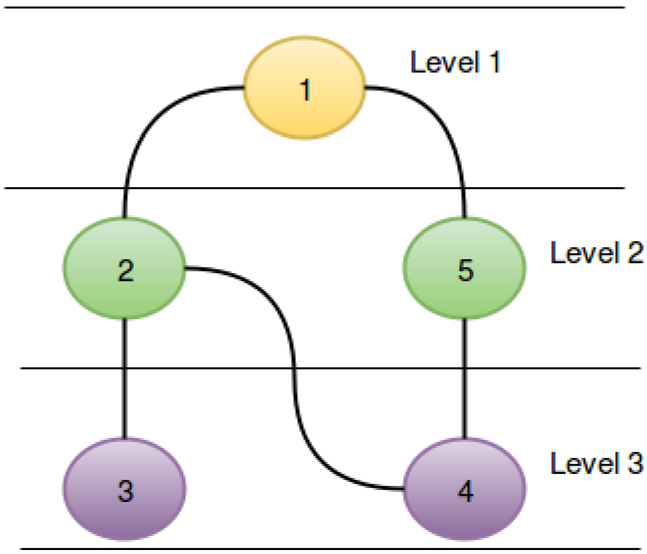
আগের পর্বগুলো পড়ে থাকলে হয়তো ডেপথ ফার্স্ট সার্চ বা ডিএফএস এতদিনে নিজেই শিখে ফেলেছো। তারপরেও এই টিউটোরিয়ালটি পড়া দরকার কিছু কনসেপ্ট জানতে।

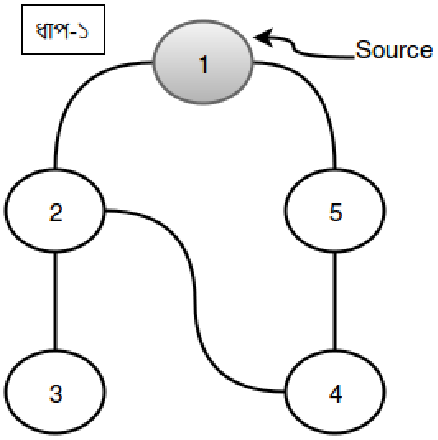
বিএফএস এ আমরা গ্রাফটাকে লেভেল বাই লেভেল সার্চ করেছিলাম,নিচের ছবির মতো করে:

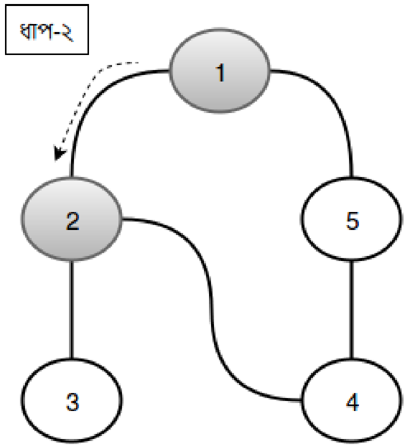


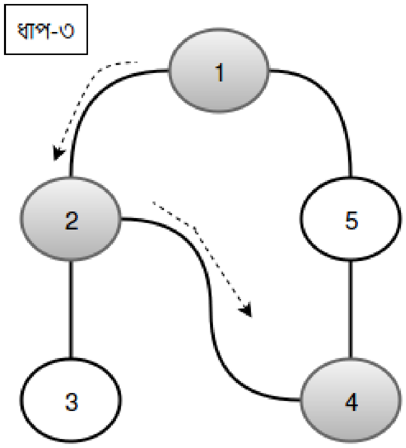
এবার আমরা কোনো নোড পেলে সাথে সাথে সে নোড থেকে আরো গভীরে চলে যেতে থাকবো,যখন আর গভীরে যাওয়া যাবেনা তখন আবার আগের নোডে ফিরে এসে অন্য আরেক দিকে যেত চেষ্টা করবো,এক নোড কখনো ২বার ভিজিট করবোনা। আমরা নোডের ৩টি রং(কালার) দিবো:

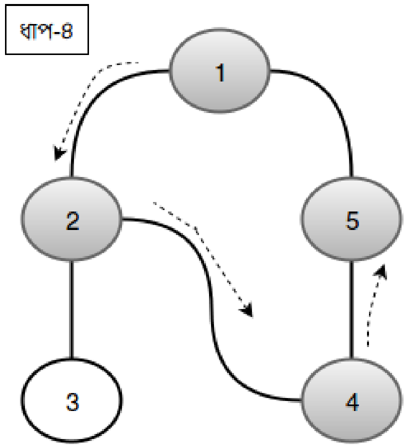
সাদা নোড= যে নোড এখনো খুজে পাইনি/ভিজিট করিনি।
গ্রে বা ধুসর নোড= যে নোড ভিজিট করেছি কিন্তু নোডটি থেকে যেসব চাইল্ড নোডে যাওয়া যায় সেগুলো এখনো ভিজিট করে শেষ করিনি, অর্থাত নোডটিকে নিয়ে কাজ চলছে।
কালো নোড= যে নোডের কাজ সম্পূর্ণ শেষ।

এবার আমরা ধাপগুলো দেখতে পারি:

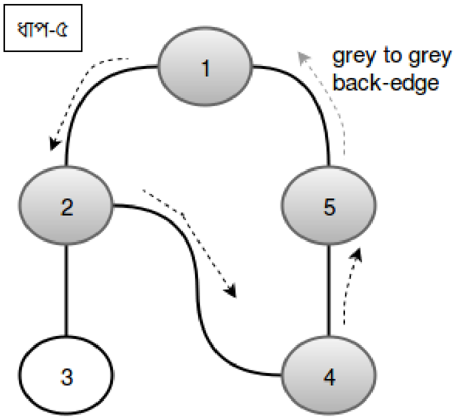

ধাপ-১
1
Source
2
5
3
4


ধাপ-২
1
2
5
3
4

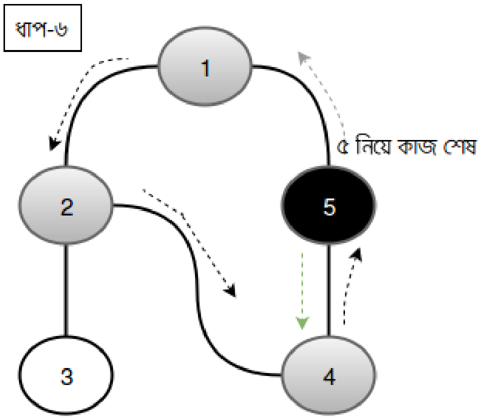

ধাপ-৩
1
2
5
3
4


ধাপ-৪
1
2
5
3
4

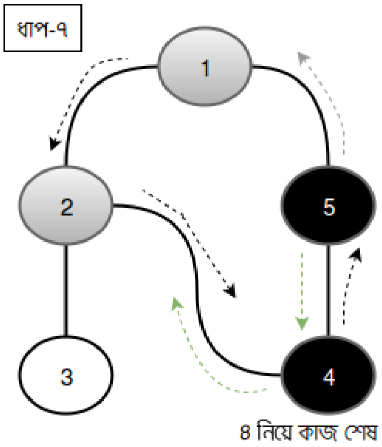
ধাপ-৫



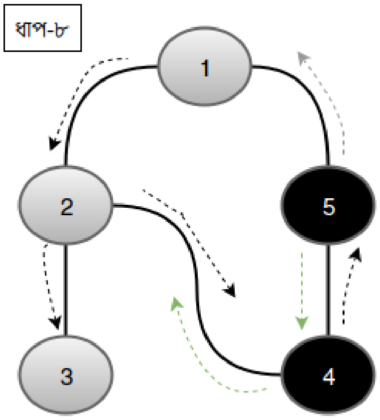
ধাপ-৬



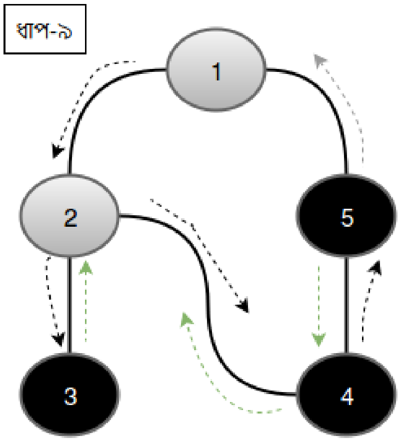
ধাপ-৭



ধাপ-৮

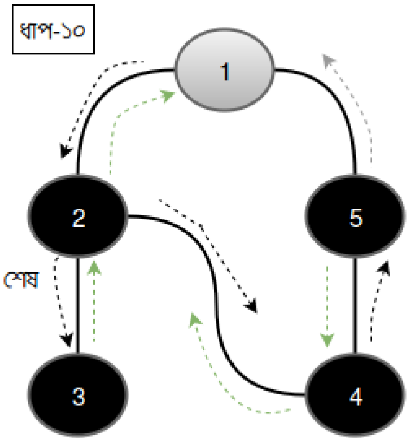


ধাপ-৯



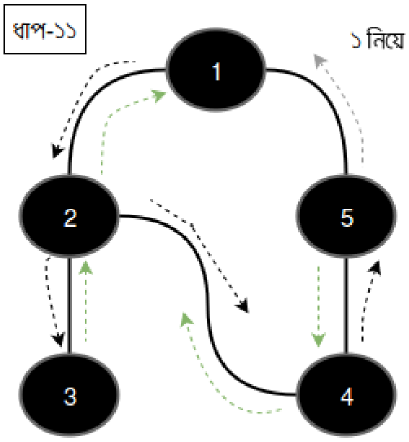
৩ নিয়ে কাজ শেষ

ধাপ-১০



২ নিয়ে কাজ শেষ

ধাপ-১১

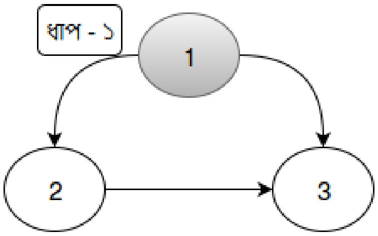


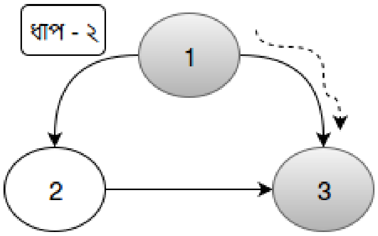
১ নিয়ে কাজ শেষ

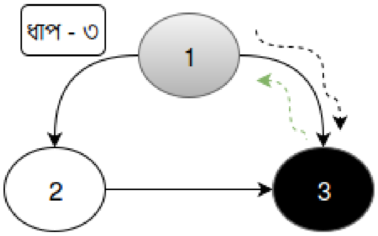
আশা করি ডিএফএস কিভাবে কাজ করে এটা পরিস্কার,খুব সহজ জিনিস এটা। এবার আমরা একটা খুব গুরুত্বপূর্ণ টার্ম শিখবো,সেটা হলো ব্যাকএজ(backedge)। লক্ষ করো ৫-১ কে ব্যাকএজ বলা হয়েছে। এর কারণ হলো তখনও ১ এর কাজ চলছে,৫ থেকে ১ এ যাওয়া মানে এমন একটা নোড ফিরে যাওয়া যাকে নিয়ে কাজ এখনো শেষ হয়নি,তারমানে অবশ্যই গ্রাফে একটি সাইকেল আছে। এ ধরনের এজকে ব্যাকএজ বলে,dfs এ **যদি কোনো সময় একটি গ্রে নোড থেকে আরেকটি গ্রে নোডে যেতে চেষ্টা করে তাহলে সে এজটি ব্যাকএজ এবং গ্রাফে অবশ্যই সাইকেল আছে**। dfs এর সোর্স নোড এবং নোড ভিজিট করার অর্ডার এর উপর নির্ভর করে সাইকেলে যে কোনো এজকে ব্যাকএজ হিসাবে পাওয়া যেতে পারে,যেমন ১ থেকে আগে ২ এ না গিয়ে ৫ এ গেলে পরে ২-১ কে ব্যাকএজ হিসাবে পাওয়া যেতো।

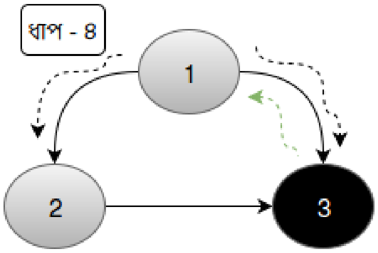
আর যখন আমরা স্বাভাবিক ভাবে গ্রে থেকে সাদা নোডে যাচ্ছি তখন সে এজগুলোকে বলা হয় **ট্রি এজ**। শুধুমাত্র ট্রি এজ গুলো রেখে বাকি এজগুলো মুছে দিলে যে গ্রাফটা থাকে তাকে বলা হয় **ডিএফএস ট্রি।**

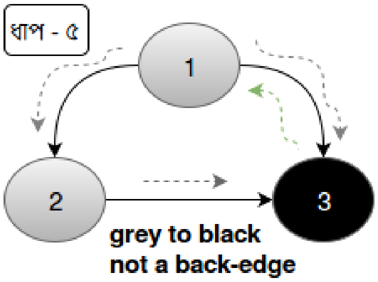
আনডিরেক্টেড গ্রাফের ক্ষেত্রে আগে ভিজিট করা কোনো নোডে ফিরে গেলেই সেটা ব্যাকএজ,কালার চেক না করলেও হয়। তবে ডিরেক্টেড গ্রাফের ক্ষেত্রে অবশ্যই করতে হবে। পরের ছবিটা দেখো:

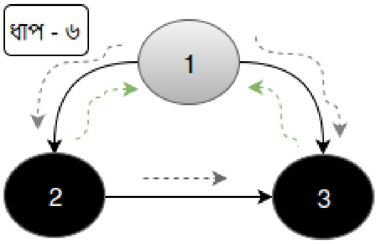


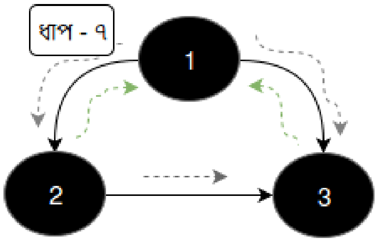












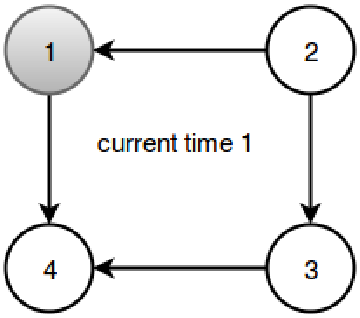
২-৩এর এজটাকে ব্যাকএজ বলা যাচ্ছেনা,কারণ ৩ এর কাজ আগেই শেষ হয়ে গেছে।

প্রতিটা নোড আর এজ নিয়ে একবার করে করছি, dfs এর কমপ্লেক্সিটি O(V+E)।

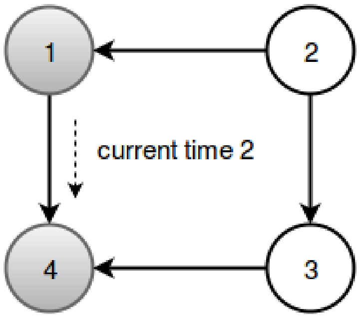
আমরা টপোলজিকাল সর্টের সমস্যা সমাধান করেছিলাম বারবার indegree উঠিয়ে। এবার আমরা খুব সহজে dfs দিয়ে এটা করবো। টপোলজিকাল কি সেটা না জানলে আগে এই পোস্টটা পড়ো,তারপর আগাও।
মনে করি আমাদের এজগুলো হলো: ২-১,২-৩,৩-৪,১-৪। অর্থাত ১ নম্বর কাজ করার আগে ২ নম্বরটি করতে হবে ইত্যাদি। এবার আমরা dfs চালানোর সময় একটি স্টপওয়াচ চালু করে দিবো। আর কোনো নোড নিয়ে কাজ শুরু করলে ঘড়ি দেখে নোডটি starting time/discovery time লিখে রাখবো,কাজ শেষ হলো নোডটির finishing time লিখে রাখবো।

d[]=discovery time
f[]=finishing time

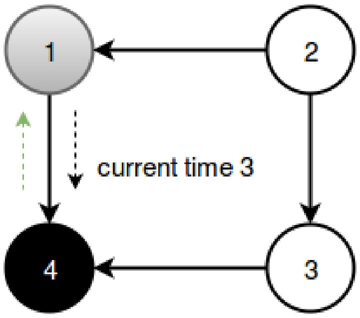
node	d[]	f[]
1	1	null
2	null	null
3	null	null
4	null	null



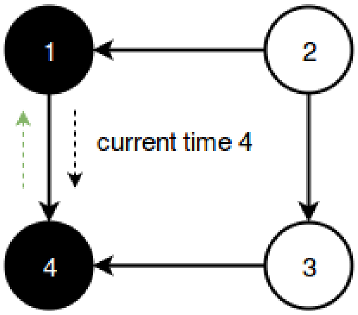
node	d[]	f[]
1	1	null
2	null	null
3	null	null
4	2	null



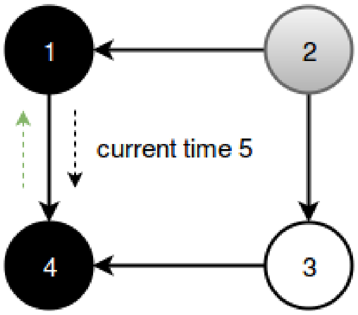
node	d[]	f[]
1	1	null
2	null	null
3	null	null
4	2	3



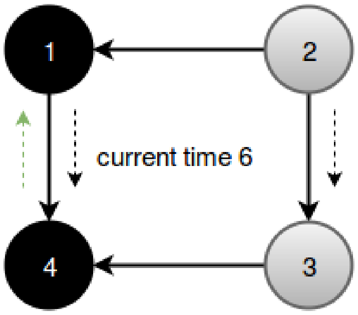
node	d[]	f[]
1	1	4
2	null	null
3	null	null
4	2	3

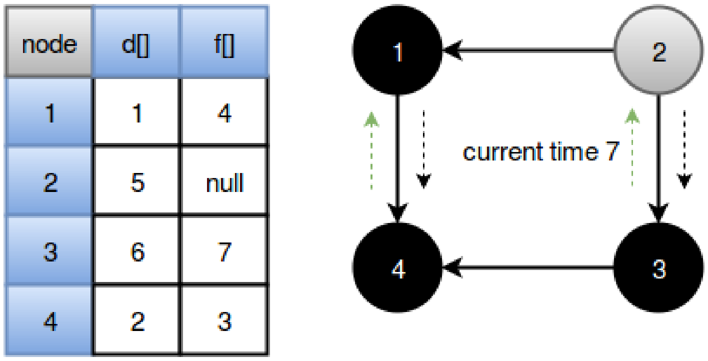


node	d[]	f[]
1	1	4
2	5	null
3	null	null
4	2	3

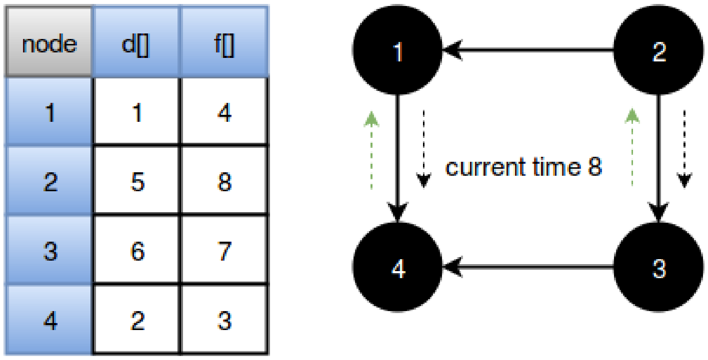


node	d[]	f[]
1	1	4
2	5	null
3	6	null
4	2	3





node	d[]	f[]
1	1	4
2	5	null
3	6	7
4	2	3



node	d[]	f[]
1	1	4
2	5	8
3	6	7
4	2	3

ফিনিশিং টাইম অনুযায়ি সর্ট করে পাই: ২,৩,১,৪

finishing time দেখে আমরা সহজেই টপসর্ট করতে পারি। যে নোডটি সবার আগে আসবে তার finishing time অবশ্যই সবথেকে বেশি হবে,কারণ প্রথম নোডের উপর নির্ভরশীল সব নোড ঘুরে আসার পরে সে নোডের finishing time assign করা হয়। uva 11504-dominos প্রবলেমে আগে নোডগুলোকে finishing time দিয়ে সর্ট করে তারপর আবার dfs চালাতে হয়,প্রবলেমটা চেষ্টা করো। ডিএফএস দিয়ে আমরা যেসব কাজ করি সেগুলোর অনেকগুলোই bfs দিয়ে করতে পারি। bfs এ সাধারণত টাইম কমপ্লেক্সিটি কম হয় তবে dfs কোডিং করতে খুব কম সময় লাগে। একটা সিম্পল dfs এর সুডোকোড এরকম:

```
procedure DFS(G, source):
  U ← source
  time  ← time+1
  d[u]  ← time
  color[u]  ← GREY
  for all edges from u to v in G.adjacentEdges(v) do
    if color[v] = WHITE
     DFS(G,v)
    end if
  end for
  color[u] ← BLACK
  time ← time+1
```

```
f[u] ← time
return
```

নিচের প্রবলেমগুলো সলভ করতে চেষ্টা করো:

http://uva.onlinejudge.org/external/2/280.html
http://uva.onlinejudge.org/external/115/11518.html
http://uva.onlinejudge.org/external/104/10452.html

এরপরে এই আর্টিকেলটা পড়ে ফেলো বিস্তারিত জানার জন্য,আমার লেখা পড়ে তুমি বেসিকটা শিখতে পারবে,বিস্তারিত জানতে এবং কঠিন প্রবলেম সলভ করতে আরো অনেক কিছু জানতে হবে।