Пример решения десятой задачи

Рассмотрим $G = (V, E) \quad : \quad$ связный, ациклический, неориентированный $\equiv$ дерево
Определим функцию расстояния для пар вершин: $d \quad : \quad V \times V \to R$
$\forall v, u \in V(G) \quad (u \neq v) \quad d(v, u) = |\{e \quad : \quad e \in v \rightsquigarrow u\}|$
Так как рассматриваемый граф - дерево, то для всех пар вершин расстояние определяется однозначно.
Задача: $\forall v \in V(G)$ найти $u \quad : \quad d(v, u) = d$ или определить, что $u \notin V(G)$
Решение:

1.

   Определение: центральная вершина дерева
   Пусть $r(p) = \sup\limits_{u:u \in V(G)} d(p, u)$, тогда $p -$ центральная, если $r(p) = \inf\limits_{v:v \in V(G)} r(v)$
   Найти центральную вершину можно сделав 2 поиска в ширину.
   Рассмотрим $v, u \in V(G)$ если $r(v) = d(v, u)$ то запустив поиск в ширину из u мы найдем пару наиболее удаленных друг от друга вершин, тогда гарантируется, что центральная вершина дерева находится в середине пути между ними.
   Подвесим дерево за одну из центральных вершин и определим функцию высоты узла: $\forall v \in V(G) \quad h(v) = d(root, v),$ а также то, какому из поддеревьев корня принадлежит текущая вершина:
   $t(v) = u \quad : \quad (u, root) \in E(G) \quad v \in T_u,$ где $T_u \subseteq G \quad : \quad root(T_u) = u$

2.

   Разделим все запросы вида $v \quad d$ на три типа:
   1) $\quad h(v) = d,$ тогда ответом, очевидно, будет $root$
   2) $\quad d > h(v),$ тогда найти $u \quad : \quad u \notin T_{t(v)}$ и $h(u) = d - h(v),$ сделать это можно бинарным поиском за $O(log|V(G)|),$ или определить, что такой вершины нет.
   Этот вариант поиска ответа не был бы правельным если бы дерево не было подвешено за одну из центральных вершин.

3) $d < h(v)$, тогда найти u : $u \in T_{t(v)}$ и $u \in v \rightsquigarrow root$ $h(u) = h(v) - d$
Быстро найти такую вершину можно методом двоичного подъема.
Необходимо посчитать функцию $up[i][v]$ – это вершина на пути к корню от вершины v, которая находится на расстоянии 2^i

- $up[0][v] = parent[v]$— это предок вершины v в дереве (как раз на расстоянии $2^0 = 1$)
- $up[i][v] = up[i-1][up[i-1][v]]$— дважды поднялись на 2^{i-1}

Представив d в двоичной системе, сможем найти u за $O(log|V(G)|)$

Временная сложность данного решения $O(Q \cdot log|V(G)|)$
Код на C++

```
#define FOR(i, n) for(int i=0; i< (int)n; i++)

int n, q, root, l;
vector< vector<int> > g, h, up;
vector<int> p, d, part;

int bfs(int s, bool f) {
  d.assign(n, 0);
  p.assign(n, -1);
  queue<int> q; q.push(s);
  int far = s;

  while(!q.empty()) {
    int v = q.front(); q.pop();

    for(auto to : g[v]) {
      if(to != p[v]) {
        p[to] = v;
        d[to] = d[v] + 1;
        if(d[far] < d[to]) far = to;
        q.push(to);
      }
    }
  }
```

```
  if(f) {
    vector<int> path;
    for(int v = far; v!=-1; v = p[v])
      path.push_back(v);
    return path[ path.size()/2 ];
  }

  return far;
}

void dfs(int v, int prt) {
  part[v] = prt;
  h[ d[v] ].push_back(v);

  up[v][0] = p[v];
  for(int i=1; i<=l; i++)
    up[v][i] = up[ up[v][i-1] ][i-1];

  for(auto to : g[v])
    if(to != p[v]) {
      p[to] = v;
      d[to] = d[v] + 1;
      dfs(to, prt);
    }
}

int upper_bound(int prt, int dist) {
  int first = 0, last = h[dist].size();
  int count = last-first, step, it;

  while(count > 0) {
    it = first; step = count/2; it+=step;
    if(!(prt < part[ h[dist][it] ]))
      first=++it, count -= step+1;
    else count = step;
  }

  if(h[dist].size()==0) return -1;

  if(first == h[dist].size()) first--;
  return h[dist][first];
}

inline int log2(int x) {
  return floor(log((double)x)/log(2.0));
```

```
}

int rise(int v, int dist) {
  int p = log2(dist), delta;
  delta = dist - pow(2.0, p);
  if(delta == 0)
    return up[v][p];
  else
    return rise(up[v][p], delta);
}

int find(int v, int dist) {
  if(d[v] == dist) return root;

  if(dist > d[v]) {
    dist -= d[v];
    int u = upper_bound(part[v], dist);

    if(u==-1) return -1;

    if(part[u] == part[v]) u = h[dist][0];
    if(part[u] == part[v]) u = -1;
    return u;
  }

  return rise(v, dist);

  return -1;
}

int main()
{
#ifndef ONLINE_JUDGE
  fstream cin("test.txt");
#endif
  cin >> n >> q;
  g.resize(n), up.resize(n);
  h.resize(n);
  FOR(i, n-1) {
    int v, u; cin >> v >> u;
    v--, u--;
    g[v].push_back(u), g[u].push_back(v);
  }

  int far = bfs(0, false);
```

```
  root = bfs(far, true);

  part.resize(n); p.assign(n, -1); d.assign(n,0);
  p[root] = 0;

  l=1;
  while((1<<l) <= n) ++l;
  FOR(i, n) up[i].resize(l+1);

  FOR(i, g[root].size()) {
    d[ g[root][i] ] = 1; p[ g[root][i] ] = root;
    dfs(g[root][i], i);
  }

  FOR(i, q) {
    int v, d, u; cin >> v >> d;
    u = --v;
    if(d!=0) {
      if(g[v].size()==0) u = -1;
      else if(d==1 && g[v].size()!=0) u = g[v][0];
      else if(v==root) {
        if(h[d].size()) u = h[d][0];
        else u = -1;
      }
      else u = find(v, d);
    }
    cout << u+1 << endl;
  }

  return 0;
}
```