جامعة دمشق

كلية الهندسة المعلوماتية

السنة الثانية

الخوارزميات وبنى المعطيات /٢/

الوظيفة الفصلية

المشروع /٥/

تقديم الطلاب:

رائد محمد زهير السبيناتي

راما معين ريحاوي

رغد عامر الحلبي

بإشراف:

م.سعيد سريحني

الاثنين 3/8/2020

المسألة الأولى:

✓ الطريقة الأولى:

الكود البرمجي:

```
#include <bits/stdc++.h>

using namespace std;
vector < int >l, r, lvl, p, sum;
int q, n;

void dfs (int x = 1, int pa = 0)
{
 if (x == -1)
  return;
 lvl[x] = lvl[pa] + 1;
 sum[x] = sum[pa] + x;
 p[x] = pa;
 dfs (l[x], x), dfs (r[x], x);
}

int lca (int u, int v)
{
 if (lvl[u] > lvl[v])
  swap (u, v);

 while (lvl[v] != lvl[u])
  v = p[v];
 while (p[u] != p[v])
  u = p[u], v = p[v];
 return (u==v)?u:p[u];
}

double res (int u, int v)
{
 int a = lca (u, v);
 double r = sum[u] + sum[v] - 2 * sum[a] + a;
 double d = lvl[u] + lvl[v] - 2 * lvl[a] + 1;
 return r / d;
}
```

```
int main ()
{
 cin >> n;

 l.resize(n+1),r.resize(n+1),lvl.resize(n+1);
 p.resize(n+1),sum.resize(n+1);
 for (int i = 1; i <= n; i++)
     cin >> l[i] >> r[i];

 dfs ();
 cin >> q;
 while (q--)
  {
   int u, v;
   cin >> u >> v;
   cout << res (u, v) << endl;
  }
  return 0;
}
```

بنى المعطيات المستخدمة:

Vector <int> l,r: لتخزين الشجرة المدخلة حيث أن l[i]:left تدل على الابن اليسار للأب الذي دليله i و r[i]:right تدل على الابن اليسار للأب الذي دليله i.

Vector<int> lvl: لتخزين مستوى كل عقدة من العقدة بدءا من المستوى ١.

Vector<int> p: لتخزين آباء العقد.

Vector<int> sum: لتخزين كمية البنزين من العقدة الحالية وحتى الجذر أي مجموع العقد عند الـ route من الجذر إلى العقدة الحالية.

خوارزمية الحل:

أولاً: تقوم الإجرائية dfs() بحساب مستوى كل عقدة وأبوها ومجموع كمية البنزين من عندها وحتى الجذر وتقوم بهذا عودياً حيث:

مستوى كل عقدة هو مستوى أبوها +١.

وكذلك المجموع من الجذر حتى العقدة الحالية هو المجموع من الجذر حتى أب العقدة الحالية+العقدة الحالية.

ثانياً: يقوم التابع lca(int u,int v) بإيجاد أقرب سلف مشترك لعقدتين حيث في البداية إذا كان مستوى العقدة u أكبر من مستوى العقدة v نقوم بالتبديل بينهما لأننا على أي حال سنقوم بجعل العقدة v في نفس مستوى العقدة u حيث نبدلها بأبيها هكذا حتى تصبحان في نفس المستوى ثم نقوم برفع كلتا العقدتين حتى يصبح أباهما نفسه وذلك من خلال تبديل u,v بأبيهما.

ثم نتحقق من الشرط أنه إذا كانت u وv متساويتان فهذا يعني أن إحداهما كانت سلفا للأخرى عندها يكون أقرب سلف لهما هو إحداهما وأما إذا لم يتحقق هذا الشرط عندها يكون lca للعقدتين u,v هو الأب المشترك لسلفهما.

ثالثاً: يقوم التابع res(int u,int v) بحساب المطلوب(المتوسط الحسابي لمجموع العقد على الطريق الأقصر بين (u,v حيث نوجد lca هاتين العقدتين فيكون مجموع كمية البنزين على طول هذا الطريق (من u إلى lca إلى v) مساوياً لـ:

$$Sum(u) + sum(v) - 2 * sum(lca(u,v)) + lca(u,v)$$

قمنا بطرح ضعفي المجموع عند lca لأننا عندما جمعنا المجموع من u حتى الجذر مع v حتى الجذر جمعنا المجموع من lca حتى الجذر مرتين لذلك نقوم بطرحه وبما أننا طرحنا أيضا كمية البنزين عند lca فنعيد جمعها.

ونوجد أيضاً عدد العقد على الطريق ويكون مساوياً لـ:

$$lvl(u) + lvl(v) - 2 * lvl(lca(u,v)) + 1$$

تعقيد الخوارزمية الزمني هو N+Q.h

و

تعقيد الخوارزمية المكاني هو N

✓ **الطريقة الثانية:**

الكود البرمجي:

```
#include <bits/stdc++.h>
using namespace std;
int n,lg,q;
vector<int> l,r,lvl;
vector<vector <int>> p,s;
void dfs(int x=1,int pa=0)
{
  if(x==-1) return;
  lvl[x]=lvl[pa]+1,p[x][0]=pa,s[x][0]=x;
```

```
  for(int j=1;j<=lg;j++)
  {
    p[x][j]=p[ p[x][j-1] ][j-1];
    s[x][j]=s[x][j-1]+s[ p[x][j-1] ][j-1];
  }

  dfs(l[x],x),dfs(r[x],x);
}
int lca(int u,int v)
{
  if(lvl[u]>lvl[v]) swap(u,v);
  int d=lvl[v]-lvl[u];

  for(int i=lg;i>=0;i--)
    if(d>>i&1)
      v=p[v][i];

  for(int i=lg;i>=0;i--)
    if(p[u][i]!=p[v][i])
      u=p[u][i],v=p[v][i];

  return (u==v)?u:p[u][0];
}

double res(int u,int v)
{
  int a=lca(u,v);
  double r=s[u][lg]+s[v][lg]-2*s[a][lg]+s[a][0];
  double d=lvl[u]+lvl[v]-2*lvl[a]+1;
  return r/d;
}

int main()
{
  cin>>n;
  lg=log2(n)+1;
  l.resize(n+1),r.resize(n+1),lvl.resize(n+1);
  p.resize(n+1),s.resize(n+1);

  for(int i=0;i<=n;i++)
    p[i].resize(lg+1),s[i].resize(lg+1);

  for(int i=1;i<=n;i++)
```

```
    cin>>l[i]>>r[i];

  dfs();
  cin>>q;

  while(q--)
  {
    int u,v;cin>>u>>v;
    cout<<res(u,v)<<endl;
  }
  return 0;
}
```

بنى المعطيات المستخدمة: Sparse table

تعقيد الخوارزمية الزمني هو N+Q.log(N)
و
تعقيد الخوارزمية المكاني هو N.log(N)

المسألة الثانية:

الكود البرمجي:

```
#include <bits/stdc++.h>
using namespace std;
vector <string> s;
vector <vector <int>> g;
vector <bool> vis;


void dfs(int x)
{
   vis[x]=1;
   for(int y:g[x])
      if(!vis[y])
         dfs(y);
}
int main()
```

```
{
  int n,m,cmp=0;
  cin>>n>>m;
  s.resize(n+1),g.resize(n+1),vis.resize(n+1);

  for(int i=1;i<=n;i++)
    cin>>s[i];

  for(int i=1;i<=n;i++)
  {
    for(int j=i+1;j<=n;j++)
    {
      int cnt=0;

      for(int c=0;c<m;c++)
        if(s[i][c]!=s[j][c])
          cnt++;

      if(cnt<=1)
        g[i].push_back(j),g[j].push_back(i);
    }
  }

  for(int i=1;i<=n;i++)
    if(!vis[i])
      dfs(i),cmp++;

  cout<<cmp;
  return 0;
}
```

بنى المعطيات المستخدمة:

vector <string> s: لتخزين سلاسل القاموس.

vector<vector<int>> g: لربط السلاسل التي تنتج عن بعضها بتغيير محرف واحد فقط.

vector <bool> vis: لتحديد رقم السلسلة المزارة.

خوارزمية الحل:

نقوم أولاً بالمرور على السلاسل ومقارنة كل سلسلة مع السلاسل التي تليها من حيث عدد المحارف المختلفة، إذا اختلفت سلسلة عن الأخرى بمحرف واحد على الأكثر (أي كل منهما تنتج عن الأخرى بتغيير محرف واحد) سنخزن هذه الوصلة في g (undirected graph)

ثم نستدعي التابع dfs عند العقد غير المزارة حيث سيقوم التابع بزيارة كل العقد المرتبطة بالعقدة الحالية داخل الغراف وبالتالي تكون عدد مرات استدعاء التابع هي عدد الـ "components" في الغراف وهي أقل عدد من السلاسل الواجب تجريبها على القفل حتى يفتح على إحداها بالتأكيد.

تعقيد الخوارزمية الزمني هو $N^2.M$

9

تعقيد الخوارزمية المكاني هو N.M