OI 模板

panda_2134

2018 年 4 月 2 日

目录

1 图论

1.1 强连通分量

一有向图上每个点有非负权值，求一条路径，使得路径上点权值和最大。点和边都可以多次经过，但是权值只计入答案一次。

Solution: 缩点后直接在 DAG 上 DP.

GraphTheory/TarjanSCC.cpp

```cpp
#include <bits/stdc++.h>
#define fst first
#define snd second
using namespace std;

typedef pair<int, int> pii;
const int MAXN = 1e5, INF = 0x3f3f3f3f;

struct Graph {
    struct Edge {
        int v, next;
    };

    int n, m, e_ptr = 1, head[MAXN+10]; Edge E[(MAXN+10)<<1];

    void add_edge(int u, int v) {
        E[++e_ptr] = (Edge) { v, head[u] }; head[u] = e_ptr;
    }
} G1, G2;

int dfs_clock, scc_cnt, sccno[MAXN+10], dfn[MAXN+10], low[MAXN+10];
int ans, topo_cnt, topo_seq[MAXN+10], w[MAXN+10],
    tot[MAXN+10], vis[MAXN+10], dp[MAXN+10];

stack<int> S;
void dfs(int u) {
    dfn[u] = low[u] = ++dfs_clock;
    S.push(u);
    for(int j=G1.head[u]; j; j=G1.E[j].next) {
        int v = G1.E[j].v;
        if(!dfn[v]) {
            dfs(v);
            low[u] = min(low[u], low[v]);
        } else if(!sccno[v])
            low[u] = min(low[u], dfn[v]);
    }
    if(low[u] == dfn[u]) {
        int v; ++scc_cnt;
        do {
            v = S.top(); S.pop();
            sccno[v] = scc_cnt;
            tot[scc_cnt] += w[v];
        } while(u != v);
    }
}

void Tarjan() {
```

```
48      for(int u = 1; u <= G1.n; u++)
49          if(!dfn[u]) dfs(u);
50  }
51
52  void scc_graph() {
53      set<pii> evis;
54      for(int u = 1; u <= G1.n; u++)
55          for(int j=G1.head[u]; j; j=G1.E[j].next) {
56              int v = G1.E[j].v;
57              if(sccno[u] == sccno[v] || evis.count(make_pair(sccno[u], sccno[v])))
58                  continue;
59              else {
60                  evis.insert(make_pair(sccno[u], sccno[v]));
61                  G2.add_edge(sccno[u], sccno[v]);
62              }
63          }
64      G2.n = scc_cnt;
65  }
66
67  bool topo_dfs(int u) {
68      vis[u] = -1;
69      for(int j=G2.head[u]; j; j=G2.E[j].next) {
70          int v = G2.E[j].v;
71          if(vis[v] == -1 || (vis[v] == 0 && !topo_dfs(v)))
72              return false;
73      }
74      vis[u] = 1;
75      topo_seq[topo_cnt--] = u;
76      return true;
77  }
78
79  bool toposort() {
80      topo_cnt = G2.n;
81      for(int u = G2.n; u >= 1; u--)
82          if(vis[u] == 0 && !topo_dfs(u)) return false;
83      return true;
84  }
85
86  inline int readint() {
87      int f=1, r=0; char c=getchar();
88      while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
89      while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
90      return f*r;
91  }
92
93  void init() {
94      int u, v;
95      G1.n = readint(); G1.m = readint();
96      for(int i = 1; i <= G1.n; i++)
97          w[i] = readint();
98      for(int i = 1; i <= G1.m; i++) {
99          u = readint(); v = readint();
100         G1.add_edge(u, v);
101     }
102     Tarjan(); scc_graph();
103     assert(toposort());
104 }
```

```
105
106 void work() {
107     for(int i = G2.n; i >= 1; i--) {
108         int u = topo_seq[i], maxv = 0;
109         for(int j=G2.head[u]; j; j=G2.E[j].next) {
110             int v = G2.E[j].v;
111             if(dp[v] > maxv) maxv = dp[v];
112         }
113         dp[u] = tot[u] + maxv;
114         ans = max(ans, dp[u]);
115     }
116     printf("%d", ans);
117 }
118
119 int main() {
120     init(); work();
121     return 0;
122 }
```

1.2 桥和割点

1.3 点双连通分量

1.4 边双连通分量

2 最短路

2.1 Dijkstra

2.1.1 std :: priority_queue

2.1.2 __gnu_pbds :: priority_queue

3 网络流

3.1 最大流

NetworkFlow/MaximumFlow.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

struct Edge {
    int v, flow, cap, next;
};

const int MAXN = 1e4, MAXM = 1e5, INF = 0x3f3f3f3f;
int n, m, s, t, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1];
int d[MAXN+10], cur[MAXN+10];

void AddEdge(int u, int v, int cap) {
    E[++e_ptr] = (Edge) { v, 0, cap, head[u] }; head[u] = e_ptr;
    E[++e_ptr] = (Edge) { u, 0,   0, head[v] }; head[v] = e_ptr;
}

bool BFS() {
    queue<int> Q;
    memset(d, 0xff, sizeof(d));
    Q.push(s); d[s] = 0;
    while(!Q.empty()) {
        int u = Q.front(); Q.pop();
        for(int j=head[u]; j; j=E[j].next) {
            int v = E[j].v, f = E[j].flow, c = E[j].cap;
            if(f < c && d[v] == -1) {
                d[v] = d[u] + 1;
                if(v == t) return true;
                else Q.push(v);
            }
        }
    }
    return false;
}

int DFS(int u, int flow) {
    if(u == t || flow == 0) return flow; // !!!!!
    int res = flow;
    for(int &j=cur[u]; j; j=E[j].next) { // !!!!!
        int v = E[j].v, f = E[j].flow, c = E[j].cap;
        if(f < c && d[v] == d[u] + 1) {
            int aug = DFS(v, min(res, c-f));
            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%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  }
```

3.2 最小费用最大流

3.2.1 zkw 费用流

NetworkFlow/zkw.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  typedef long long int64;
5  struct Edge {
6      int u, v;
7      int64 flow, cap, cost;
8      int next;
9  };
10
11 const int MAXN = 5e3, MAXM = 5e4;
12 const int64 LL_INF = 0x3f3f3f3f3f3f3f3fLL;
13 int n, m, s, t, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1]; // ** E[(MAXM+10)<<1] **
14 int64 MaxFlow, MinCost, dist[MAXN+10], inq[MAXN+10], vis[MAXN+10];
15
16 void add_edge(int u, int v, int64 cap, int64 cost) {
17     E[++e_ptr] = (Edge) { u, v, 0, cap, cost, head[u] }; head[u] = e_ptr;
18     E[++e_ptr] = (Edge) { v, u, 0,  0, -cost, head[v] }; head[v] = e_ptr;
19 }
20
21 bool spfa() {
22     queue<int> Q;
```

```
23      memset(dist, 0x3f, sizeof(dist));
24      Q.push(t); dist[t] = 0; inq[t] = true;
25      while(!Q.empty()) {
26          int u = Q.front(); Q.pop(); inq[u] = false;
27          for(int j=head[u]; j; j=E[j].next) {
28              int v = E[j].v; int64 f = E[j^1].flow, c = E[j^1].cap, len = E[j^1].cost;
29              if(f < c && dist[v] > dist[u] + len) {
30                  dist[v] = dist[u] + len;
31                  if(!inq[v]) {
32                      inq[v] = true; Q.push(v);
33                  }
34              }
35          }
36      }
37      return dist[s] != LL_INF;
38  }
39
40  int64 dfs(int u, int64 flow) {
41      if(u == t || flow == 0) return flow;
42      vis[u] = true;
43      int64 res = flow;
44      for(int j=head[u]; j; j=E[j].next) {
45          int v = E[j].v; int64 f = E[j].flow, c = E[j].cap, len = E[j].cost;
46          if(f < c && !vis[v] && dist[v] == dist[u] - len) {
47              int64 aug = dfs(v, min(res, c-f));
48              E[j].flow += aug; E[j^1].flow -= aug;
49              res -= aug;
50              if(res == 0LL) break;
51          }
52      }
53      return flow - res;
54  }
55
56  void zkw() {
57      int64 CurFlow = 0LL;
58      while(spfa()) {
59          while(memset(vis, 0, sizeof(vis)),
60              CurFlow = dfs(s, LL_INF)) {
61              MaxFlow += CurFlow;
62              MinCost += dist[s] * CurFlow;
63          }
64      }
65  }
66
67  template<typename T>
68  inline void readint(T &x) {
69      T f=1, r=0; char c=getchar();
70      while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
71      while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
72      x = f*r;
73  }
74
75  int main() {
76      int u, v; int64 w, c;
77      readint(n); readint(m); readint(s); readint(t);
78      for(int i = 1; i <= m; i++) {
79          readint(u); readint(v); readint(w); readint(c);
```

```
80        add_edge(u, v, w, c);
81    }
82    zkw();
83    printf("%lld %lld", MaxFlow, MinCost);
84    return 0;
85 }
```

3.2.2 Primal Dual

NetworkFlow/PrimalDual.cpp

```
1  #include <bits/stdc++.h>
2  #include <bits/extc++.h>
3  #define fst first
4  #define snd second
5
6  using namespace std;
7
8  typedef long long int64;
9  typedef pair<int64, int> HeapNode;
10 typedef __gnu_pbds::priority_queue<HeapNode, greater<HeapNode>,
11         __gnu_pbds::pairing_heap_tag> PairingHeap;
12 const int MAXN = 5e3, MAXM = 5e4;
13 const int64 LL_INF = 0x3f3f3f3f3f3f3f3fLL;
14
15 struct Edge {
16     int u, v;
17     int64 flow, cap, cost;
18     int next;
19 };
20
21 int n, m, s, t, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1];
22 int64 MaxFlow, MinCost, delta, dist[MAXN+10], vis[MAXN+10], inq[MAXN+10];
23 PairingHeap::point_iterator it[MAXN+10];
24
25 void add_edge(int u, int v, int64 cap, int64 cost) {
26     E[++e_ptr] = (Edge) { u, v, 0, cap, cost, head[u] }; head[u] = e_ptr;
27     E[++e_ptr] = (Edge) { v, u, 0,  0, -cost, head[v] }; head[v] = e_ptr;
28 }
29
30 void Reduce() {
31     for(int i = 2; i <= e_ptr; i++)
32         E[i].cost -= (dist[E[i].u] - dist[E[i].v]);
33     delta += dist[s];
34 }
35
36 bool BellmanFord() {
37     queue<int> Q;
38     memset(dist, 0x3f, sizeof(dist));
39     Q.push(t); dist[t] = 0; inq[t] = true;
40     while(!Q.empty()) {
41         int u = Q.front(); Q.pop(); inq[u] = false;
42         for(int j=head[u]; j; j=E[j].next) {
43             int v = E[j].v; int64 f = E[j^1].flow, c = E[j^1].cap, len = E[j^1].cost;
44             if(f < c && dist[v] > dist[u] + len) {
45                 dist[v] = dist[u] + len;
```

```
46          if(!inq[v]) {
47              inq[v] = true; Q.push(v);
48          }
49       }
50     }
51   }
52   return dist[s] != LL_INF;
53 }
54
55 bool Dijkstra() {
56   PairingHeap pq;
57   memset(dist, 0x3f, sizeof(dist));
58   memset(it, 0, sizeof(it));
59   dist[t] = 0; it[t] = pq.push(make_pair(dist[t], t));
60   while(!pq.empty()) {
61     HeapNode t = pq.top(); pq.pop();
62     int u = t.snd;
63     for(int j=head[u]; j; j=E[j].next) {
64       int v = E[j].v; int64 f = E[j^1].flow, c = E[j^1].cap, len = E[j^1].cost;
65       if(f < c && dist[v] > dist[u] + len) {
66         dist[v] = dist[u] + len;
67         if(it[v] == NULL)
68           it[v] = pq.push(make_pair(dist[v], v));
69         else
70           pq.modify(it[v], make_pair(dist[v], v));
71       }
72     }
73   }
74   return dist[s] != LL_INF;
75 }
76
77 int64 dfs(int u, int64 flow) {
78   if(u == t || flow == 0) return flow;
79   vis[u] = true;
80   int64 res = flow;
81   for(int j=head[u]; j; j=E[j].next) {
82     int v = E[j].v; int64 f = E[j].flow, c = E[j].cap, len = E[j].cost;
83     if(f < c && !vis[v] && len == 0) {
84       int64 aug = dfs(v, min(res, c-f));
85       E[j].flow += aug; E[j^1].flow -= aug;
86       res -= aug;
87       if(res == 0) break;
88     }
89   }
90   return flow - res;
91 }
92
93 void Augment() {
94   int64 CurFlow = 0;
95   while( memset(vis, 0, sizeof(vis)),
96     (CurFlow = dfs(s, LL_INF)) ) {
97     MaxFlow += CurFlow;
98     MinCost += delta * CurFlow;
99   }
100 }
101
102 void PrimalDual() {
```

```
103        if(!BellmanFord()) return;
104        Reduce(); Augment();
105        while(Dijkstra()) {
106            Reduce(); Augment();
107        }
108    }
109
110    template<typename T>
111    inline void readint(T &x) {
112        T f=1, r=0; char c=getchar();
113        while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
114        while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
115        x = f*r;
116    }
117
118    int main() {
119        int u, v; int64 w, c;
120        readint(n); readint(m); readint(s); readint(t);
121        for(int i = 1; i <= m; i++) {
122            readint(u); readint(v); readint(w); readint(c);
123            add_edge(u, v, w, c);
124        }
125        PrimalDual();
126        printf("%lld %lld", MaxFlow, MinCost);
127        return 0;
128    }
```

4 树

4.1 倍增 LCA

Tree/DoublingLCA.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

struct Edge { int v, next; };

const int MAXN = 1e6, LOG = 20;
int n, q, s, e_ptr = 1, head[MAXN+10]; Edge E[(MAXN+10)<<1];
int dep[MAXN+10], anc[MAXN+10][LOG+1];

void add_edge(int u, int v) { E[++e_ptr] = (Edge) { v, head[u] }; head[u] = e_ptr; }
void add_pair(int u, int v) { add_edge(u, v); add_edge(v, u); }

void dfs(int u) {
    for(int i = 1; i <= LOG; i++)
        anc[u][i] = anc[anc[u][i-1]][i-1];
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v;
        if(v == anc[u][0]) continue;
        anc[v][0] = u; dep[v] = dep[u] + 1;
        dfs(v);
    }
}

int lca(int u, int v) {
    if(dep[u] < dep[v]) swap(u, v);
    for(int i = LOG; i >= 0; i--)
        if(dep[anc[u][i]] >= dep[v])
            u = anc[u][i];
    if(u == v) return u;
    for(int i = LOG; i >= 0; i--)
        if(anc[u][i] != anc[v][i])
            u = anc[u][i], v = anc[v][i];
    u = anc[u][0], v = anc[v][0];
    return u;
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

int main() {
    int u, v;
    n = readint(); q = readint(); s = readint();
    for(int i = 1; i <= n-1; i++) {
        u = readint(); v = readint();
        add_pair(u, v);
    }
    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  }
```

4.2 欧拉序列求 LCA

Tree/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(idx[u], idx[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        add_pair(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 }
```

4.3 树链剖分

Tree/HLD.cpp

```
1  // call Dfs1(1) and Dfs2(1, 1)
2  const int MAXN = 1e5;
3  int dfs_clock, Fa[MAXN+10], Son[MAXN+10], Sz[MAXN+10],
4      Dep[MAXN+10], Top[MAXN+10], Dfn[MAXN+10];
5
6  void Dfs1(int u) { // Fa Son Sz Dep
7      int maxsz = 0; Sz[u] = 1;
8      for(int j=head[u]; j; j=E[j].next) {
9          int v = E[j].v;
10         if(v == Fa[u]) continue;
11         Fa[v] = u; Dep[v] = Dep[u] + 1; // !
12         Dfs1(v); Sz[u] += Sz[v];
13         if(Sz[v] > maxsz) {
14             maxsz = Sz[v];
15             Son[u] = v;
16         }
17     }
18 }
19
```

```
20  void Dfs2(int u, int anc) { // Top Dfn
21      Dfn[u] = ++dfs_clock; Top[u] = anc;
22      if(Son[u]) Dfs2(Son[u], anc);
23      for(int j=head[u]; j; j=E[j].next) {
24          int v = E[j].v;
25          if(v == Fa[u] || v == Son[u]) continue;
26          Dfs2(v, v);
27      }
28  }
29
30  int LCA(int u, int v) {
31      while(Top[u] != Top[v]) {
32          if(Dep[Top[u]] < Dep[Top[v]]) swap(u, v);
33          u = Fa[Top[u]];
34      }
35      if(Dep[u] > Dep[v]) swap(u, v);
36      return u;
37  }
38
39  int HLDQuery(int u, int v) {
40      int ret = -INF;
41      while(Top[u] != Top[v]) {
42          if(Dep[Top[u]] < Dep[Top[v]]) swap(u, v);
43          ret = max(ret, st_query(1, 1, n, Dfn[Top[u]], Dfn[u]));
44          u = Fa[Top[u]];
45      }
46      if(Dep[u] > Dep[v]) swap(u, v);
47      ret = max(ret, st_query(1, 1, n, Dfn[u], Dfn[v]));
48      return ret;
49  }
```

4.4   点分治

5 单调数据结构

5.1 单调队列 (滑动窗口)

Monotonic/SlidingWindow.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 1e6;

int n, k, Hd, Tl, A[MAXN+10], Q[MAXN+10];

void SlideMin() {
    Hd = 1, Tl = 0;
    for(int i = 1; i <= k; i++) {
        while(Hd <= Tl && A[Q[Tl]] >= A[i]) Tl--;
        Q[++Tl] = i;
    }
    printf("%d ", A[Q[Hd]]);
    for(int i = k+1; i <= n; i++) {
        while(Hd <= Tl && Q[Hd] < i-k+1) Hd++;
        while(Hd <= Tl && A[Q[Tl]] >= A[i]) Tl--;
        Q[++Tl] = i;
        printf("%d ", A[Q[Hd]]);
    }
}

void SlideMax() {
    Hd = 1, Tl = 0;
    for(int i = 1; i <= k; i++) {
        while(Hd <= Tl && A[Q[Tl]] <= A[i]) Tl--;
        Q[++Tl] = i;
    }
    printf("%d ", A[Q[Hd]]);
    for(int i = k+1; i <= n; i++) {
        while(Hd <= Tl && Q[Hd] < i-k+1) Hd++;
        while(Hd <= Tl && A[Q[Tl]] <= A[i]) Tl--;
        Q[++Tl] = i;
        printf("%d ", A[Q[Hd]]);
    }
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

int main() {
    n = readint(); k = readint();
    for(int i = 1; i <= n; i++) A[i] = readint();
    SlideMin();
    putchar(10);
    SlideMax();
    return 0;
```

```
52  }
```

5.2   单调栈

6 线段树

6.1 Lazy-Tag

6.2 动态开点线段树

6.3 可持久化线段树

7 离线二维数点

7.1 带修改

7.1.1 静态：线段树 + 扫描线

7.1.2 动态：CDQ 分治

8 在线二维数点

8.0.1 动态：二维线段树

时间复杂度 插入$O(\lg^2 n)$ – 查询$O(\lg n)$ 空间复杂度 $O(n^2)$

8.0.2 动态：树状数组套动态开点线段树

8.0.3 动态：树状数组套平衡树

9 平衡树

9.1 Treap

9.2 Splay

10　动态树

10.1　Link-cut Tree

11 字符串

11.1 KMP 字符串匹配

1-indexed

11.2 AC 自动机

0-indexed

<div align="center">String/ACAutomaton.cpp</div>

```cpp
#include <bits/stdc++.h>
#define CLEAR(x) memset((x), 0, sizeof(x))
using namespace std;

const int SIGMA = 26, MAX_TEMP_LEN = 70, MAXN = 150,
MAX_LEN = 1e6, MAX_NODE = MAXN * MAX_TEMP_LEN;

int N, sz, ch[MAX_NODE + 10][SIGMA + 2], f[MAX_NODE + 10], last[MAX_NODE+10],
    val[MAX_NODE + 10], found_cnt[MAX_NODE+10];
char str[MAX_LEN+10], tpl[MAXN+10][MAX_TEMP_LEN+10];
unordered_map<string, int> ms;

inline int idx(char c) { return c - 'a' + 1; }

void insert(char *str) {
    int u = 0, len = strlen(str);
    for(int i = 0; i < len; i++) {
        int c = idx(str[i]);
        if(!ch[u][c]) ch[u][c] = ++sz;
        u = ch[u][c];
    }
    ms[string(str)] = u;
    ++val[u];
}

void get_fail() {
    queue<int> Q;
    f[0] = 0;
    for(int c = 1; c <= SIGMA; c++) if(ch[0][c]) {
        int v = ch[0][c];
        f[v] = last[v] = 0;
        Q.push(v);
    }
    while(!Q.empty()) {
        int u = Q.front(); Q.pop();
        for(int c = 1; c <= SIGMA; c++) {
            int v = ch[u][c];
            if(!v) {
                ch[u][c] = ch[f[u]][c];
                continue;
            }

            Q.push(v);

            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  }
```

11.3 后缀数组

0-indexed

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAXLEN = 1e6, SIGMA = 100;

inline int idx(char c) {
    if(!c) return 0;
    else if(isdigit(c)) return c - '0' + 1;
    else if(isupper(c)) return c - 'A' + 1 + 10;
    else if(islower(c)) return c - 'a' + 1 + 10 + 26;
    else throw "Invalid Character";
}

struct SuffixArray {
    int sa[MAXLEN+10], rk[MAXLEN+10], buf[3][MAXLEN+10], height[MAXLEN+10], c[MAXLEN+10];
    void build_sa(char *s, int len) {
        int m = SIGMA + 10, n = len + 1, *x = buf[0], *y = buf[1];
        for(int i = 0; i < m; i++) c[i] = 0;
        for(int i = 0; i < n; i++) ++c[x[i] = idx(s[i])];
        for(int i = 1; i < m; i++) c[i] += c[i-1];
        for(int i = n-1; i >= 0; i--) sa[--c[x[i]]] = i;
        for(int k = 1; k <= n; k <<= 1) {
            int p = 0;
            for(int i = n-k; i < n; i++) y[p++] = i;
            for(int i = 0; i < n; i++) if(sa[i] >= k) y[p++] = sa[i] - k;
            for(int i = 0; i < m; i++) c[i] = 0;
            for(int i = 0; i < n; i++) ++c[x[y[i]]];
            for(int i = 1; i < m; i++) c[i] += c[i-1];
            for(int i = n-1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
            swap(x, y);
            p = 1, x[sa[0]] = 0;
            for(int i = 1; i < n; i++)
                x[sa[i]] = (y[sa[i]] == y[sa[i-1]] && y[sa[i] + k] == y[sa[i-1] + k] ? p-1 : p++);
            if(p >= n) break;
            m = p;
        }
        memcpy(rk, x, sizeof(rk));
        int k = 0;
        for(int i = 0; i < n; i++) {
            if(!rk[i]) continue;
            if(k) k--;
            int j = sa[rk[i]-1];
            while(s[i+k] == s[j+k]) k++;
            height[rk[i]] = k;
        }
    }
} SA;
inline void readstr(char* str) {
    char c=getchar(); int p=0;
    while(!isalnum(c) && !ispunct(c)) c=getchar();
    while(isalnum(c) || ispunct(c)) {
        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  }
```

12 Miscellaneous

12.1 ST 表

12.2 Fenwick Tree

12.3 左偏树

13 悬线法

13.1 Algorithm 1

13.2 Algorithm 2

14 莫队

14.1 普通莫队

14.2 带修改莫队

15　分块相关

15.1　分块

15.2　区间众数

15　分块相关