Contents

1 Setups

1.1 vimrc [6c9876]

```
se nu ai rnu cin ts=4 sw=4 | sy on
inoremap {<CR> {<CR>}<Esc>O
inoremap jk <Esc>
```

1.2 pbds [9f7c3e]

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

template <typename T>
using ordered_set = tree<T,null_type,less
    <T>,rb_tree_tag,tree_order_statistics_node_update>;

int main(){
    ordered_set<int> st;
    st.insert(1);
    st.find_by_order(0);//iterator to 1
    st.order_of_key(1);//returns 0
}
```

1.3 terminal [46fc34]

```
-- terminal --
$ setxkbmap -option caps:swapescape
```

1.4 debug [33c0d3]

```
#ifdef MIKU
string
     dbmc = "\033[1;38;2;57;197;187m", dbrs = \033[0m;
#define debug
    (x...) cout << dbmc << "[" << #x << "] : ", dout(x)
void dout() { cout << dbrs << endl; }
template <typename T, typename ...U>
void dout(T t, U ...u) { cout
     << t << sizeof...(u) ? ", " : ""; dout(u...); }
#else
#define debug(...) 39
#endif

int main(){
    int a = 49;
    char c = '8';
    debug(a);// outputs "[a] : 49"
    debug(a, c); // outputs "[a, c] : 49, 8"
    debug("PCCORZ"); // outputs "["PCCORZ"] : PCCORZ"
    debug(); // outputs "[] : "
}
```

1.5 template [5116af]

```
#include <bits/stdc++.h>
using namespace std;
#define fs first
#define sc second
#define F first
#define S second
#define FOR(i, j, k) for (int i = j, Z = k; i < Z; i++)
using ll = long long;
using lll = __int128_t;
typedef pair<int, int> pii;
typedef tuple<int,int,int> tiii;
typedef pair<ll,ll> pll


int main(){
}
```

2 Graph

2.1 Dominator Tree [3b89c3]

```
struct DominatorTree{
  //1-indexed
  //not reachable from s -> not on tree
  int n;
  vector<vector<int>> G,rG;
  vector<int> pa,dfn,id;
  int dfnCnt;
  vector<int> semi,idom,best;
  vector<vector<int>> ret;
  void init(int _n){
    n=_n;
    G = rG = ret = vector<vector<int>>(n+1);
    pa = dfn = id = vector<int>(n+1,-1);
    dfnCnt = 0;
    semi = idom = best = vector<int>(n+1,-1);
  }
  void add_edge(int u,int v){
    G[u].push_back(v);
    rG[v].push_back(u);
  }
  void dfs(int u){
    id[dfn[u]=++dfnCnt]=u;
    for(auto v:G[u]) if(!dfn[v]){
      dfs(v),pa[dfn[v]]=dfn[u];
    }
  }
  int find(int y,int x){
    if(y<=x)return y;
    int tmp=find(pa[y],x);
    if(semi[best[y]]>semi[best[pa[y]]])
      best[y]=best[pa[y]];
    return pa[y]=tmp;
  }
  void tarjan(int root){
    dfnCnt=0;
    for(int i=1;i<=n;++i){
      dfn[i]=idom[i]=0;
      ret[i].clear();
      best[i]=semi[i]=i;
    }
    dfs(root);
    for(int i=dfnCnt;i>1;--i){
      int u=id[i];
      for(auto v:rG[u]) if(v=dfn[v]){
        find(v,i);
        semi[i]=min(semi[i],semi[best[v]]);
      }
      ret[semi[i]].push_back(i);
      for(auto v:ret[pa[i]]){
        find(v,pa[i]);
        idom[v
            ] = semi[best[v]]==pa[i] ? pa[i] : best[v];
      }
      ret[pa[i]].clear();
    }
    for(int i=2; i<=dfnCnt; ++i){
      if(idom[i]!=semi[i]) idom[i]=idom[idom[i]];
      ret[id[idom[i]]].push_back(id[i]);
    }
  }
  vector<vector<int>> solve(int s){
    tarjan(s);
    return ret;
  }
};
```

2.2 Incremental SCC [d8b556]

```
struct IncrementalSCC{
#define pii pair<int,int>
#define fs first
#define sc second
#define tiii tuple<int,int,int>
  //if u == v : ans[i] = -1
  //if not connected : ans[i] = m
  //all 0-indexed
  int n;
  vector<int> ans;
  int m;
  vector<tiii> all;
  vector<int> SCC(int n,vector<vector<int>>& paths){
    vector<int> scc_id(n,-1),idx(n,-1),low(n,-1),st;
    int cnt = 0,gcnt = 0;
```

```
  function<void(int)> dfs = [&](int now)->void{
    low[now] = idx[now] = cnt++;
    st.push_back(now);
    for(auto nxt:paths[now]){
      if(scc_id[nxt] != -1)continue;
      if(idx[nxt] == -1){
        dfs(nxt);
        low[now] = min(low[now],low[nxt]);
      }
      else{
        low[now] = min(low[now],idx[nxt]);
      }
    }
    if(low[now] == idx[now]){
      int id = -1;
      while(id != now){
        id = st.back();
        st.pop_back();
        scc_id[id] = gcnt;
      }
      gcnt++;
    }
  };
  for(int i = 0;i<n;i++){
    if(scc_id[i] == -1)dfs(i);
  }
  //cerr<<"SCC: "<<n<<"::";for(int
       i = 0;i<n;i++)cerr<<scc_id[i]<<',';cerr<<endl;
  return scc_id;
}
vector<int> mapping;
void dc(int l,int r,vector<tiii> &edges){
  //cerr<<l<<' '<<r<<":"<<endl;
  if(l == r){
    for(auto
        &[id,_,__]:edges)ans[id] = min(ans[id],l);
    return;
  }
  int mid = (l+r)>>1;
  int cnt = 0;
  for(auto &[t,u,v]:edges){
    if(mapping[u] == -1)mapping[u] = cnt++;
    if(mapping[v] == -1)mapping[v] = cnt++;
  }
  n = cnt;
  vector<vector<int>> paths(n);
  vector<int> vv;
  for(auto &[t,u,v]:edges){
    vv.push_back(u);
    vv.push_back(v);
    u = mapping[u],v = mapping[v];
    if(t<=mid)paths[u].push_back(v);
  }
  //for(auto &i:vv)cerr<<i<<',';cerr<<endl;
  for(auto &i:vv)mapping[i] = -1;

  auto scc_id = SCC(n,paths);
  //for(auto
       [t,u,v]:edges)cerr<<t<<','<<u<<','<<v<<endl;
  //cerr<<endl;
  vector<tiii> vl,vr;
  for(auto &[t,u,v]:edges){
    if(scc_id[u] == scc_id[v]){
      ans[t] = min(ans[t],mid);
      vl.push_back(tiii(t,u,v));
    }
    else{
      u = scc_id[u],v = scc_id[v];
      vr.push_back(tiii(t,u,v));
    }
  }
  vector<tiii>().swap(edges);
  dc(l,mid,vl);
  dc(mid+1,r,vr);
  return;
}
void add_edge(int u,int v){
  all.push_back(tiii(all.size(),u,v));
}
vector<tiii> solve(){//[time,u,v]
  m = all.size();
  vector<tiii> ret(m);
  for(auto [t,u,v]:all)ret[t] = tiii(m,u,v);
  for(auto [t,u,v]:all)n = max({n,u,v});
  n++;
  ans = vector<int>(m,m);
  for(auto [t,u,v]:all){
    if(u == v)ans[t] = -1;
  }
  mapping = vector<int>(n,-1);
  dc(0,m,all);
  for(int i = 0;i<m;i++)get<0>(ret[i]) = ans[i];
  return ret;
}
IncrementalSCC(){
  ans.clear();
  n = m = 0;
}
#undef tiii
#undef pii
#undef fs
#undef sc
};
```

2.3 Round Square Tree [2de108]

```
struct RoundSquareTree{
  //0-indexed
  //returns a forest if the graph is not connected
  vector<vector<int>> g;
  vector<vector<int>> groups;
  vector<vector<int>> tr;
  vector<int> idx,low,st;
  int cnt,gcnt;
  int n;
  RoundSquareTree(int _n = 0){
    cnt = gcnt = 0;
    n = _n;
    g = vector<vector<int>>(n);
  }
  void add_edge(int a,int b){//adds bidirectional edges
    g[a].push_back(b);
    g[b].push_back(a);
  }
  void dfs(int now){
    idx[now] = low[now] = cnt++;
    st.push_back(now);
    for(auto nxt:g[now]){
      if(idx[nxt] == -1){
        dfs(nxt);
        low[now] = min(low[now],low[nxt]);
        if(low[nxt] == idx[now]){
          int id = -1;
          tr.push_back(vector<int>());
          while(id != nxt){
            id = st.back();st.pop_back();
            groups[id].push_back(gcnt);
            tr[id].push_back(gcnt+n);
            tr[gcnt+n].push_back(id);
          }
          groups[now].push_back(gcnt);
          tr[now].push_back(gcnt+n);
          tr[gcnt+n].push_back(now);
          gcnt++;
        }
      }
      else idx[now] = min(idx[now],idx[nxt]);
    }
    return;
  }
  vector<vector<int>> solve(){//
      returns the tree (round vertices numbered [0,n))
    idx = low = vector<int>(n,-1);
    tr = vector<vector<int>>(n);
    for(int i = 0;i<n;i++){
      if(idx[i] == -1)dfs(i);
    }
    return tr;
  }
};
```

3 Data Structure
4 String
5 Math
6 Geometry