

# শাফায়েতের ব্লগ

প্রোগ্রামিং ও অ্যালগরিদম টিউটোরিয়াল

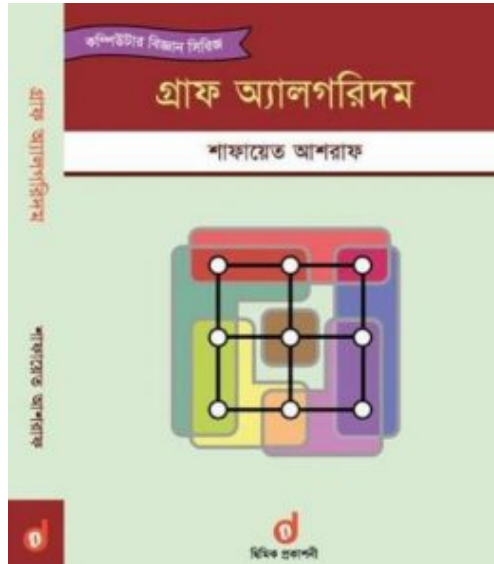
Home

অ্যালগরিদম নিয়ে যত লেখা!

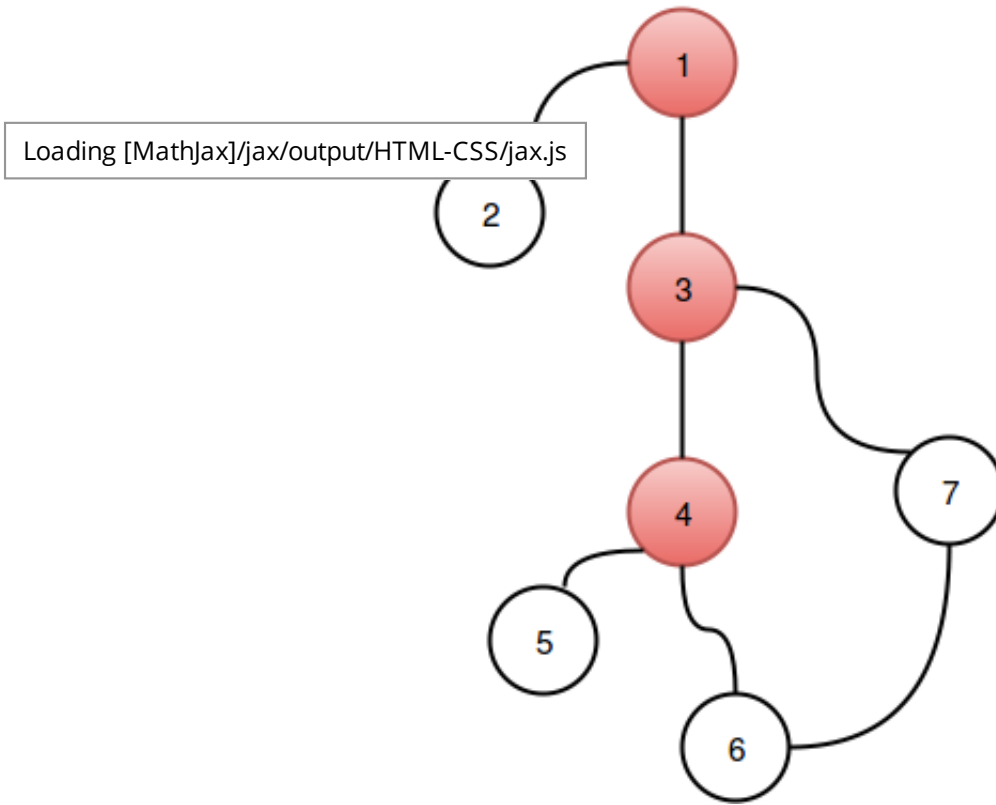
আমার সম্পর্কে...

## গ্রাফ থিওরিতে হাতেখড়ি ১৩: আর্টিকুলেশন পয়েন্ট এবং ব্রিজ

📅 নভেম্বর ২৮, ২০১৫ by শাফায়েত



আর্টিকুলেশন পয়েন্ট হলো আনডিরেক্টেড গ্রাফের এমন একটা নোড যেটা গ্রাফ থেকে মুছে ফেললে বাকি গ্রাফটুকু একাধিক কম্পোনেন্ট এ ভাগ হয়ে যায়।



উপরের ছবিতে ১, ৩ অথবা ৪ নম্বর নোড এবং সেই নোডের অ্যাডজেসেন্ট এজগুলোকে মুছে দিলে গ্রাফটা একাধিক ভাগ হয়ে যাবে, তাই ১, ৩ ও ৪ হলো এই গ্রাফের আর্টিকুলেশন পয়েন্ট। আর্টিকুলেশন পয়েন্টকে অনেকে কাট-নোড(cut node), আর্টিকুলেশন নোড বা ক্রিটিকাল পয়েন্ট (critical point) ও বলে।

আর্টিকুলেশন পয়েন্ট বের করার একটা খুব সহজ উপায় হলো, ১টা করে নোড গ্রাফ থেকে মুছে দিয়ে দেখা যে গ্রাফটি একাধিক কম্পোনেন্ট এ বিভক্ত হয়ে গিয়েছে নাকি।

```

1 procedure articulationPointNaive(G):
2   articulation_points = []
3   for all nodes u in G
4     G.removeNode(u)
5     if get_number_of_component(G) > 1
6       articulation_points.add(u)
7     end if
8     G.addBackNode(u)
9   end for
10  return articulation_points

```

কম্পোনেন্ট সংখ্যা ডিএফএস বা বিএফএস দিয়ে খুব সহজে বের করা যায়। এই পদ্ধতিতে  $V$  বার ডিএফএস চালাতে হবে যেখানে  $V$  হলো নোড সংখ্যা, মোট কমপ্লেক্সিটি  $O(V \times (V + E))$  বা  $O(V^3)$  কারণ সর্বোচ্চ এজ সংখ্যা  $V^2$ ।

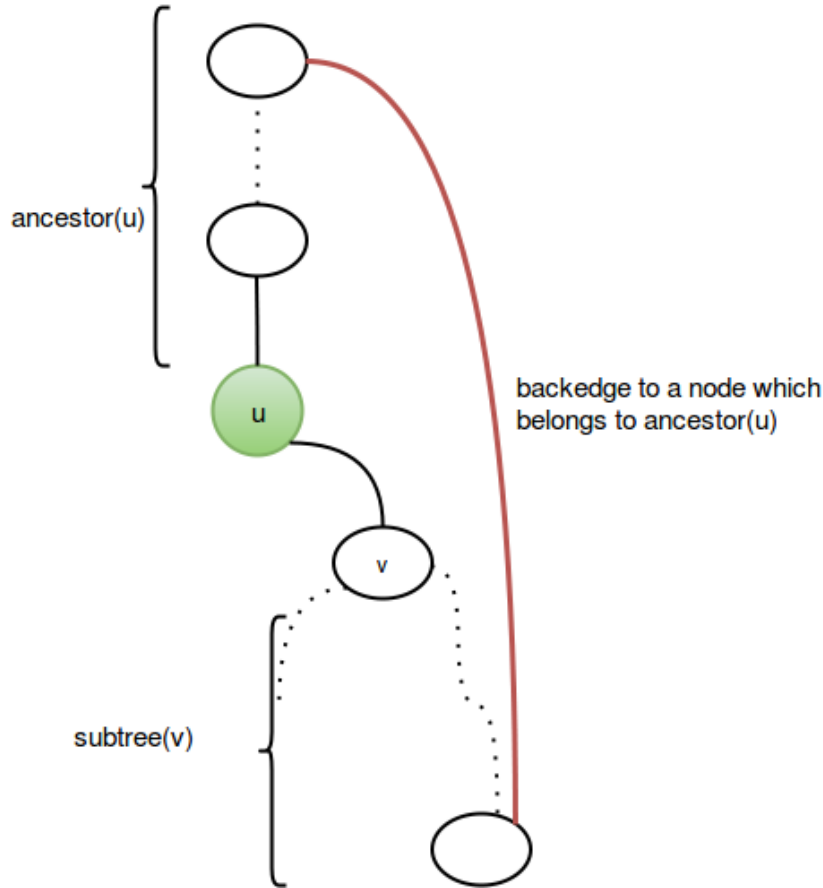
এখন আমরা একবার মাত্র ডিএফএস চালিয়ে আর্টিকুলেশন পয়েন্ট বের করবো। এই অ্যালগোরিদম শেখার জন্য ডিএফএস এর ডিসকভারি/ফিনিশিং টাইম এবং ট্রি এজ ও ব্যাক এজ নিয়ে ধারণা থাকতে হবে।

একটা গ্রাফে ডিএফএস চালালে যেসব ট্রি এজ পাওয়া যায় সেগুলো নিয়ে তৈরি হয়ে ডিএফএস ট্রি।

দুটি কেস থাকতে পারে। যদি একটা নোড ট্রি এর রুট হয় তাহলে একভাবে কাজ করবো, রুট না হলে আরেকভাবে কাজ করবো।

একটা নোড  $u$  যদি ট্রি এর রুট হয় এবং ডিএফএস ট্রি তে নোডটার একাধিক চাইল্ড নোড থাকে তাহলে নোডটা আর্টিকুলেশন পয়েন্ট।

রুট ছাড়া বাকি নোডের জন্য কাজটা একটু জটিল।



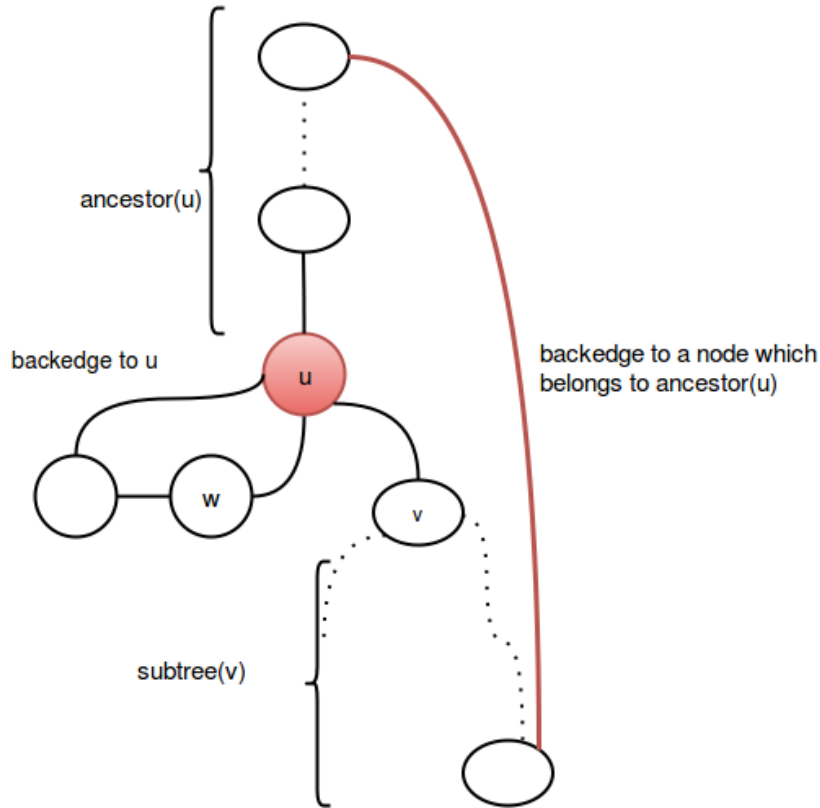
ডিএফএস ট্রি এর একটা এজ  $u - v$  এর কথা চিন্তা করো। রুট থেকে  $u$  তে আসার পথে যেসব নোড ভিজিট করেছো তাদের আমরা বলবো  $ancestor(u)$ । এখন  $v$  যে সাবট্রি এর রুট সেই সাবট্রির সবগুলো নোডের সেটকে আমরা বলবো  $subtree(v)$ ।

এখন  $u$  একটা আর্টিকুলেশন পয়েন্ট হবে যদি মূল গ্রাফে  $u$  কে মুছে দিলে  $subtree(v)$  এর নোডগুলো একটা আলাদা কম্পোনেন্ট এ পরিণত হয়।  $subtree(v)$  আলাদা কম্পোনেন্ট এ পরিণত হবে যদি না মূল গ্রাফে সাবট্রি

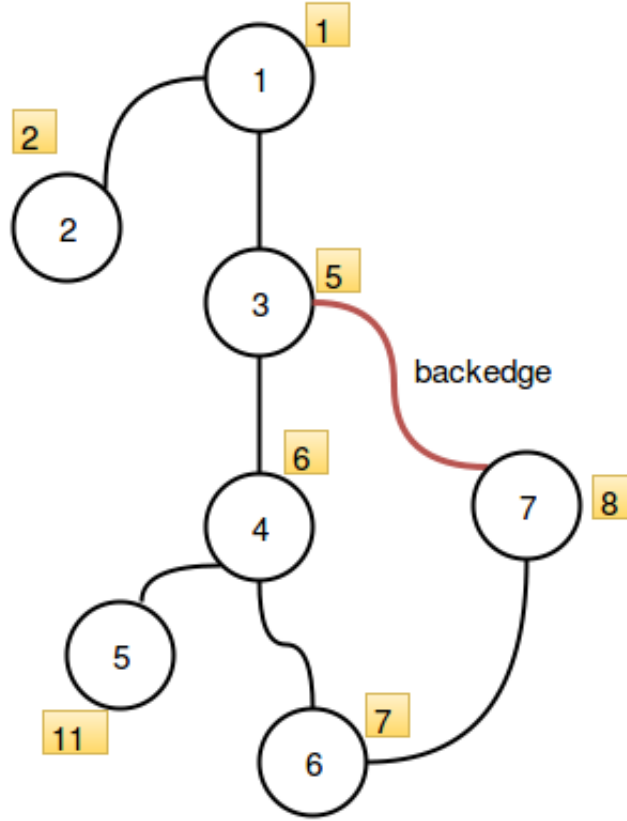
$subtree(v)$  এর কোনো নোড থেকে  $ancestor(u)$  তে একটা ব্যাকএজ থাকে। যদি ব্যাকএজ থাকে তাহলে নোড  $u$  এবং অ্যাডজেসেন্ট এজগুলো মুছে গেলেও  $ancestor(u)$  থেকে ব্যাকএজ দিয়ে  $subtree(v)$  তে পৌঁছানো যাচ্ছে, নতুন কম্পোনেন্ট তৈরি হচ্ছে না।

$u$  এর যেকোনো একটা চাইল্ড নোড  $v$  এর জন্য যদি  $subtree(v)$  থেকে  $ancestor(u)$  তে পৌঁছানো না যায়, তাহলে  $u$  আর্টিকুলেশন পয়েন্ট,  $u$  কে মুছে দিলে সেইসব  $subtree(v)$  নতুন কম্পোনেন্ট এ পরিণত হবে যাদের সাথে  $ancestor(u)$  এর কোনো ব্যাকএজ সংযোগ নেই।

নিচের ছবিতে  $subtree(v)$  যদিও ব্যাকএজ দিয়ে  $ancestor(u)$  এর সাথে সংযুক্ত,  $subtree(w)$  থেকে  $ancestor(u)$  তে ব্যাকএজ নেই। তাই  $u$  একটা আর্টিকুলেশন পয়েন্ট।



এবার প্রথম গ্রাফটায় ফিরে আসি। গ্রাফের নোডগুলো ১,২,৩,৪,৬,৭,৫ এই অর্ডারে ভিজিট করলে আমরা প্রতিটা নোডের যা ডিসকভারি টাইম পাবো সেটা পাশে ছোটো করে লেখা হয়েছে:



ডিসকভারি টাইম কিভাবে বের করতে হয় না বুঝলে **ডিএফএস নিয়ে টিউটোরিয়ালটা** দেখো।  $d[u]$  দিয়ে আমরা ডিসকভারি টাইম বুঝাবো।

গ্রাফের ব্যাকএজ টা লাল এজ দিয়ে দেখানো হয়েছে। বাকি কালো এজগুলো ডিএফএস ট্রি এর অংশ। 1 হলো রুট নোড।

ডিএফএস ট্রি তে রুট নোড 1 এর চাইল্ড সংখ্যা এখানে ২টা (২ এবং ৩)। তাই 1 একটা আর্টিকুলেশন পয়েন্ট।

লক্ষ্য করো নোড  $ancestor(u)$  এর যেকোনো নোডের ডিসকভারি টাইম  $d[u]$  এর থেকে ছোটো। আবার  $u$  এর অ্যাডজেসেন্ট যেকোনো এজ  $u - v$  এর জন্য  $subtree(v)$  এর সব নোডের ডিসকভারি টাইম  $d[u]$  এর থেকে বড়। এখন  $subtree(v)$  এর কোনো নোড থেকে যদি এমন একটা ব্যাকএজ  $v - w$  থাকে যেন  $d[w] < d[u]$  হয় তাহলে বুঝতে হবে তুমি  $u - v$  এজ পার হয়ে  $subtree(v)$  দিয়ে  $ancestor(u)$  তে পৌঁছে গেছো এবং  $w \in ancestor(u)$ । তারমানে  $u$  মুছে দিলেও  $subtree(v)$  থেকে  $w$  তে পৌঁছানো যাবে।

যেমন 4 নম্বর নোডের কথা চিন্তা করো। 4 এর ডিসকভারি টাইম  $d[4] = 6$  এবং  $ancestor(4) = \{1, 2, 3\}$ । এখন 4-6 এজটার কথা ভাবি।  $subtree(6)$  এ একটা ব্যাকএজ  $7 - 3$  আছে, এবং  $d[3] = 5$  যা  $d[4]$  এর থেকে ছোটো। তারমানে  $3 \in ancestor(4)$ । তাহলে তুমি 4 নোডটা মুছে দিলেও  $subtree(6)$  ব্যাকএজের মাধ্যমে  $ancestor(4)$  এর সাথে সংযুক্ত থাকবে।

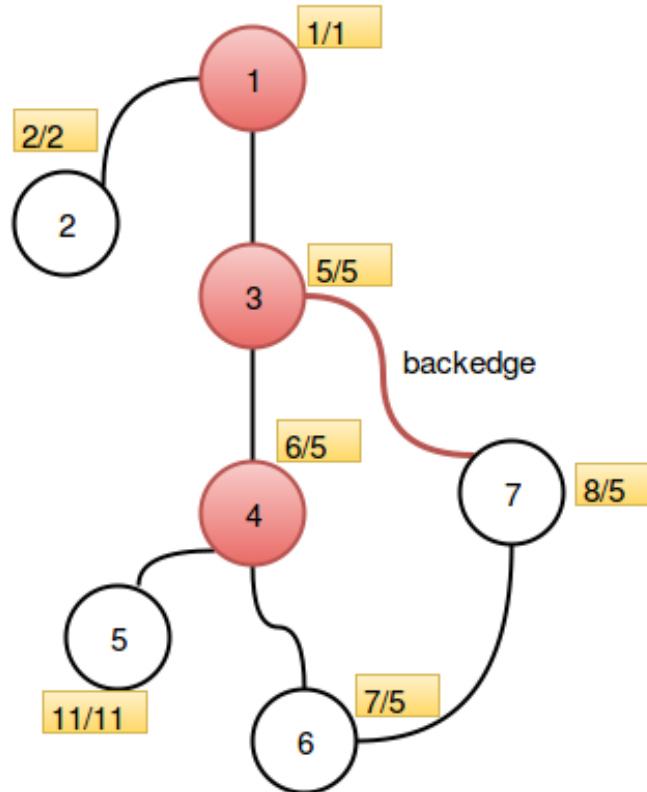
এবার আমরা আরেকটা ভ্যারিয়েবল ডিফাইন করবো  $low[u]$ । মনে করো  $subtree(u)$  এবং  $subtree(u)$  এর সাথে

ব্যাকএজ দিয়ে সংযুক্ত সবগুলো নোডের একটা সেট বানানো হলো, সেটা টা হলো  $\{x_1, x_2 \dots x_m\}$ । তাহলে  $low[u]$  হবে  $\min(d[x_1], d[x_2], \dots, d[x_m])$ ।

যেমন 4 নম্বর নোডের জন্য  $subtree(u) = \{5, 6, 7\}$  এবং  $subtree(u)$  এর সাথে ব্যাকএজ দিয়ে যুক্ত আছে নোড 3। তাহলে  $low[u] = \min(d[5], d[6], d[7], d[3]) = 5$ ।

এখন চিন্তা করো কোনো একটা এজ  $u - v$  এর জন্য  $d[u] > low[v]$  হবার অর্থ কি?  $d[u]$  এর থেকে ডিসকভারি টাইম ছোটো একমাত্র  $ancestor(u)$  সেটের নোডগুলোর।  $subtree(v)$  এর কোনো নোড ব্যাকএজ দিয়ে  $ancestor(u)$  এর সাথে যুক্ত, সেজন্য়  $low[v]$  এর মান  $d[u]$  এর থেকে কমে গিয়েছে। যদি  $d[u] \leq low[v]$  হয়, তাহলেই শুধুমাত্র  $u$  একটা আর্টিকুলেশন পয়েন্ট হবে।

আগের গ্রাফেই ডিসকভারি টাইমের পাশাপাশি  $low[u]$  এর মানগুলোও দেখি:



তাহলে আমরা আর্টিকুলেশন পয়েন্ট বের করার একটা অ্যালগোরিদম পেয়ে গিয়েছি। প্রতিটা নোডের জন্য  $d[u]$ ,  $low[u]$  বের করতে পারলেই কাজ শেষ।  $low[u]$  বের করা কঠিন কিছু না, সুডোকোড দেখলেই পরিষ্কার হবে:

```

    articulation_point[] ← false
    visited[] ← false
    low[] = d[u] ← 0
    time ← 0
1  Procedure FindArticulationPoint(G, u):
2  time ← time+1

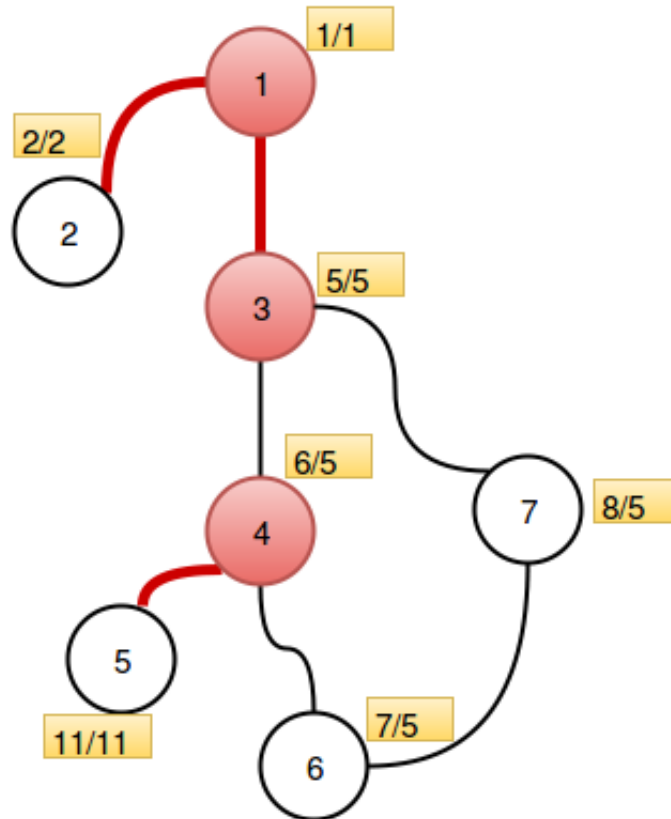
```

```

3  low[u]=d[u] ← time
4  visited[u] ← true
5  no_of_children ← 0
6  for each edge u to v in G.adjacentEdges(u) do
7      if(v == parent[u]) continue
8      if visited[v] //This is a backedge
9          low[u] = min(low[u], d[v])
10     end if
11     if not visited[v] //This is a tree edge
12         parent[u] = v
13         FindArticulationPoint(G, v)
14         low[u] = min(low[u], low[v])
15         if d[u] <= low[v] and u is not root:
16             articulation_point[u]=true
17         end if
18         no_of_children=no_of_children+1
19     end if
20     if(no_of_children>1 u is root):
21         articulation_point[u]=true
22     end if
23 end for

```

ব্রিজ জিনিসটা আর্টিকুলেশন পয়েন্টের মতই। গ্রাফ থেকে যে এজ তুলে দিলে গ্রাফটা একাধিক কম্পোনেন্ট ভাগ হয়ে যায় তাকেই বলা হয় ব্রিজ।



উপরের গ্রাফে 4-5, 1-2, আর 1-3 এই ৩টি এজ হলো ব্রিজ।

ব্রিজ আর আর্টিকুলেশন পয়েন্টের সুডোকোডের পার্থক্য খালি এক জায়গায় ১৫ নম্বর লাইনে  $d[u] \leq low[v]$  এর জায়গায়  $d[u] < low[v]$  লিখতে হবে। এটা কেন কাজ করে তুমি সহজেই বুঝতে পারবে যদি তুমি

সুডোকোডটা বুঝে থাকো, তাই আর ব্যাখ্যা করলাম না।

দুটি নোডের মধ্যে একাধিক এজ থাকলে অবশ্য এটা কাজ করবে না। তখন কি করতে হবে সেটা চিন্তা করা তোমার কাজ!

সলভ করার জন্য কিছু প্রবলেম পাবে **এখানে**।

হ্যাপি কোডিং!

ফেসবুকে মন্তব্য

0 comments

Powered by **Facebook Comments**



Posted in অ্যালগোরিদম/প্রবলেম সলভিং, প্রোগ্রামিং ? Tagged আর্টিকুলেশন পয়েন্ট, গ্রাফ, গ্রাফ থিওরি, ব্রিজ

5,667 বার পড়া হয়েছে

◀ ডাটা স্ট্রাকচার : স্ট্যাক

গ্রাফ থিওরিতে হাতেখড়ি ১৪ - স্ট্রংলি কানেক্টেড কম্পোনেন্ট ▶

12 thoughts on "গ্রাফ থিওরিতে হাতেখড়ি ১৩: আর্টিকুলেশন পয়েন্ট এবং ব্রিজ"



**Sajal**

নভেম্বর ২৯, ২০১৫ at ১০:১৪ am

ভাইয়া ৪-৫ ব্রিজ হলে ১-২ ব্রিজ হোলোনা কেনো?

Reply



**শাফায়েত**

নভেম্বর ২৯, ২০১৫ at ৭:০২ pm

ওটা ভুল ছিল, ঠিক করে দিয়েছি।

Reply

**Nashir**

নভেম্বর ৩০, ২০১৫ at ১০:৩১ pm

১৪ নাম্বার লাইনের পর ১৫ নাম্বার লাইন তো সবসময়ই true হয় ... !

Reply

**শাফায়েত**

ডিসেম্বর ১, ২০১৫ at ৫:৩৬ pm

ভুল ছিলো, ঠিক করে দিয়েছি এখন, ধন্যবাদ।

Reply

**Nashir**

নভেম্বর ৩০, ২০১৫ at ১১:৩৯ pm

আমার মতে ১৫ নাম্বার লাইনের চেকটা  $d[u] \geq low[v]$  এরকম হবে 😞

Reply

**Mahim**

ডিসেম্বর ২, ২০১৫ at ৬:৪৭ am

১৫ নম্বর লাইনে বোধহয়  $parent[v]=u$  হবে

Reply

**Mahim**

ডিসেম্বর ৪, ২০১৫ at ২:১২ pm

If there is only one edge from the root, shouldn't that edge be a Bridge?

Reply

**শাফায়েত**

ডিসেম্বর ৬, ২০১৫ at ৪:২৭ pm

হ্যা। রুট এর নিয়মটা খালি আর্টিকুলেশন পয়েন্টের জন্য, ব্রিজের জন্য না।

Reply

**Tanmoy**

ডিসেম্বর ১৮, ২০১৫ at ৮:২৬ pm

Vhai at first thank u for your great work on this blog 😊 and I think there is a problem on line : 9 . because if we low array initialize with 0 then the minimum value will remain unchanged.

Reply

**শাফায়েত**

ডিসেম্বর ২০, ২০১৫ at ১২:২৪ am

We are initializing low[u] by time.  $\text{low}[u] = d[u] \leftarrow \text{time}$

Reply

**Arif**

জুন ১৪, ২০১৬ at ৭:৪৯ am

৩ নং নোডের ডিসকভারি টাইম ৪ হওয়ার কথা। যদি ৪ না হয় তাহলে যুক্তি অনুযায়ী ৫ নং নোডের ১২ হওয়া দরকার।

[Reply](#)**Sadman Sakib**

আগস্ট ২৬, ২০১৬ at ৪:০৮ pm

The Discovery times of the given graph is wrong. Correct discovery time should be following:

 $d[1] : 1$  $d[2] : 2$  $d[3] : 4$  $d[4] : 5$  $d[5] : 6$  $d[6] : 8$  $d[7] : 9$ [Reply](#)

## Leave a Reply

Connect with:

Powered by [OneAll Social Login](#)

Your email address will not be published. Required fields are marked \*

Comment

Name \*

Email \*

Website



-



= 6

Post Comment

phonetic

probhat

english

## সাবস্ক্রাইব

---

Powered by [OneAll Social Login](#)

## আমার সম্পর্কে

---

শাফায়েত, সফটওয়্যার ইঞ্জিনিয়ার @ **HACKERRANK** (বিস্তারিত...)



## প্রোগ্রামিং কনটেস্ট এবং অ্যালগোরিদম

### অনুপ্রেরণা(৩):

কেন আমি প্রোগ্রামিং শিখবো?

কম্পিউটার বিজ্ঞান কেন পড়বো?

প্রোগ্রামিং কনটেস্ট এবং অনলাইন জাজে হাতেখড়ি

### অ্যালগোরিদম বেসিক(৬):

বিগ "O" নোটেশন

কমপ্লেক্সিটি ক্লাস(P-NP, টুরিং মেশিন ইত্যাদি)

হাল্টিং প্রবলেম(নতুন)

বাইনারি সার্চ - ১

বাইনারি সার্চ - ২(বাইসেকশন)

ফ্লয়েড সাইকেল ফাইন্ডিং অ্যালগোরিদম

### ডাটা স্ট্রাকচার(১১):

লিংকড লিস্ট

স্ট্যাক

কিউ+সার্কুলার কিউ(নতুন)

স্লাইডিং রেঞ্জ মিনিমাম কুয়েরি (ডিকিউ)

ডিসজয়েন্ট সেট(ইউনিয়ন ফাইন্ডিং)

ট্রাই(প্রিফিক্স ট্রি/রেডিক্স ট্রি)

সেগমেন্ট ট্রি-১

সেগমেন্ট ট্রি-২(লেজি প্রপাগেশন)

অ্যারে কমপ্রেসন/ম্যাপিং

লোয়েস্ট কমন অ্যানসেস্টর

বাইনারি ইনডেক্সড ট্রি

গ্রাফ থিওরি(১৮):

গ্রাফ থিওরিতে হাতেখড়ি

অ্যাডজেসেন্সি ম্যাট্রিক্স

অ্যাডজেসেন্সি লিস্ট

ব্রেথড ফার্স্ট সার্চ (বিএফএস)

মিনিমাম স্প্যানিং ট্রি ১ (প্রিমস অ্যালগোরিদম)

মিনিমাম স্প্যানিং ট্রি ২ (ক্রুসকাল অ্যালগোরিদম)

টপোলজিকাল সর্ট

ডেপথ ফার্স্ট সার্চ এবং আবারো টপোলজিকাল সর্ট

ডায়াক্সট্রা

ফ্লয়েড ওয়ার্শল

বেলম্যান ফোর্ড

আর্টিকুলেশন পয়েন্ট এবং ব্রিজ

স্ট্রংলি কানেক্টেড কম্পোনেন্ট

ম্যাক্সিমাম ফ্লো-১

ম্যাক্সিমাম ফ্লো-২

স্টেবল ম্যারেজ প্রবলেম

মিনিমাম ভারটেক্স কভার

ট্রি এর ডায়ামিটার নির্ণয়

লংগেস্ট পাথ প্রবলেম(নতুন)

অ্যালগোরিদম গেম থিওরি(৩):

গেম থিওরি-১

গেম থিওরি-২

গেম থিওরি-৩

## ডাইনামিক প্রোগ্রামিং(৮):

শুরুর কথা

ডিপি 'স্টেট', NcR, o-১ ন্যাপস্যাক

কয়েন চেঞ্জ, রক ক্লাইম্বিং

ডিপি সলিউশন প্রিন্ট করা এবং LIS

বিটমাস্ক ডিপি

মিনিমাম ভারটেক্স কভার(গ্রাফ+ডিপি)

লংগেস্ট কমন্ সাবসিকোয়েন্স(LCS)

ম্যাট্রিক্স চেইন মাল্টিপ্লিকেশন

## ব্যাকট্র্যাকিং(১):

ব্যাকট্র্যাকিং বেসিক এবং পারমুটেশন জেনারেটর

## নাম্বার থিওরি/গণিত(৪):

মডুলার অ্যারিথমেটিক

প্রাইম জেনারেটর (Sieve of Eratosthenes)

বিটওয়াইজ সিভ

ডিরেঞ্জমেন্ট

## স্ট্রিং ম্যাচিং(১):

রবিন-কার্প স্ট্রিং ম্যাচিং(নতুন)

## অন্যান্য(৩):

ডিরেকশন অ্যারে

মিট ইন দ্যা মিডল

কোয়ান্টাম কম্পিউটার(২)

কোয়ান্টাম কম্পিউটার কী?

কোয়ান্টাম কম্পিউটারের শক্তি এবং সীমাবদ্ধতা

