48         last[v] = val[f[v]] ? f[v] : last[f[v]];
49     }
50 }
51 }
52
53 void found(int u) {
54     for(; u; u = last[u])
55         found_cnt[u] += val[u];
56 }
57
58 void search(char *str) {
59     int u = 0, len = strlen(str);
60     for(int i = 0; i < len; i++) {
61         int c = idx(str[i]);
62         u = ch[u][c];
63         if(val[u]) found(u);
64         else if(last[u]) found(last[u]);
65     }
66 }
67
68 inline void readstr(char *str) {
69     char c=getchar(); int p=0;
70     while(!isalnum(c) && !ispunct(c)) c = getchar();
71     while(isalnum(c) || ispunct(c)) {
72         str[p++] = c;
73         c = getchar();
74     }
75     str[p++] = '\0';
76 }
77
78 int main() {
79     while(true) {
80         int ans = 0;
81         sz = 0; CLEAR(ch); CLEAR(f); CLEAR(found_cnt);
82         CLEAR(last); CLEAR(tpl); CLEAR(val); CLEAR(str);
83
84         scanf("%d", &N); if(N == 0) break;
85         for(int i = 1; i <= N; i++) {
86             readstr(tpl[i]); insert(tpl[i]);
87         }
88         get_fail();
89
90         readstr(str); search(str);
91
92         for(int i = 0; i <= sz; i++)
93             ans = max(ans, found_cnt[i]);
94         printf("%d\n", ans);
95         for(int i = 1; i <= N; i++)
96             if(found_cnt[ms[string(tpl[i])]] == ans)
97                 printf("%s\n", tpl[i]);
98     }
99     return 0;
100 }

```

8.3 后缀数组

0-indexed

SuffixArray.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXLEN = 1e6, SIGMA = 100;
5
6 inline int idx(char c) {
7     if(!c) return 0;
8     else if(isdigit(c)) return c - '0' + 1;
9     else if(isupper(c)) return c - 'A' + 1 + 10;
10    else if(islower(c)) return c - 'a' + 1 + 10 + 26;
11    else throw "Invalid Character";
12 }
13
14 struct SuffixArray {
15     int sa[MAXLEN+10], rk[MAXLEN+10], buf[3][MAXLEN+10], height[MAXLEN+10], c[MAXLEN+10];
16     void build_sa(char *s, int len) {
17         int m = SIGMA + 10, n = len + 1, *x = buf[0], *y = buf[1];
18         for(int i = 0; i < m; i++) c[i] = 0;
19         for(int i = 0; i < n; i++) ++c[x[i] = idx(s[i])];
20         for(int i = 1; i < m; i++) c[i] += c[i-1];
21         for(int i = n-1; i >= 0; i--) sa[--c[x[i]]] = i;
22         for(int k = 1; k <= n; k <= 1) {
23             int p = 0;
24             for(int i = n-k; i < n; i++) y[p++] = i;
25             for(int i = 0; i < n; i++) if(sa[i] >= k) y[p++] = sa[i] - k;
26             for(int i = 0; i < m; i++) c[i] = 0;
27             for(int i = 0; i < n; i++) ++c[x[y[i]]];
28             for(int i = 1; i < m; i++) c[i] += c[i-1];
29             for(int i = n-1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
30             swap(x, y);
31             p = 1, x[sa[0]] = 0;
32             for(int i = 1; i < n; i++)
33                 x[sa[i]] = (y[sa[i]] == y[sa[i-1]] && y[sa[i] + k] == y[sa[i-1] + k] ? p-1 : p++);
34             if(p >= n) break;
35             m = p;
36         }
37         memcpy(rk, x, sizeof(rk));
38         int k = 0;
39         for(int i = 0; i < n; i++) {
40             if(!rk[i]) continue;
41             if(k) k--;
42             int j = sa[rk[i]-1];
43             while(s[i+k] == s[j+k]) k++;
44             height[rk[i]] = k;
45         }
46     }
47 } SA;
48 inline void readstr(char* str) {
49     char c=getchar(); int p=0;
50     while(!isalnum(c) && !ispunct(c)) c=getchar();
51     while(isalnum(c) || ispunct(c)) {
52         str[p++] = c;
```



```
53     c = getchar();
54 }
55 str[p++] = '\0';
56 }
57
58 int len;
59 char str[MAXLEN+10];
60
61 int main() {
62     readstr(str); len = strlen(str);
63     SA.build_sa(str, len);
64     for(int i = 1; i <= len; i++)
65         printf("%d ", SA.sa[i]+1);
66     return 0;
67 }
```

9 Miscellaneous

9.1 ST 表

9.2 Fenwick Tree

9.3 左偏树

10 悬线法

10.1 Algorithm 1

10.2 Algorithm 2

11 莫队

11.1 普通莫队

11.2 带修改莫队

12 分块相关

12.1 分块

12.2 区间众数