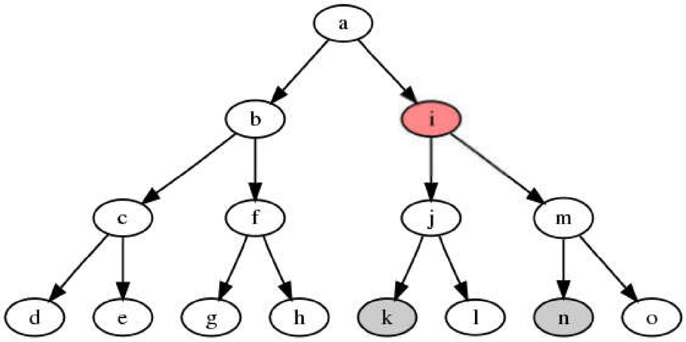
Home
অ্যালগরিদম নিয়ে যত লেখা!
আমার সম্পর্কে...

লোয়েস্ট কমন অ্যানসেস্টর

মার্চ ১৩, ২০১৪ by শাফায়েত

লোয়েস্ট কমন অ্যানসেস্টর জিনিসটা শুনতে একটু কঠিন মনে হলেও জিনিসটা সহজ আর খুবই কাজের। বেশ কিছু ধরণের প্রবলেম সলভ করে ফেলা যায় এই অ্যালগোরিদম দিয়ে। আমরা প্রথমে লোয়েস্ট কমন অ্যানসেস্টর বের করার ব্রুটফোর্স অ্যালগোরিদম দেখবো, তারপর স্পার্স টেবিল নামের নতুন একটা ডাটা স্ট্রাকচার শিখে কমপ্লেক্সিটি লগ এ নামিয়ে আনবো।

প্রথমেই আমরা জেনে নেই লোয়েস্ট কমন অ্যানসেস্টর বা এল.সি.এ(LCA) কি সেটা। নিচের ছবিটা দেখ:



ছবিতে k আর n নোডের প্যারেন্ট ধরে পিছে যেতে থাকলে তারা i নোডে এসে মিলবে। i হলো k,n এর লোয়েস্ট কমন অ্যানসেস্টর। ‘a’ ও দুইজনের কমন অ্যানসেস্টর কিন্তু i হলো ‘লোয়েস্ট’ বা সবথেকে কাছাকাছি।

top

মনে করো ট্রি টা দিয়ে বুঝানো হচ্ছে একটা অফিসে কে কার সুপারভাইজর বা বস। a হলো অফিসের হেড, b এবং i এর বস হলো a, আবার c এবং f এর বস হলো b ইত্যাদি। তাহলে লোয়েস্ট কমন অ্যানসেস্টর অ্যালগোরিদম দিয়ে দুইজন এমপ্লয়ির লোয়েস্ট/সব থেকে কাছের কমন বস খুজে বের করে ফেলতে পারি সহজেই।

প্রথমে আমরা একদম নেইভ একটা উপায় দেখি যেটা খুব বড় ইনপুটের জন্য কাজ করবেনা। মনে করি আমরা k,n এর এল.সি.এ বের করতে চাই, প্রথমেই k এর প্যারেন্ট ধরে ধরে উপরে উঠে সবগুলো অ্যানসেস্টরের লিস্ট করে ফেলি:

k এর অ্যানসেস্টর:

j	i	a

n এর অ্যানসেস্টর:

m	i	a

দুইটা লিস্টেই বাম থেকে ডানে যেতে থাকলে প্রথম যে দুইটা নোড কমন পাবো সেটাই হবে এল.সি.এ! এই উদাহরণে প্রথমে কমন পাবো i, তাই সেটাই হবে এল.সি.এ। ‘a’ ও একটা কমন এনসেস্টর কিন্তু সেটা “লোয়েস্ট” না। একটা ভিজিটেড অ্যারে বা ফ্ল্যাগ ব্যবহার করে খুব সহজেই আমরা এই অ্যালগোরিদম ইমপ্লিমেন্ট করতে পারি। প্রথমে k এর লিস্টে যারা আছে সেগুলোর ইনডেক্সের ফ্ল্যাগ অন করে দিবো, তারপর n এর লিস্টে লুপ চালিয়ে দেখবো ফ্ল্যাগ অন আছে কোন নোডের।

প্রতিটা লিস্ট বানাতে আমাদের সর্বোচ্চ n টা নোড ভিজিট করা লাগতে পারে, তাই কমপ্লেক্সিটি O(n) ।

O(n) কমপ্লিক্সিটি খারাপ না যদি আমাদের মাত্র এক জোড়া নোডের এল.সি.এ বের করতে হয়। যদি আমাদের m জোড়া নোড দিয়ে বলে প্রতিটা পেয়ারের এল.সি.এ বের করতে তাহলে কমপ্লেক্সিটি হয়ে যাবে O(m*n) । এটাকে আমরা চাইলে O(m*log(n)) এ সলভ করতে পারি, অর্থাৎ প্রতি কুয়েরি কাজ করবে O(logn) এ। তবে তার আগে কিছু প্রি-প্রসেসিং করে নিতে হবে।

প্রতিটা নোডের জন্য আমরা সেভ করে রাখবো নোডটার:

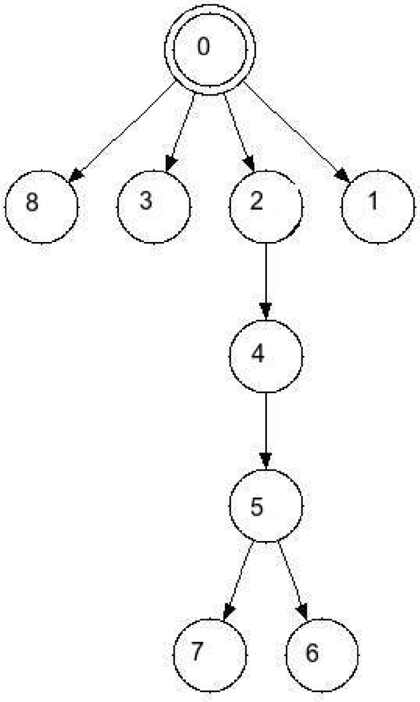
> *“প্রথম প্যারেন্ট বা ২^০ তম প্যারেন্ট*
>
> *২য় প্যারেন্ট বা ২^১ তম প্যারেন্ট*

top

> *৪র্থ প্যারেন্ট বা ২^২ তম প্যারেন্ট*

> *৮ম প্যারেন্ট বা ২^৩ তম প্যারেন্ট*

যদি K-তম প্যারেন্ট বলতে কিছু না থাকে তাহলে আমরা -১ রেখে দিবো। যেমন রুট নোডের প্রথম প্যারেন্ট বলতে কিছু নেই, তাই প্রথম প্যারেন্ট হবে -১।



তাহলে উপরের ট্রি এর জন্য নিচের মতো একটা ২-ডিমেনশনাল টেবিল তৈরি হবে:

নোড	২^০ তম প্যারেন্ট	২^১ তম প্যারেন্ট	২^২ তম প্যারেন্ট
০	-১	-১	-১
১	০	-১	-১
২	০	-১	-১

৩	০	-১	-১
৪	২	০	-১
৫	৪	২	-১
৬	৫	৪	০
৭	৫	৪	০
৮	০	-১	-১

তাহলে উপরের ট্রি এর জন্য নিচের মতো একটা ২-ডিমেনশনাল টেবিল তৈরি হবে:

এই টেবিলটাকে বলে **স্পার্স টেবিল**। এই টেবিলটা তৈরি করে কি লাভ হলো? এখন তুমি কোনো নোডের k তম প্যারেন্ট খুব সহজেই খুজে বের করতে পারবে। যেমন কারো ২৫ তম প্যারেন্ট দরকার হলে প্রথমে ২৫ এর নিচে ২ এর সবথেকে বড় পাওয়ার ২^৪=১৬ তম প্যারেন্ট খুজে বের করবে। তারপর ১৬তম প্যারেন্টের (২৫-১৬)=৯তম প্যারেন্ট খুজে বের করবে একই পদ্ধতিতে! প্রতিটা সংখ্যাকে ২ এর পাওয়ারের যোগফল হিসাবে প্রকাশ করা যায় সেটারই সুবিধা নিচ্ছি আমরা। এভাবে করে O(logn) কমপ্লেক্সিটিতে আমরা কারো k তম প্যারেন্ট বের করতে পারবো যেটা পরবর্তীতে LCA বের করতে কাজে লাগবে।

এখন কথা হলো উপরের টেবিলটা বানাবো কিভাবে? ২^০ তম প্যারেন্ট বের করা সহজ, ট্রি এর উপর একটা ডেপথ-ফার্স্ট-সার্চ বা ব্রেথড-ফার্স্ট-সার্চ চালিয়ে একটা অ্যারেতে সেভ করে রাখবো কার প্যারেন্ট কে। মনে করি অ্যারেটা হলো T। তাহলে উপরের ট্রি এর জন্য পাবো T[০]=-১, T[১]=০,T[৬]=৫ ইত্যাদি।

এবার আমরা কলাম-বাই-কলাম টেবিলটা ভরাট করবো। প্রথমেই প্রথম কলাম ভরাট করবো যেটা আসলে T অ্যারের সমান।

টেবিলটা নাম P হলে প্রথম কলাম ভরাট করবো এভাবে,

```
for (i = 0; i < N; i++)
        P[i][0] = T[i];
```

এখন চিন্তা করে দেখো একটা নোডের ২^৪=১৬ তম প্যারেন্ট হলো নোডটার ২^৩=৮ম প্যারেন্টের ২^৩=৮ম প্যারেন্ট। তাহলে কোন নোডের 2^j তম প্যারেন্ট হবে নোডটার 2^(j-1) তম প্যারেন্টের 2^(j-1) তম প্যারেন্ট! তাহলে

top

যেকোন নোড i এর জন্য:

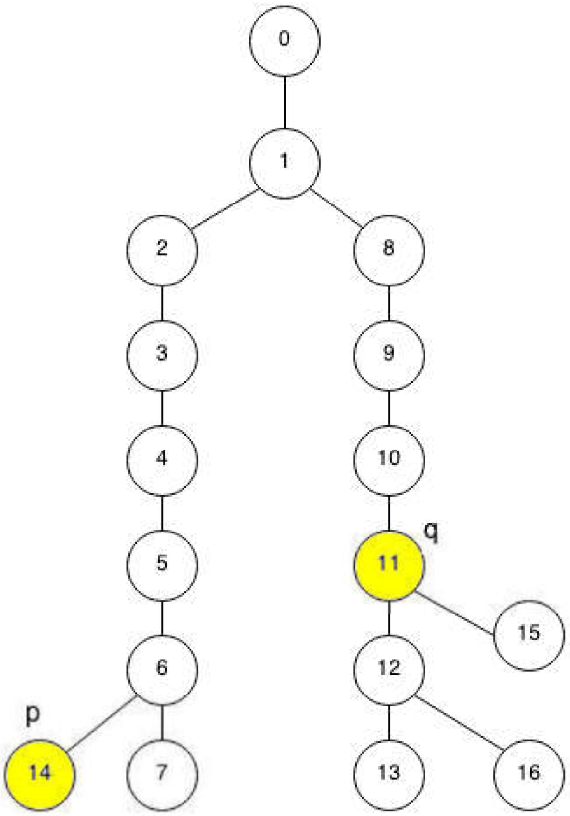
> *P[i][j]=P[P[i][j - 1]][j - 1] যেখানে P[i][j - 1] হলো j-1 তম প্যারেন্ট।*

তাহলে পুরো টেবিলটা ভরাট করে ফেলতে পারি এভাবে:

```
for (j = 1; (1 << j) < N; j++)
        for (i = 0; i < N; i++)
              if (P[i][j - 1] != -1)
                P[i][j] = P[P[i][j - 1]][j - 1];
```

2^j এর মান যদি N এর ছোট না হয় তাহলে নতুন কোনো প্যারেন্ট পাবার সম্ভাবনা নেই, তাই লুপ চলবে (1<<j) বা 2^j যতক্ষণ N এর ছোট হবে।

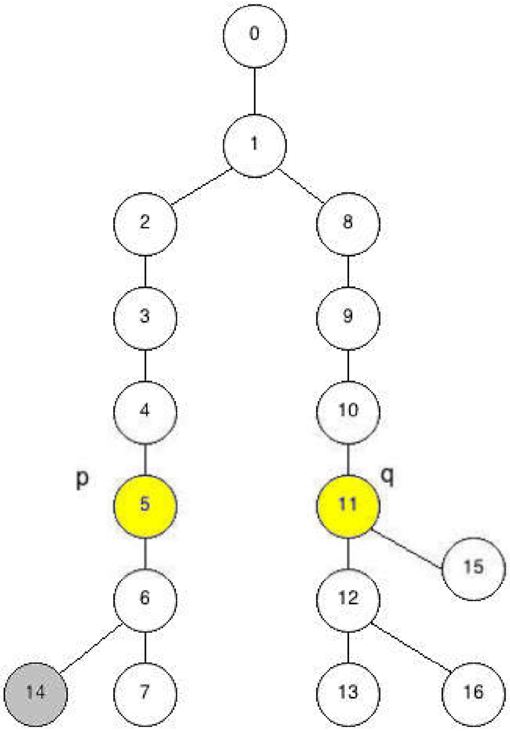
এখন আমরা দুটি নোডের এল.সি.এ বের করবো। আমাদের আরেকটা অতিরিক্ত অ্যারে লাগবে **L[]** যেখানে থাকবে প্রতিটা নোডের ডেপথ বা লেভেল। **রুট নোডের লেভেল হবে ০।**



এখন আমরা হলুদ নোড দুইটার এল.সি.এ বের করতে চাই:

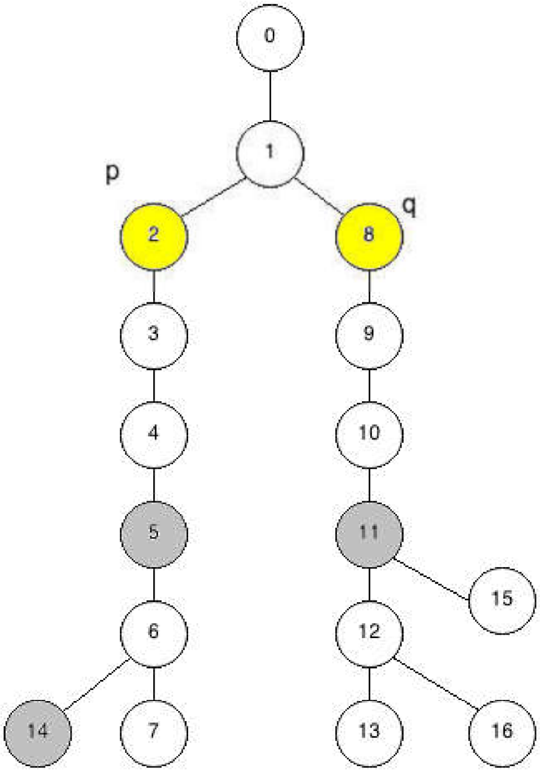
প্রথম কাজ হবে এদেরকে একই লেভেলে নিয়ে আসা। p এর লেভেল থেকে ২ এর পাওয়ার বিয়োগ করতে থাকবো এবং একই সাথে স্পার্স টেবিল ব্যবহার করে উপরে উঠাতে থাকবো যতক্ষণনা q এর লেভেলের সমান হয়।

top



```
for (i = log; i >= 0; i--)
        if (L[p] - (1 << i) >= L[q]) //2^i তম প্যারেন্টে যাও
            p = P[p][i];
```

এখন দুইজন সমান লেভেলে আসলো। এবার কাজ হবে দুইটা নোডকেও টেনে একসাথে উপরে উঠানো যতক্ষনা নোড দুটির প্যারেন্ট একসাথে এসে মিলে:



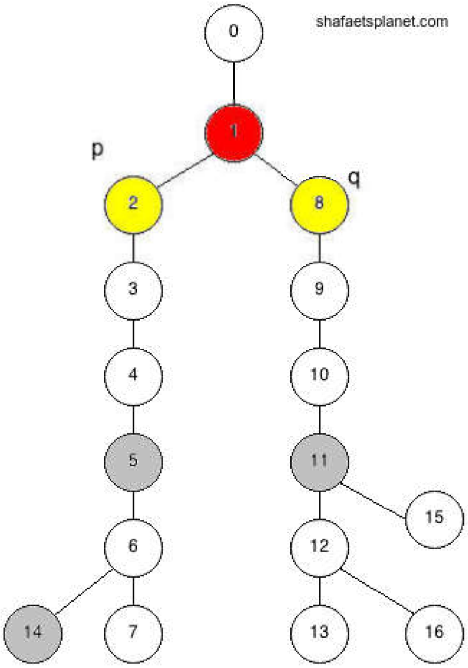
top

কোডটা হবে এরকম:

```
for (i = log; i >= 0; i--)
        if (P[p][i] != -1 && P[p][i] != P[q][i])
            p = P[p][i], q = P[q][i];
```

আমরা এমন জায়গায় p,q কে আনছি যেখানে তাদের দুইজনেরই প্যারেন্ট একই। তাহলে এখন p বা q এর প্যারেন্টই হবে এল.সি.এ।

```
return T[p];
```



আমরা যদি লুপের মধ্যেই ২ আর ৮ নাম্বার নোডকে p,q তে না এনে ডিরেক্ট ১ নম্বর নোডে আনার চেষ্টা করতাম তাহলে কি ঘটতে পারতো? P[p][i] != P[q][i] এই শর্তটা না থাকলে কি হত? এটা চিন্তা করা তোমার কাজ 🙂।

কমপ্লেক্সিটি:

স্পার্স টেবিলে রো থাকবে n টা, প্রতিটা রো তে logn টা কলাম থাকবে, প্রি-প্রোসেসিং কমপ্লেক্সিটি O(nlogn)।

প্রতি কুয়েরিতে কমপ্লেক্সিটি O(logn)

রিলেটেড প্রবলেম:

১. একটা ওয়েটেড ট্রি দেয়া আছে। দুটি নোড দিয়ে বলা হলো তাদের মধ্যের দূরত্ব বের করতে হবে। (Spoj QTree)

top

২. একটা ওয়েটেড ট্রি দেয়া আছে। দুটি নোড দিয়ে বলা হলো তাদের মধ্যকার পাথের সর্বোচ্চ ওয়েটের এজ এর মান বলতে হবে। (LOJ A Secret Mission)

সলিউশন ১: শুরুতে ডিএফস চালিয়ে রুট থেকে প্রতিটা নোডের দূরত্ব সেভ করে রাখো। এখন নোড দুইটা a,b এবং তাদের কমন অ্যানসেস্টর L হলে সলিউশন হবে dist(root,a)+dist(root,b)-2*dist(root,L)।

সলিউশন ২: স্পার্স টেবিলে অতিরিক্ত একটা ইনফরমেশন রাখতে হবে। প্রতিটা নোড থেকে 2^0, 2^1, 2^2 ইত্যাদি তম প্যারেন্টের জন্য, প্যারেন্ট থেকে ওই নোডের মধ্যকার পাথের সর্বোচ্চ ওয়েটের এজের মান সেভ করে রাখতে হবে। বিস্তারিত লিখলাম, এটা সলভ করা তোমার কাজ।

আরো কিছু প্রবলেম:

TJU Closest Common Ancestors
UVA Flea Circus
LightOJ LCA Category

রিসোর্স:

এল.সি.এ বের করার আরো কিছু পদ্ধতি আছে যেগুলো পাবে এখানে। এগুলো ছাড়াও টারজানের একটা অফলাইন অ্যালগোরিদম আছে যেটা আগে সব কুয়েরি ইনপুট নিয়ে একটি ডিএফস চালিয়ে সবগুলো পেয়ারের এল.সি.এ বের করে দেয়।

সম্পূর্ণ কোড:

ভেক্টর g তে ট্রি সেভ করতে হবে। প্রথমে dfs ফাংশন কল করে লেভেল,প্যারেন্ট ফিক্সড করে তারপর lca_init কল করে প্রিপ্রসেসিং করতে হবে।

```
//LCA using sparse table
//Complexity: O(NlgN,lgN)
#define mx 100002
int L[mx]; //লেভেল
int P[mx][22]; //স্পার্স টেবিল
int T[mx]; //প্যারেন্ট
vector<int>g[mx];
void dfs(int from,int u,int dep)
{
    T[u]=from;
    L[u]=dep;
    for(int i=0;i<(int)g[u].size();i++)
    {
        int v=g[u][i];
        if(v==from) continue;
        dfs(u,v,dep+1);
```

top

```
    }
}

int lca_query(int N, int p, int q) //N=নোড সংখ্যা
  {
      int tmp, log, i;

      if (L[p] < L[q])
          tmp = p, p = q, q = tmp;

        log=1;
      while(1) {
        int next=log+1;
        if((1<<next)>L[p])break;
        log++;

      }

        for (i = log; i >= 0; i--)
          if (L[p] - (1 << i) >= L[q])
              p = P[p][i];

      if (p == q)
          return p;

        for (i = log; i >= 0; i--)
          if (P[p][i] != -1 && P[p][i] != P[q][i])
              p = P[p][i], q = P[q][i];

      return T[p];
  }

void lca_init(int N)
  {
      memset (P,-1,sizeof(P)); //শুরুতে সবগুলো ঘরে -১ থাকবে
      int i, j;
       for (i = 0; i < N; i++)
          P[i][0] = T[i];

      for (j = 1; 1 << j < N; j++)
         for (i = 0; i < N; i++)
             if (P[i][j - 1] != -1)
                 P[i][j] = P[P[i][j - 1]][j - 1];
 }

```

top

```
int main(void) {
        g[0].pb(1);
        g[0].pb(2);
        g[2].pb(3);
        g[2].pb(4);
        dfs(0, 0, 0);
        lca_init(5);
        printf( "%d\n", lca_query(5,3,4) );
        return 0;
}
```

হ্যাপি কোডিং!