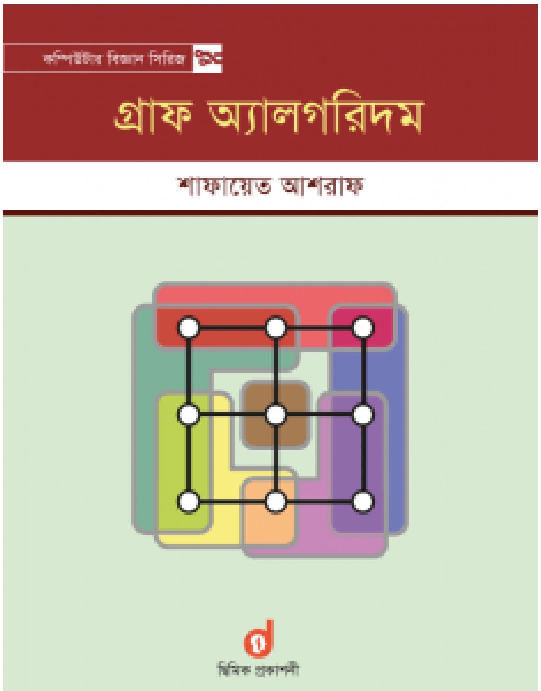
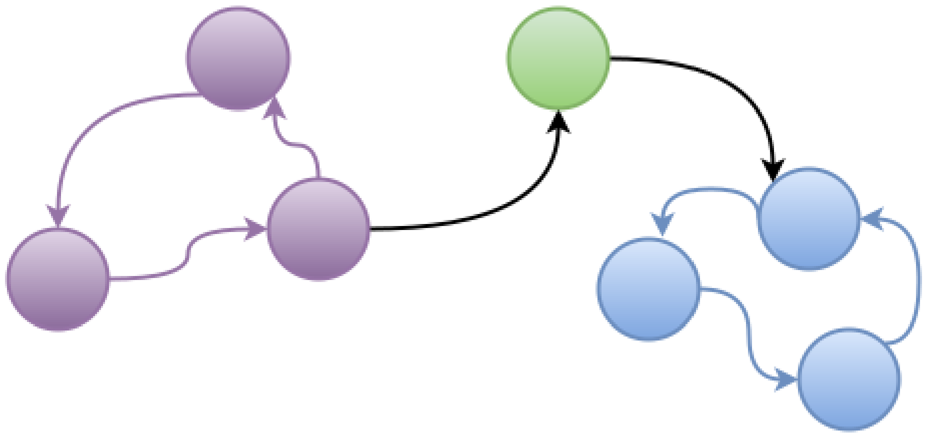
গ্রাফ থিওরিতে হাতেখড়ি ১৪ – স্ট্রংলি কানেক্টেড কম্পোনেন্ট

নভেম্বর ২৯, ২০১৫ by *শাফায়েত*



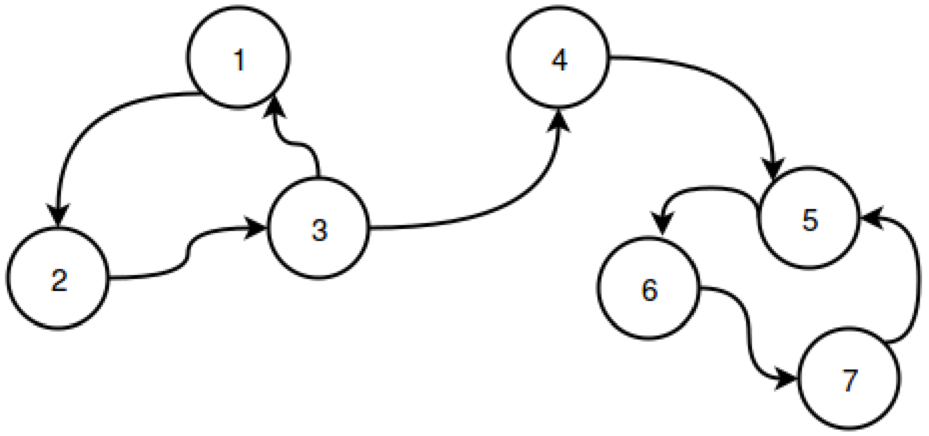
একটা ডিরেক্টেট গ্রাফের স্ট্রংলি কানেক্টেড কম্পোনেন্ট বা SCC হলো এমন একটা কম্পোনেন্ট যার প্রতিটা নোড থেকে অন্য নোডে যাবার পথ আছে। নিচের ছবিতে একটা গ্রাফের প্রতিটা স্ট্রংলি কানেক্টেড কম্পোনেন্ট ত রঙ দিয়ে দেখানো হয়েছে।

top



ডেপথ ফার্স্ট সার্চ এর ফিনিশিং টাইমের ধারণা ব্যবহার করে আমরা $O(V+E)$ তে একটা গ্রাফের স্ট্রংলি কানেক্টেড কম্পোনেন্ট গুলোকে আলাদা করে ফেলতে পারি। এই লেখাটা পড়ার আগে অবশ্যই টপলোজিকাল সর্টিং আর ডেপথ ফার্স্ট সার্চ এর ডিসকভারি এবং ফিনিশিং টাইম সম্পর্কে ধারণা থাকতে হবে।

নিচের গ্রাফটা দেখ:

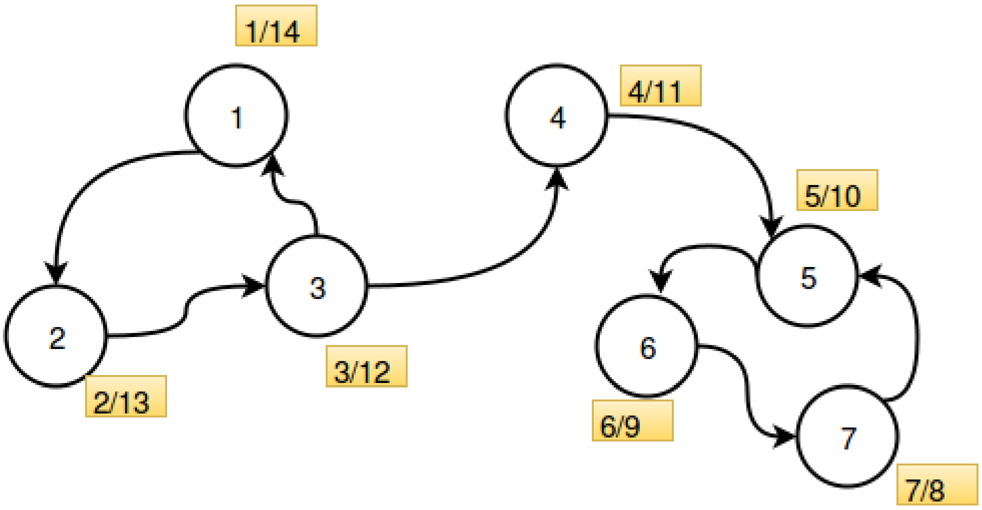


প্রথমেই একটা ভুল পদ্ধতিতে অনেকে SCC বের করার চেষ্টা করে। সেটা হলো যেকোনো নোড থেকে ডিএফএস চালিয়ে যেসব নোডে যাওয়া যায় তাদেরকে একটা কম্পোনেন্ট হিসাবে ধরা। কিন্তু খুব সহজেই বোঝা যায় এটা কাজ করবে না, উপরের গ্রাফে ১ থেকে ডিএফএস চালালে সবগুলো নোড ভিজিট করা যাবে, কিন্তু ১ থেকে ৪ এ যাওয়া গেলেও ৪ থেকে ১ এ যাবার কোনো পথ নেই, তাই এরা একই কানেক্টেড কম্পোনেন্ট এর অংশ না। এই পদ্ধতিতে সমস্যা হলো ডিএফএস কানেক্টেড কম্পোনেন্ট থেকে বের হয়ে অন্য কম্পোনেন্ট এ চলে যায়। এই সমস্যা সমাধান করতে আমরা একটু বুদ্ধিমানের মত ডিএফএস চালাবো।

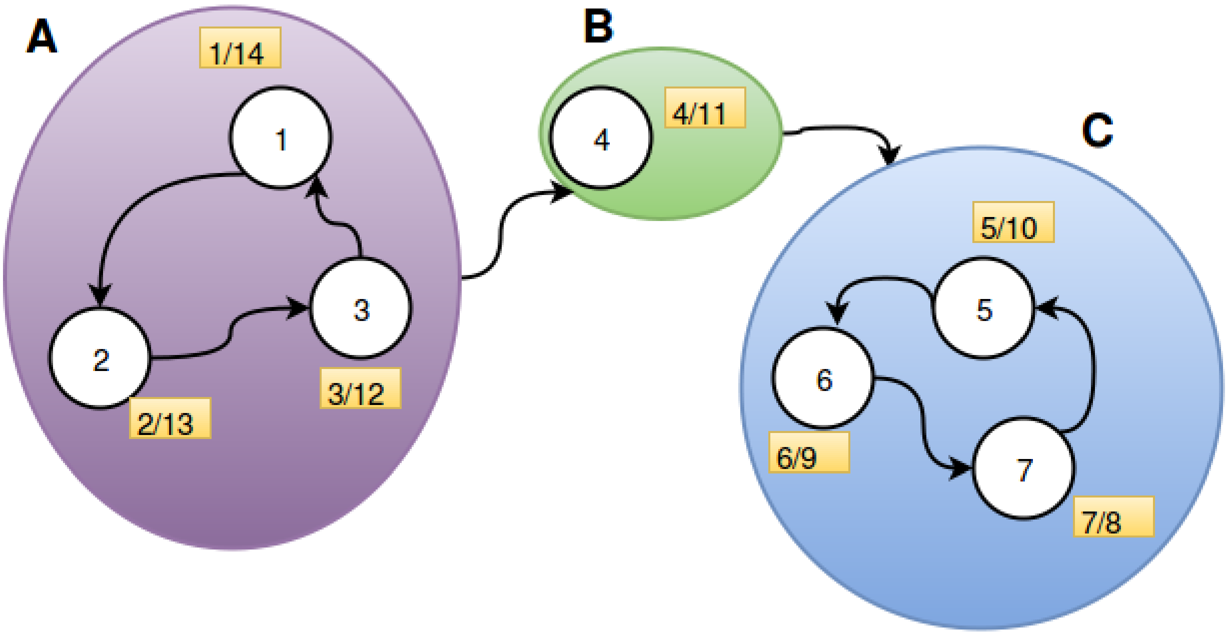
দুটি নোড u,v একই SCC তে থাকবে শুধুমাত্র যদি u থেকে v তে যাবার পথ থাকে এবং v থেকে u তে যাবারও পথ থাকে।

প্রথমে আমরা ১ থেকে ডিএফএস চালিয়ে সবগুলো নোডের ডিসকভারি টাইম আর ফিনিশিং টাইম লিখে ফেলি। নোডগুলো ১,২,৩,৪,৫,৬,৭ অর্ডারে ভিজিট করলে আমরা নিচের ছবির মত স্টার্টিং/ফিনিংশিং টাইম পাবো:

top



এখন বোঝার সুবিধার জন্য স্ট্রংলি কানেক্টেড কম্পোনেন্টের সবগুলো নোডকে একটা বড় নোড মনে করি:



লক্ষ্য করো, গ্রাফটাকে এভাবে 'ডিকম্পোজ' করার পর গ্রাফটিতে আর কোনো সাইকেল থাকা সম্ভব না, অর্থাৎ গ্রাফটি একটি ড্যাগ বা ডিরেক্টেড অ্যাসাইক্লিক গ্রাফে পরিণত হয়েছে। এখন বড় নোডগুলোকে সহজেই টপোলজিকাল অর্ডারে সাজানো সম্ভব, অর্ডারটা হবে A,B,C।

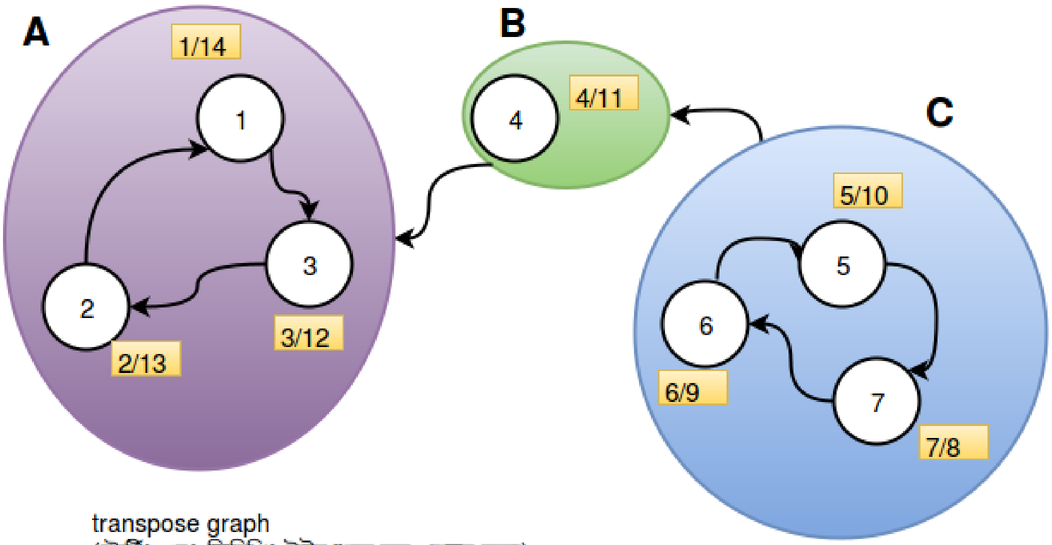
এখন লক্ষ্য করো ড্যাগে একটা নোড D1 থেকে অন্য নোড D2 এ যাওয়া যায় তাহলে D1 টপোলজিকাল অর্ডারে D2 এর আগে অবশ্যই থাকবে। আবার আমরা আগেই জানি যে টপোলজিকাল অর্ডারে যে আগে থাকে তার ফিনিশিং টাইম বেশি হয় কারণ অন্যান্য সব নোডের কাজ শেষ করে ওই নোডে ফিরে আসতে হয়।

তাহলে D1 যদি টপোলজিকাল অর্ডারে D2 এর আগে থাকে তাহলে যেসব ছোটো ছোটো নোড নিয়ে D2 গঠিত হয়েছে তাদের সবার ফিনিংশিং টাইম অবশ্যই D1 এর সব নোডের থেকে কম হবে।

এখন u থেকে v তে যাওয়া গেলেই তারা একই SCC এর অন্তর্ভূক্ত না, v থেকে u তে যাবার পথও থাকতে হবে। অথবা আমরা বলতে পারি 'উল্টো-গ্রাফ' এও u থেকে v তে যাবার পথ থাকতে হবে!

top

যদি গ্রাফের এজগুলো উল্টে দেয়া হয়, তাহলেও SCC গুলো একই থাকে। একে বলা হয় ট্রান্সপোজ গ্রাফ(transpose) , ট্রান্সপোজ গ্রাফে সাইকেল গুলোর কোনো পরিবর্তন হয় না। মূল গ্রাফে যদি u-v একই SCC এর মধ্যে থাকে তাহলে তারা অবশ্যই একই সাইকেলের মধ্যে থাকবে। এটাই আমাদের অ্যালগোরিদমের মূল ভিত্তি।



transpose graph
(স্টার্টিং এবং ফিনিশিং টাইমগুলো মূল গ্রাফের জন্য)

উপরের ছবিতে আগের গ্রাফের এজগুলো উল্টে দেয়া হয়েছে। ডিসকভারি এবং ফিনিশিং টাইম আগেরটাই লেখা আছে।

এখন লক্ষ্য করো তুমি যদি শুরুতে টপলোজিকাল অর্ডারে আগে থাকা নোডগুলো থেকে ডিএফএস চালাও অর্থাৎ যার ফিনিশিং টাইম বড় সেখান থেকে শুরু করো তাহলে তুমি প্রথম প্রথম SCC টা পেয়ে যাবে।

উপরের গ্রাফে 1 এর ফিনিশিং টাইম সবথেকে বেশি (14)। 1 থেকে ডিএফএস চালালে তুমি যেতে পারবে {1,2,3} নোডগুলোতে যারা একই SCC'র অংশ। এবার {১,২,৩} নোডগুলো গ্রাফ থেকে মুছে ফেল। এরপর 4 এর ফিনিশিং টাইম বড়। 4 থেকে শুধুমাত্র {4} এ যাওয়া যায়। এরপর 5 থেকে ডিএফএস চালাবো, সেখান থেকে যাওয়া যায় {5,6,7} নোডগুলোতে যারা একটি SCC এর অংশ।

যেসব নোডগুলো একই কম্পোনেন্ট এর অংশ তাদের কে আমরা আলাদা লিস্টে সেভ করে রাখবো নিজের সুডোকোডটা দেখো:

```
 procedure DFS(G, u):
      color[u]  ← GREY
      for all edges from u to v in G.adjacentEdges(u) do
             if color[v]=WHITE
                     DFS(G,v)
             end if
     end for
     stk.add(source)
     return

procedure DFS2(R,u, mark)
      components[mark].add(u) //save the nodes of the new component
      visited[u] ← true
      for all edges from u to v in R.adjacentEdges(u) do
              if visited[v] ← false
```

```
19                          DFS2(R,v, mark)
20                  end if
21          end for
22          return

23      procedure findSCC(G):
24           stk ← an empty stack
25           visited[] ← null
26           color[] ← null
27           components[] ← null
28           mark=0
29           for each u in G
30                 if color[u]=WHITE
31                        DFS(G,u)
32                 end if
33           end for
34           R=reverseEdges(G)
35           while stk not empty
36                 u=stk.removeTop()
37                 if visited[u]=false
38                      mark=mark+1 //A new component found, it will be identified by ‘mark’
39                      DFS2(R,u,mark)
40                 end if
41            end for
42            return components
```

কোডটা একটু বড় মনে হলেও বোঝা খুব সহজ। প্রথমে একটা ডিএফএস চালিয়ে ফিনিশিং টাইম অনুযায়ী নোডগুলো সর্ট করছি। একটা স্ট্যাক ব্যবহার করে কাজটা করছি। যার ফিনিশিং টাইম কম সে কাজ আগে শেষ করে ১১ নম্বর লাইনে আসবে, তখন সেই নোডটা স্ট্যাকে ঢুকিয়ে রাখবো। সবশেষে স্ট্যাকের উপরে যে নোড থাকবে তার ফিনিশিং টাইম হবে সব থেকে বেশি। এর পর ২য় ডিএফএস চালিয়ে কম্পোনেন্টগুলো আলাদা করে ফেলবো। mark নামের ভ্যারিয়েবল টা ব্যবহার করছি প্রতিটা কম্পোনেন্ট এর আলাদা নাম দেয়ার জন্য, ছবিতে যেভাবে A,B,C নাম দেয়া হয়েছে।

সলভ করার জন্য কিছু প্রবলেম পাবে এখানে।

ফেসবুকে মন্তব্য

0 comments

top