**গ্রাফ থিওরিতে হাতেখড়ি**

লেখক: শাফায়েত

[গ্রাফ থিওরিতে হাতেখড়ি – ১](http://www.shafaetsplanet.com/planetcoding/?p=143)

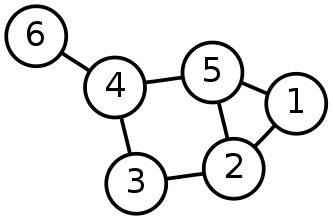
মনে করো তুমি একটি পিরামিডের ভিতরে ঢুকেছো হাজার বছর আগের ধনরত্নের সন্ধানে। তোমার কাছে আছে একটি মাত্র টর্চ লাইট যার ব্যাটারি শেষ হয়ে যাবে ২ ঘন্টা পরেই, এর মধ্যেই তোমাকে সেই রত্ন নিয়ে বের হয়ে আসতে হবে। তোমার কাছে একটি ম্যাপ আছে যেখানে পিরামিডের ভিতরের টানেলগুলো আকা আছে, তুমি জানো কোন টানেল দিয়ে যেতে তোমার কত সময় লাগে। তুমি কি পারবে ২ঘন্টার মধ্যে রত্ন নিয়ে বের হয়ে আসতে? হিসাবে গন্ডগোল হলে অন্ধকার টানেলে তুমি আটকা পড়বে! সাধারণ কারো কাছে এটা একটা বড় সমস্যা হলেও তোমার কাছে এটা কিছুই না, কারণ তুমি এখন শিখতে যাচ্ছো গ্রাফ থিওরী নামক অসাধারণ একটা বিষয়। এরপরে তুমি চোখের নিমিষে তোমার ল্যাপটপে কয়েক লাইন কোড লিখে রত্ন নিয়ে নিরাপদে ফিরে আসতে পারবে!

গ্রাফ থিওরি আমার খুবই প্রিয় একটা টপিক। রিয়েল লাইফের অনেক ধরণের সমস্যার সমাধান করে ফেলা যায় গ্রাফ দিয়ে মডেলিং করে তাই এটা খুবই গুরুত্বপূর্ণ একটা টপিক। তুমি স্কুল-কলেজে থাকতে গ্রাফ বলতে যা বুঝতে এটা আসলে ঠিক সেটা না। গ্রাফ থিওরীর কাজ হলো দুনিয়ার তাবৎ সব প্রবলেমকে শহর-রাস্তার প্রবলেম বানিয়ে ফেলে কিছু অ্যালগোরিদম চালিয়ে সেটাকে সলভ করা, অতিসরলীকরণ হলেও কথাটা খুব একটা ভুল না।

যেমন ধরো পিরামিডের প্রবলেমটাকেই একজন প্রোগ্রামারের কাছে দিলে সে বলবে মনে করি প্রতিটা টানেল হলো একটা করে রাস্তা আর পিরামিডের ভিতরের ঘরগুলো হলো শহর, এবার শহরের প্রবেশপথ থেকে রাজধানীতে যাবার (যে ঘরে রত্ন আছে!) দ্রুততম পথ বের করতে হবে! হয়ে গেলো প্রবলেমটাকে পরিচিত একটা প্রবলেমের সাথে মিলানো, এবার ধাম করে ডায়াক্সট্রা অ্যালগোরিদম বসিয়ে দাও কাজ শেষ। এভাবে যেকোনো প্রবলেমকে পরিচিত একটা প্রবলেমে রূপান্তর করাকে মডেলিং করা বলে। গ্রাফ থিওরি শেখার পর তোমার চিন্তা-ভাবনা ধরণ অনেকটা পাল্টে যাবে,সম্পূর্ণ নতুন উপায়ে তুমি বিভিন্ন সমস্যা সমাধান করতে পারবে যেসব সমস্যা আগে কখনো সমাধান করতে পারোনি।

আমরা একটু জেনে নেই গ্রাফ জিনিসটা আসলে কি, তাহলে এটা কি কাজে লাগে সেটা নিয়ে আরো আলোচনা করতে পারবো। শুরুতেই গ্রাফের সংজ্ঞা বা ফর্মাল কথাবার্তার দিকে যাবনা, সহজ ভাষায় বুঝাতে চেষ্টা করবো। সংক্ষেপে বলি গ্রাফ আসলে কি।

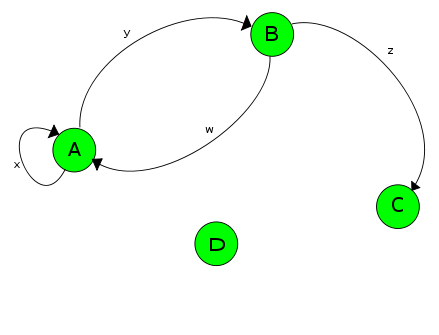
ধরা যাক ৬টি শহরকে আমরা ১,২,৩,৪,৫,৬ দিয়ে চিহ্নিত করলাম, তুমি চাইলে ঢাকা,কুমিল্লা দিয়েও করতে পারো। এবার যে শহর থেকে যে শহরে সরাসরি রাস্তা আছে তাদের মধ্যে লাইন টেনে দিলাম:



এটা একটা খুবই সিম্পল গ্রাফ। প্রতিটি শহরকে আমরা বলব একটি করে **নোড বা ভারটেক্স(node or vertex)** আর যে লাইন টেনেছি তাদের বলবো **এজ(edge)**। ১ থেকে ৫ এর মধ্যে এজ আছে তার মানে হলো ১ থেকে ৫ এ সরাসরি যাওয়া যায়। নোড/ভারটেক্স,এজ শব্দদুটি ঘুরেফিরে বারবার আসবে। পিরামিডের প্রবলেমে রুমগুলোকে তুমি বলতে পারো নোড আর অন্ধকার টানেলগুলো এজ।

এখন নোড দিয়ে শহর না বুঝিয়ে বুঝাতে পারে একজন করে মানুষ,এজ গুলো দিয়ে বুঝাতে পারে কার কার সাথে কার পরিচয় আছে। **এককথায় নোডের কাজ কোন একধরণের অবজেক্টকে রিপ্রেজেন্ট করা আর এজ এর কাজ হলো দুটি অবজেক্টের মধ্যে সম্পর্কটা দেখানো।**অবজেক্ট অনেক কিছুই হতে পারে, সম্পর্কও সেরকম অনেক কিছু হতে পারে। যেমন তুমি ইন্টারনেটের কথা চিন্তা করো, তোমার আমার পিসিগুলো হলো একটি করে নোড আর তাদের মধ্যে সম্পর্ক হলো যে ডাটা পাঠানো হচ্ছে সেটা পাঠাতে কত সময় লাগছে, কয়টা প্যাকেট লস হচ্ছে ইত্যাদি ইত্যাদি।

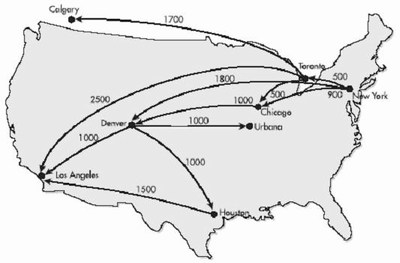
এজগুলো ডিরেক্টেড বা আনডিরেক্টেড হতে পারে। যেমন ঢাকা থেকে সিলেটে যাবার রাস্তা থাকলেও সিলেট থেকে ঢাকায় যাবার সরাসরি রাস্তা নাও থাকতে পারে,এক্ষেত্রে আমরা তীর চিহ্ন নিয়ে দিক নির্দেশ করব। তীর চিহ্ন দেয়া না থাকলে ধরে নিব এজটিটি আনডিরেক্টেড বা বাইডিরেকশনাল। সবগুলো নোড একসাথে যুক্ত নাও থাকতে পারে,সে ক্ষেত্র তাকে বলব ডিসকানেক্টেড(disconnected) গ্রাফ। যেমন এমন হতে পারে আমার বাসা এমন কোনো জায়গায় যে তোমার বাসা থেকে আসার কোনো পথ নেই।



ছোট করে লেখা x,y,w এগুলো কি? এগুলো হলো ওয়েট(weight)। ওয়েট বলতে বুঝাতে পারে সময়,টাকা,দূরত্ব,তাপমাত্রা ইত্যাদি ইত্যাদি, অর্থাৎ নোডদুটোর রিলেশনই আসলে ওয়েট। যদি ওয়েটের মান উল্লেখ করে না দেয়া থাকে তাহলে আমরা ধরে নিব সবগুলো এজ এর ওয়েট সমান।

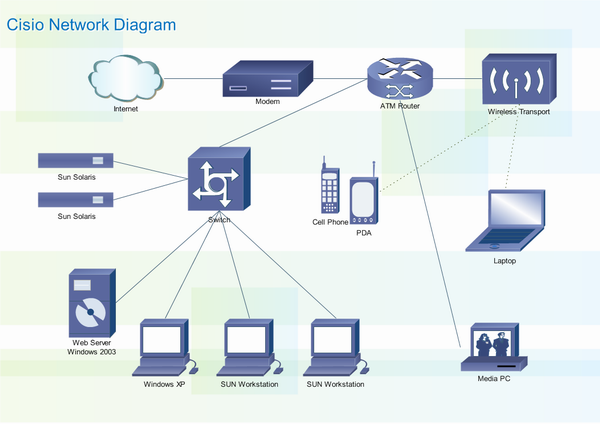
আপাতত এসব টার্মগুলো মোটামুটি মাথায় রাখলেই চলবে,পরবর্তিতে আরো কিছু টার্মের সাথে আমরা পরিচিত হবো।

গ্রাফ অ্যালগোরিদমের কাজ হলো এই গ্রাফটা থেকে দরকারী সব তথ্য বের করে আনা। পিরামিডের প্রবলেমটায় আমাদের জরুরী তথ্যটা ছিলো একটা নির্দিষ্ট সময়ের মধ্যে রত্ন নিয়ে ফিরে আসতে পারবো নাকি। একটা বিখ্যাত প্রবলেম আছে যার নাম চাইনিজ পোস্টম্যান প্রবলেম, পোস্টম্যান প্রতিটি রাস্তায় হেটে হেটে চিঠি দিবে, তার হাটার দূরত্ব কমাতে হবে। সমস্যাটির খুব ভালো কোনো সমাধান কেও দিতে পারেনি, তুমি পারলে এক নিমিষে সারা বিশ্বে বিখ্যাত হয়ে যাবে!

গ্রাফ রিয়েল লাইফে কিভাবে কাজে লাগে তার আরেকটি উদাহরণ নিচের ছবিটি:  


এটা একটা ফ্লাইট চার্ট, গ্রাফে কোন শহর থেকে কোন শহরে দূরত্ব কত দেখানো হয়েছে। এই গ্রাফ ব্যবহার করে এরোপ্লেন কোম্পানি তাদের শিডিউল,খরচ ইত্যাদি নির্ধারণ করতে পারবে।

ইন্টারনেটে ডাটা ট্রান্সফার কাজ করে পুরোপুরি গ্রাফ থিওরির উপর নির্ভর করে। নিচের ছবিটি দেখ:



এরকম একটি নেটওয়ার্কে তুমি কোন পিসিতে বসে অন্য পিসি থেকে কিছু ডাউনলোড করলে সেটা অন্য কয়েকটি নোড বা পিসি ঘুরে তোমার কাছে আসবে। যত কম নোড ঘুরতে হবে তত তাড়াতাড়ি ডাউনলোড হবে। গ্রাফ অ্যালগোরিদম ব্যবহার করে সহজেই বের করা যায় কোন পথে গেলে সময় সবথেকে কম লাগবে,এ ক্ষেত্রে সময় হলো ওয়েট। তুমি যখন নেটওয়ার্কিং পড়বে তখন দেখবে নেটওয়ার্কিং বইতে বেশ কিছু গ্রাফ অ্যালগোরিদম আছে যেগুলা ব্যবহার করে ইন্টারনেট কাজ করে।

গ্রাফ থিওরি অনেক প্রাচীন একটা টপিক, কম্পিউটার আবিষ্কারের বহু আগে থেকে মানুষ গ্রাফ সম্পর্কে জানে। তোমরা নিশ্চয়ই কনিসবার্গ ব্রীজের প্রবলেমের সাথে পরিচিত, ওই যে ৭টা ব্রীজ দিয়ে শহরের বিভিন্ন অংশে যেতে হয়, কোনো ব্রীজে দুইবার যাওয়া যায়না, সেই প্রবলেমটা। ১৭৩৬ সালে অয়লার এই প্রবলেমের উপর পেপার পাবলিশ করেন, গ্রাফ থিওরির উপর এটাই প্রথম পেপার। তুমি একটু সার্চ করলেই এ প্রবলেম নিয়ে অসংখ্য তথ্য পাবে।

গ্রাফের বইয়ের ফর্মাল সংজ্ঞা দেখলে প্রথমে একটু ভয় লাগতে পারে, মনযোগ দিয়ে পড়লে দেখবে উপরের কথাগুলোই গাণিতিক ভাষায় লিখেছে। সেখানে হয়তো লেখা থাকবে set of vertex, set of edge কথাগুলো, এগুলো দিয়ে আসলে বুঝাচ্ছে একটা গ্রাফে কিছু ভারটেক্স আর এজ থাকে এবং তাদের মধ্যে কিছু সম্পর্ক থাকে যেটা তুমি এর মধ্যেই জেনে গিয়েছো!

এগুলো গেল একেবারেই প্রাথমিক কথাবার্তা। পরবর্তি লেখায় তুমি জানতে পারবে কিভাবে [ভ্যারিয়েবলে গ্রাফ স্টোর](http://www.shafaetsplanet.com/planetcoding/?p=184)করতে হয়। আশা করি গ্রাফ থিওরীতে তোমার যাত্রাটা দারুণ আনন্দের হবে, অনেক কিছু শিখতে পারবে।

[অন্যান্য পর্ব](http://www.shafaetsplanet.com/planetcoding/?tag=%E0%A6%97%E0%A7%8D%E0%A6%B0%E0%A6%BE%E0%A6%AB-%E0%A6%A5%E0%A6%BF%E0%A6%93%E0%A6%B0%E0%A6%BF)

Top of Form

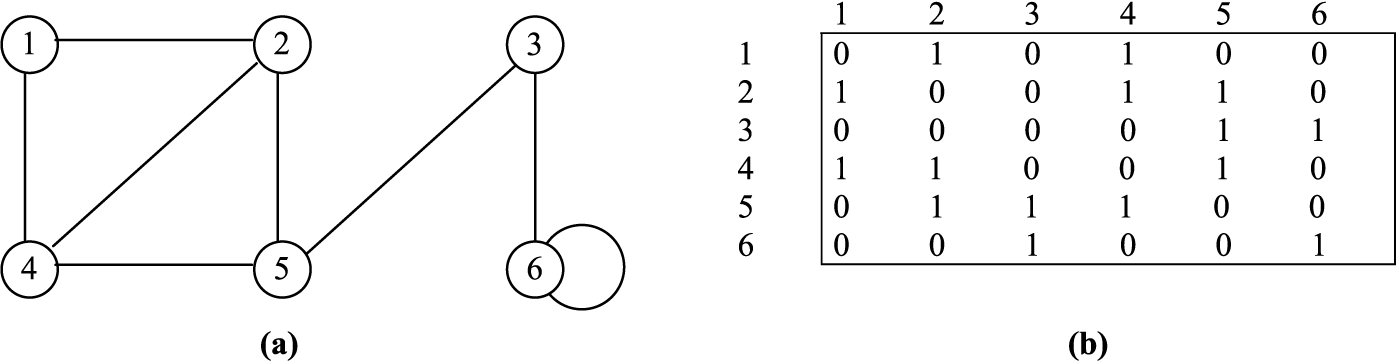
[গ্রাফ থিওরিতে হাতেখড়ি – ২ (ভ্যারিয়েবলে গ্রাফ স্টোর-১)](http://www.shafaetsplanet.com/planetcoding/?p=184)

আগের পোস্টে আমরা দেখেছি গ্রাফ থিওরি কি কাজে লাগে,আর এলিমেন্টারি কিছু টার্ম শিখেছি। এখন আমরা আরেকটু ভিতরে প্রবেশ করবো। প্রথমেই আমাদের জানা দরকার একটা গ্রাফ কিভাবে ইনপুট নিয়ে স্টোর করে রাখা যায়। অনেকগুলো পদ্ধতির মধ্যে দুটি  খুব কমন:

১. অ্যাডজেসেন্সি ম্যাট্রিক্স(adjacency matrix)

২. অ্যাডজেসেন্সি লিস্ট(adjacency list)

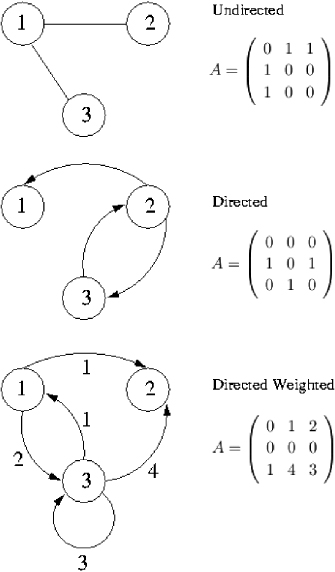
অ্যাডজেসেন্ট(adjacent) শব্দটার অর্থ “কোন কিছুর পাশে”। যেমন তোমার পাশের বাড়ির প্রতিবেশিরা  তোমার অ্যাডজেসেন্ট। গ্রাফের ভাষায় এক নোডের সাথে আরেকটা নোডে যাওয়া গেলে ২য় নোডটি প্রথমটির অ্যাডজেসেন্ট। এই পোস্টে আমরা ম্যাট্রিক্সের সাহায্যে কোন নোড কার অ্যাডজেসেন্ট অর্থাৎ কোন কোন নোডের মাঝে এজ আছে সেটা কিভাবে স্টোর করা যায় দেখবো। ম্যাট্রিক্স বলতে এখানে জাস্ট ২-ডি অ্যারে বুঝানো হয়েছে, তাই ঘাবড়ে যাবার কিছু নেই!



গ্রাফের পাশে একটি টেবিল দেখতে পাচ্ছ। এটাই আমাদের অ্যাডজেসেন্সি ম্যাট্রিক্স। ম্যাট্রিক্সের [i][j] ঘরে 1 থাকে যদি i থেকে j তে কোনো এজ থাকে, না থাকলে ০ বসিয়ে দেই।

এজগুলা ওয়েটেড হতে পারে, যেমন ঢাকা থেকে চট্টগ্রামে একটা এজ দিয়ে বলে দিতে পারে শহর দুটির দূরত্ব ৩০০ কিলোমিটার। তাহলে তোমাকে ম্যাট্রিক্সে ওয়েটও বসাতে হবে।

উপরের গ্রাফটি**বাইডিরেকশনাল বা আনডিরেক্টেড**, অর্থাৎ ১ থেকে ২ এ যাওয়া গেলে ২ থেকে ১ এও যাওয়া যাবে। যদি গ্রাফটি **ডিরেক্টেড** হতো তাহলে এজগুলোর মধ্যে তীরচিহ্ন থাকতো। তখনো আমরা আগের মতো করেই ম্যাট্রিক্সে স্টোর করতে পারবো। নিচের ছবিতে সুন্দর করে সবগুলো কেস দেখানো হয়েছে:



একটা ব্যাপার লক্ষ করো, গ্রাফ আনডিরেক্টেড হলে ম্যাট্রিক্সটি সিমেট্রিক হয়ে যায়, অর্থাৎ mat[i][j]=mat[j][i] হয়ে যায়।

**ছোট একটা এক্সারসাইজ:**

কল্পনা কর একটি গ্রাফ যার ৩টি নোড আছে edge সংখ্যা ৩,এবং সবগুলো edge bidirectional । edge গুলো হলো ১-২(cost ৫),২-৩(cost ৮),১-৩(cost ৩)। এটার adjacency matrix টা কেমন হবে?

চট করে নিজেই খাতায় একে ফেলতে চেষ্টা কর এবং নিচের উত্তরের সাথে মিলিয়ে দেখো:

0 5 3  
5 0 8  
3 8 0

আশা করি বুঝতে পারছ কিভাবে ম্যাট্রিক্সটি আকলাম। না বুঝলে উপরের অংশটা আরেকবার পড়ে ফেল।

**গ্রাফ ইনপুট যেভাবে দেয়া হবে:**

ঠিক উপরের ম্যাট্রিক্সটা প্রোগ্রামিং প্রবলেমে ইনপুট হিসাবে দিয়ে দেয়া হতে পারে,শুরুতে শুধু নোড সংখ্যা বলে দিবে। লক্ষ্য কর এই ম্যাট্রিক্সটা ইনপুট নিতে আমাদের এজ সংখ্যা জানা জরুরী না। আমাদের একটি ভ্যারিয়েবল লাগবে নোড সংখ্যা ইনপুট নিতে,আরেকটি ২-ডি অ্যরে লাগবে ম্যাট্রিক্স ইনপুট নিতে।

int N;

int matrix[100][100]; //এই সর্বোচ্চ ১০০ নোডের গ্রাফ স্টোর করা যাবে।

//ডিক্লেয়ার করার পরে ইনপুট নেবার পালা। খুব সহজ কাজ:

scanf("%d",&N);

for(int i=1;i<=N;i++)

for(int j=1;j<=N;j++)

scanf("%d" ,&matrix[i][j]);

সরাসরি ম্যাট্রিক্স না দিয়ে নোড সংখ্যা,edge সংখ্যা বলে দিয়ে edge গুলো কি কি বলে দিতে পারে,এভাবে:

3 3  //৩ টা নোড এবং ৩টা এজ  
1 2 5 //node1-node2-cost  
2 3 8  
1 3 3

এটা ইনপুট নিব এভাবে:

int Node,Edge;

int matrix[100][100];

scanf("%d%d",&Node,&Edge);

for(i=0;i<Edge;i++)

{

int n1,n2,cost;

scanf("%d%d%d",&n1,&n2,&cost);

matrix[n1][n2]=cost;

matrix[n2][n1]=cost;

}

আরো অনেক উপায়ে প্রবলেমে গ্রাফ ইনপুট দিতে পারে। নোডের নম্বর এলোমেলো হতে পারে,যেমন ৩টি নোডকে ১,২,৩ দিয়ে চিহ্নিত না করে ১০০,১০০০০,৪০০ নামে চিহ্নিত করা হতে পারে। সেক্ষেত্রে আমাদের ম্যাপিং করতে হবে। অর্থাত ১০০ কে আমরা ম্যাপ করব ১ দিয়ে,মানে ১০০ বলতে বুঝব ১,১০০০০ বলতে বুঝব ২। index নামক একটি array রেখে index[100]=1;index[100000]=2;এভাবে চিহ্নিত করে দিলেই চলবে। পরে নোড নম্বর ইনপুট দিলে আমার ইনডেক্স থেকে আমাদের দেয়া নম্বর বের করে আনব। ব্যাপারটাকে বলা হয় অ্যারে কম্প্রেশন, তুমি বিস্তারিত জানতে চাইলে পরে কোনো সময় [আমার এই লেখাটা দেখতে পারো](http://www.shafaetsplanet.com/planetcoding/?p=1388)।

**অ্যাডজেসেন্সি ম্যাট্রিক্স ব্যবহার করার সমস্যা:**

মেমরি একটা বিশাল প্রবলেম, এজ যতগুলোই থাকুকনা কেন তোমার লাগছে N\*N সাইজের ম্যাট্রিক্স যেখানে N হলো নোড সংখ্যা। ১০০০০ টা নোড হলো N\*N ম্যাট্রিক্সের সাইজ দাড়াবে ৪\*১০০০\*১০০০ বাইট বা প্রায় ৩৮১ মেগাবাইট!  এজ কম হলে এটা মেমরির বিশাল অপচয়।

কোনো একটা নোড u থেকে অন্য কোন কোন নোডে যাওয়া যায় বের করতে হলে আমাদের N টা নোডের সবগুলো চেক করে দেখতে হবে, টাইমের বিশাল অপচয়!

**অ্যাডজেসেন্সি ম্যাট্রিক্স ব্যবহার করার সুবিধা:**

u-v নোডের মধ্যে কানেকশন আছে নাকি বা cost কত সেটা খুব সহজেই mat[u][v] চেক করে জেনে যেতে পারি।

এই সমস্যাগুলা দূর করে দিবে অ্যাডজেসেন্সি লিস্ট, সাথে নতুন কিছু সমস্যাও হাজির করবে! তোমরা পরের পর্বে সেটা শিখবে। তার আগে তোমাকে একটা জিনিস শিখতে হবে, সেটা হলো C++ এর স্ট্যান্ডার্ড টেমপ্লেট লাইব্রেরি(STL)। আমরা STL এর ভেক্টর ব্যবহার করে কাজ করবো কারণ এটা ব্যবহার করা খুব সহজ। তুমি নিচের দুটি লিংকের সাহায্যে খুবই সহজে শিখতে পারবে:

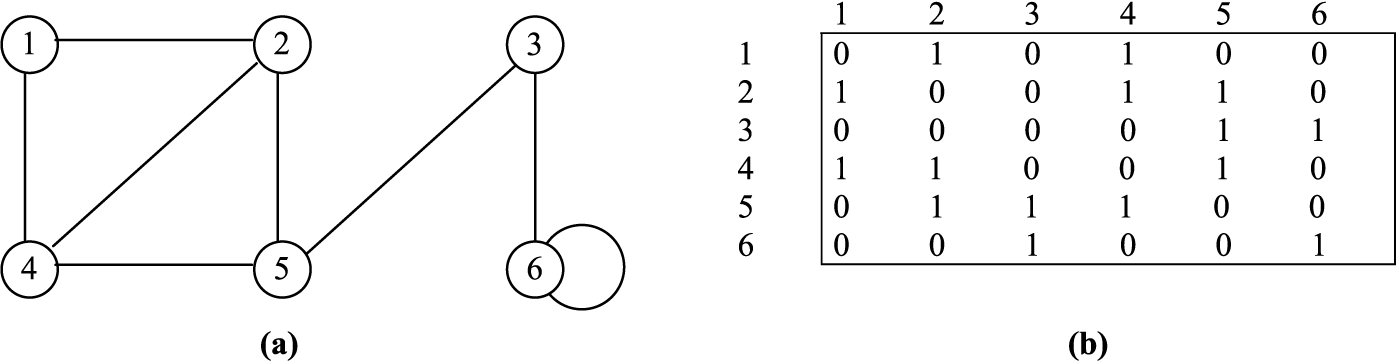
১: <http://sites.google.com/site/smilitude/stl> এটি ফাহিম ভাইয়ের ব্লগের লিংক,তার টিউটোরিয়াল গুলো অদ্বিতীয়।

২: <http://www.cplusplus.com/reference/stl/> STL এর বিভিন্ন ফাংশনের কাজ শেখার জন্য সেরা সাইট।

ভেক্টর ব্যবহার শেখা হয়ে গেলে পড়া শুরু করো পরের পর্ব: [adjacency list](http://www.shafaetsplanet.com/planetcoding/?p=211)

[গ্রাফ থিওরিতে হাতেখড়ি – ৩ (ভ্যারিয়েবলে গ্রাফ স্টোর-২)](http://www.shafaetsplanet.com/planetcoding/?p=211)

এই পর্বে গ্রাফ স্টোর করার ২য় পদ্ধতি অ্যাডজেসেন্সি লিস্ট নিয়ে লিখব। এ পদ্ধতিতে গ্রাফ স্টোর করে কম মেমরি ব্যবহার করে আরো efficient কোড লেখা যায়। এ ক্ষেত্রে আমরা ডায়নামিক্যালি মেমরি অ্যালোকেট করব,ভয়ের কিছু নেই সি++ এর standard template library(STL) ব্যবহার করে খুব সহজে কাজটা করা যায়। আগের লেখার শেষের দিকে STL এর উপর কয়েকটি টিউটোরিয়ালের লিংক দিয়েছি, আশা করছি ভেক্টর কিভাবে কাজ করে এখন তুমি জানো।

অ্যাডজেসেন্সি লিস্ট শুনতে যতটা ভয়ংকর শুনায়,ব্যাপারটি আসলে তেমনই সহজ। আমরা আবার আগের পোস্টের ছবিটিতে ফিরে যাই:  


এবার বাজার করার লিস্টের মত একটি লিস্ট বানাই:

১ নম্বর নোডের সাথে যুক্ত আছে>>> ২ এবং ৪ নম্বর নোড  
২ নম্বর নোডের সাথে যুক্ত আছে>>> ১,৪ এবং ৫ নম্বর নোড  
৩ নম্বর নোডের সাথে যুক্ত আছে>>> ৫ এবং ৬ নম্বর নোড  
৪ নম্বর নোডের সাথে যুক্ত আছে>>> ১,২ এবং ৫ নম্বর নোড  
৫ নম্বর নোডের সাথে যুক্ত আছে>>> ২,৩ এবং ৪ নম্বর নোড  
৬ নম্বর নোডের সাথে যুক্ত আছে>>> ৩ এবং ৬ নম্বর নোড

এটাই অ্যাডজেসেন্সি লিস্ট, কোন নোডের সাথে কোন নোড যুক্ত আছে সেটার একটা তালিকা।  কিন্তু কোড করার সময় কিভাবে এই লিস্টটা স্টোর করবো?

**প্রথম উপায়(অ্যারে):**

সাধারণ ২ডি অ্যারে ব্যবহার করে লিস্টটি স্টোর করা যায়। যেমন:

arr[1][1]=2, arr[1][2]=4;

arr[2][1]=1; arr[2][2]=4, arr[2][3]=5;

কিন্তু এভাবে স্টোর করলে কিছু সমস্যা আছে:

সমস্যা ১. আমাদের ৬টি নোড আছে। প্রতি নোডের সাথে সর্বোচ্চ ৬টি নোড যুক্ত থাকতে পারে(ধরে নিচ্ছি দুটি নোডের মধ্যে ১টির বেশি সংযোগ থাকবেনা)। এ ক্ষেত্রে আমাদের লাগবে [6][6]আকারের ইন্টিজার অ্যারে। যদি ১ নম্বর নোডের সাথে মাত্র ২টি নোড যুক্ত থাকে তাহলে বাকি array[1][0],array[1][1] কাজে লাগবে,array[1][2] থেকে array[1][6] পর্যন্ত জায়গা কোনো কাজেই লাগবেনা। মনে হতে পারে এ আর এমন কি সমস্যা। কিন্তু চিন্তা কর ১০০০০ টি নোড আছে এমন একটি গ্রাফের কথা। [10000][10000] integer অ্যারে তুমি ব্যবহার করতে পারবেনা,memory limit অতিক্রম করে যাবে,এছাড়া এভাবে মেমরি অপচয় করা ভালো প্রোগ্রামারের লক্ষণ নয় :) । অ্যাডজেসেন্সি ম্যাট্রিক্স ব্যবহার করার সময় যেমন মেমরির সমস্যা হয়েছিলো, এখনও সেই সমস্যা রয়ে যাবে।

সমস্যা ২. অ্যারের কোন ইনডেক্সে কয়টি এলিমেন্ট আছে তার হিসাব রাখতে প্রতি ইনডেক্সের জন্য আরেকটি ভ্যারিয়েবল মেইনটেইন করতে হবে।

**দ্বিতীয় উপায়(ভেক্টর):**

সমস্যাগুলা দূর করতে আমরা STL vector বা list ব্যবহার করে গ্রাফ স্টোর করব। ভেক্টর/লিস্টে তোমাকে লিস্টের সাইজ বলে দিতে হবেনা,খালি সর্বোচ্চ নোড সংখ্যা বলে দিলেই হবে। এই টিউটোরিয়ালে আমি ভেক্টর ব্যবহার করব কারণ list এ বেশ কিছু সমস্যা আছে।

১০০০০০ নোডের গ্রাফ ইনপুট দেয়ার সময় কখনো ম্যাট্রিক্স হিসাবে দিবেনা,তাহলে ইনপুটের আকারই মাত্রাতিরিক্ত বিশাল হয়ে যাবে। আগের পোস্টে ২য় উদাহরণে যেভাবে দেখিয়েছি সেরকম  ইনপুট দিতে পারে, অর্থাৎ প্রথমে নোড আর এজ সংখ্যা বলে দিয়ে তারপর কোন নোডের সাথে কে যুক্ত আছে বলে দিবে। উপরের গ্রাফের জন্য ইনপুট:

6 8 //node-edge  
1 2 //node1-node2  
1 4  
2 4  
2 5  
4 5  
5 3  
3 6  
6 6

এটি ভেক্টর দিয়ে ইনপুট নিব এভাবে:

#include<cstdio>

#include<vector>

using namespace std;

#define MAX 100000 //maximum node

vector<int>edges[MAX];

vector<int>cost[MAX]; //parallel vector to store costs;

int main()

{

int N,E,i;

scanf("%d%d",&N,&E);

for(i=1;i<=E;i++)

{

int x,y;

scanf("%d%d",&x,&y);

edges[x].push\_back(y);

edges[y].push\_back(x);

cost[x].push\_back(1);

cost[y].push\_back(1);

}

return 0;

}

cost নামক ভেক্টরটি এ গ্রাফের ক্ষেত্রে দরকার ছিলনা,তবে ওয়েটেড গ্রাফে অবশ্যই দরকার হবে। নিশ্চয়ই বুঝতে পারছ edge ও cost ভেক্টর দুটি সমান্তরাল ভাবে কাজ করবে,অর্থাত edge ভেক্টরের যে পজিশনে তুমি দুটি নির্দিষ্ট নোডের কানেকশন পাবে cost ভেক্টরের সেই পজিশনেই তুমি cost পাবে।

যদি ১০০০ বা তার কম নোড থাকে তাহলে ম্যাট্রিক্স বা লিস্ট কোনো ক্ষেত্রেই মেমরি সমস্যা হবেনা। তাও আমরা ভেক্টর দিয়েই গ্রাফ স্টোর করব। কারণ, চিন্তা কর তোমাকে ১০০টা নোডের ম্যাট্রিক্সে ১ এর সাথে কি কি সংযুক্ত আছে বের করতে matix[1][0],matrix[1][1]…….matrix[1][99] এভাবে ১০০টি পজিশন চেক করে কোনটায় কোনটায় ০ নেই বের করতে হবে,১ নম্বর নোডের সাথে যতটি নোডই সংযুক্ত থাকুকনা কেন। তাই এখানেও অ্যারে আমাদের বাড়তি সুবিধা দিতে পারছেনা।

**একটা নোডের সাথে কোন কোন নোড যুক্ত আছে বের করা:**

ধরো তুমি ১ নম্বর নোডের সাথে যুক্ত সবগুলো নোডের নম্বর চাও, তুমি তাহলে edges[1] এর সাইজ পর্যন্ত লুপ চালাবে এভাবে:

</pre>

size=edges[1].size();

for(int i=0; i < size ; i++)

printf("%d ",edges[1][i]);

<pre>

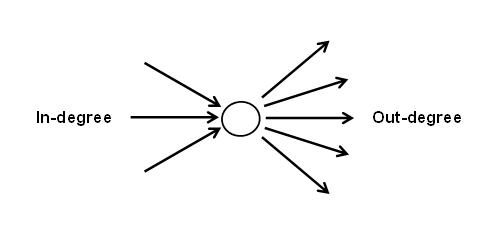
১ >> ০ ১ ০ ০ ০ ১ ১ ০ পুরোটা ঘুরে আসার থেকে ১>>২,৬,৭ ঘুরে আসতে কম সময় লাগবে,তাই নয়কি? :) ।

**অ্যাডজেসেন্সি ম্যাট্রিক্স কখন লিস্ট অপেক্ষা সুবিধাজনক?**

যদি কোনো প্রবলেমে তোমার u,v নোডের এর মধ্যে কোনো এজ আছে নাকি চেক করতে বলে তখন লিস্ট ব্যবহার করলে তোমাকে লুপ চালিয়ে চেক করতে হবে অথবা বাইনারি সার্চ করতে হবে, কিন্তু ম্যাট্রিক্সে জাস্ট matrix[u][v] ইনডেক্স চেক করেই বলে দিতে পারবে তাদের মধ্যে কানেকশন আছে নাকি।

**এক্সারসাইজ:**

এ পর্যন্ত বুঝে থাকলে তুমি মোটামুটি bfs,dfs এর মত বেসিক অ্যালগোরিদম গুলো শেখার জন্য প্রস্তুত। পরবর্তি লেখাটি পড়ার আগে একটি ছোট exercise করে ফেল। এমন একটি কোড লিখ যেটায় উপরের মত করে ইনপুট দিলে নিচের কাজগুলো করে:  
১. একটি adjacency list তৈরি করে। (গ্রাফটিকে directed ধরে নিবে,bidirectional নয়)  
২. কোন নোডের সাথে কয়টা নোড যুক্ত আছে,নোডগুলো কি কি সেগুলো প্রিন্ট করে।  
৩. indegree হলো একটি নোডে কয়টি নোড প্রবেশ করেছে,outdegree হলো ঠিক তার উল্টোটা। প্রতিটি নোডের outdegree ও indegree প্রিন্ট কর।



পর্ব-৪,বিএফএস:<http://www.shafaetsplanet.com/planetcoding/?p=604>

### মন্তব্য

**সিয়াম**

ভাইয়া প্রবলেম দিলেন সলভ দিলেন না…  
আর কিছু ইনপুট আউটপুট দেন…?  
স্টেট করার জন্য..?

**শাফায়েত**

@সিয়াম:  
এখানেতো ওরকম কোনো সমস্যা দেয়া হয়নি। ইনপুট নিজের ইচ্ছামত ফরমেটে নেন। এই গ্রাফটি ইনপুট নিতে পারেন:  
<http://gezeiten.org/public/tikz/graphviz-digraph.png>

টিউটোরিয়ালে যেভাবে নিয়েছে সে ফরমেটেই ইনপুট নিতে পারেন। তারপর কোন নোডের সাথে কি কি যুক্ত আছে প্রিন্ট করেন,যেমন:  
1 >> 2,5,8  
2 >> 3  
…..  
…..  
6 >> 3,7,8

তারপর indegree,outdegree,যেমন:  
1>> in 0 out 3  
2>> in 2 out 1  
……  
…….

এভাবে গ্রাফ ইনপুট নেয়া এবং traverse করা শেখা হবে,পরবর্তীতে অ্যালগো শিখতে কাজে লাগবে।

**সিয়াম**

১৯ সেপ্টেম্বর ২০১১

#include  
#include  
#include  
#include  
# define MAX 10000  
using namespace std;  
int main()  
{  
freopen(“in.txt”,”r”,stdin);  
unsigned e,n,i,j;  
vector edge[MAX];  
vector cost[MAX];  
map in,out;  
scanf(“%u %u”,&n,&e);  
for(i=0;i<e;i++)  
{  
int x,y;  
scanf("%d %d",&x,&y);  
edge[x].push\_back(y);  
cost[x].push\_back(1);  
}printf("Connected :\n");  
for(j=1;j<=n;j++)  
{  
printf("%d :",j);  
for(i=0; i < edge[j].size() ; i++)  
{  
in[edge[j][i]]++;  
printf("%d ",edge[j][i]);  
}  
cout<<endl;  
out[j]=edge[j].size();  
}  
printf("Indegree Outdegree: \n");  
for(j=1;j> %d out<< %d\n",j,in[j],out[j]);  
return 0;  
}

ভাইয়া দেখেন সব ঠিক মত আছে কিনা কোন সাজেশন থা্কলে দিয়েন…

**শাফায়েত**

ট্যাগের মধ্যে কোডটা পেস্ট করলে ভালো হতো,সুন্দর ফরমেটে আসতো।  
যাই হোক কোডটা দেখলাম। ঠিকই আছে মনে হচ্ছে। এখন পরের টিউটোরিয়াল খুলো বিএফএসটা শিখে ফেলো :) । আর বুঝতে কোনো সমস্যা হলে জানাবে।

[সঞ্জয় দেবনাথ](http://uhunt.felix-halim.net/id/157834)

ভাইয়া vector দিয়ে গ্রাফ ইনপুট নেয়ার জন্য আপনার প্রদত্ত কোড টা রান করানোর পর ইনপুট না নিয়ে ই স্টপ হয়ে যায় :( কোন ভুল তো খুজে পাচ্ছিনা ।

[সঞ্জয় দেবনাথ](http://uhunt.felix-halim.net/id/157834)

sorry /// হয়েছে :)  
ভেক্টর টা কে গ্লবালি ডিক্লেয়ার করার পর হয়েছে

**শাফায়েত**

ভ্যারিয়েবলের আকার অনেক বড় হলে গ্লোবাল ডিক্লেয়ার করা উচিত,আমি কোডে গ্লোবাল বানিয়ে দিচ্ছি,ধন্যবাদ।

**MMHS**

ভাইয়া, এই code টার কি মোটামুটি efficiency আছে ??  
<http://pastebin.com/ABVwrZct>  
input এই রকম ছিলঃ  
8 10  
1 2  
2 3  
3 4  
4 2  
1 5  
5 6  
6 7  
6 3  
6 8  
1 8  
(আপনার দেওয়া ছবির input অনুযায়ী)  
comment করলে উপকৃত হইতাম …

**Nakib**

ভাইয়া , ভেকটর এ আমার একটা সমস্যা হয়  
#define 10000 MAX  
vector v[MAX]

.........

v[0].push\_back(34);

এভাবে pus\_back করলে নিচের error দেখায়

erroe: ‘v’ does not name a type

**Nakib**

২১ অক্টোবর ২০১৩

স্যরি ভাইয়া উপরের কোড ভুল ছিল

#define MAX 10000  
vector v[MAX]

.........

v[0].push\_back(34);

[Arindam Pal](https://www.facebook.com/arindam1991)

২ নভেম্বর ২০১৩

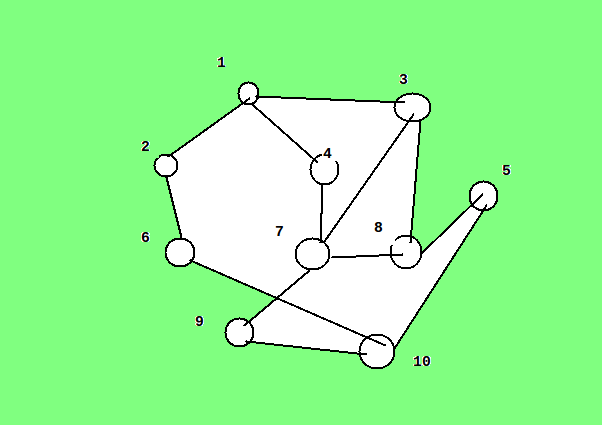
what is meant by edge[x]? is it the x-th location of the array[or vector] edges[MAX]?  
if so, how do you call size of edge[x]?

[গ্রাফ থিওরিতে হাতেখড়ি-৪(ব্রেথড ফার্স্ট সার্চ(bfs))](http://www.shafaetsplanet.com/planetcoding/?p=604)

আগের [পর্বগুলোতে](http://www.shafaetsplanet.com/planetcoding/?p=211) আমরা দেখেছি কিভাবে ভ্যারিয়েবলে গ্রাফ স্টোর করতে হয়। এবার আমরা প্রবলেম সলভিং এর দিকে যাবো। শুরুতেই আমরা যে অ্যালগোরিদমটা শিখব তার নাম breadth first search বা bfs। এটা ব্যবহার করে আমরা এখন unweighted গ্রাফে ২টি নোডের ভিতর শর্টেস্ট পাথ বের করব,unweighted বলতে বুঝাচ্ছি সবগুলো নোডের মধ্যে দুরত্ব সমান,আমরা ধরে নিব দুরত্ব ১।  
এই পর্বে আমি কোনো কোড দিবোনা,শুধুমাত্র আইডিয়া পরিস্কার করবো কারণ সেটাই আসল জিনিস,সামনে কোনো পর্বে হয়তো কোড নিয়ে আলোচনা করবো।  
আইডিয়াটি মূলত এরকম:

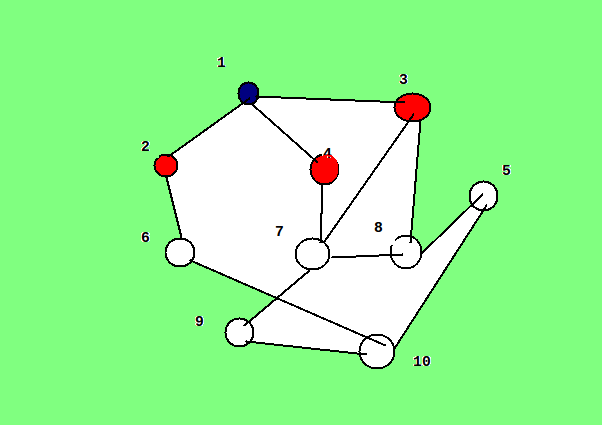
১. কোনো নোড ১ বারের বেশি visit করা হবেনা।  
২. সোর্স নোড ০ নম্বর লেভেলে অবস্থিত।  
৩. সোর্স বা ‘লেভেল ০’ থেকে ১ দুরত্বে অবস্থিত সবগুলো নোড লেভেল ১ এ।  
৪. ‘লেভেল ১’ থেকে ১ দুরত্বে অবস্থিত সবগুলো নোড লেভেল ২ এ।  
৫. ‘লেভেল n’ থেকে ১ দুরত্বে অবস্থিত সবগুলো নোড লেভেল n+1 এ।  
৬. যে নোড যত নম্বর লেভেলে,সোর্স থেকে তার শর্টেস্ট পথের দৈর্ঘ্য তত।

উপরে লেখাগুলো পুরোপুরি মাথায় না ঢুকলেও ক্ষতি নেই,মোটামুটি বুঝলেই হলো,এখন আমরা ছবি দেখে বাকিটা পরিস্কার করব।

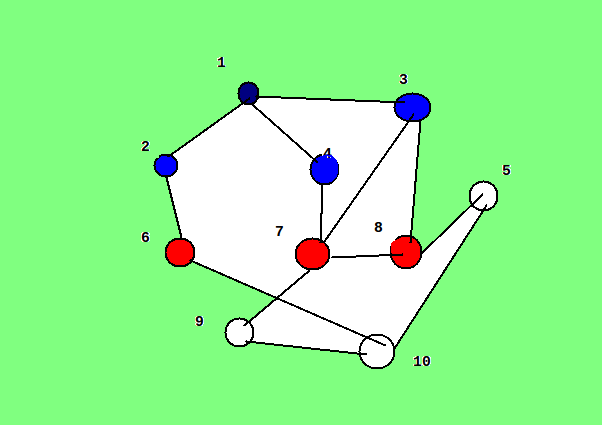


ধর তুমি ১নম্বর শহর থেকে ১০ নম্বর শহরে যেতে চাও। মাথায় রাখবে,১০ নম্বর শহরে যাবার জন্য মাঝে তোমাকে যেসব শহর ভিজিট করতে হবে সেগুলোও শর্টেস্ট পথে করতে হবে। প্রথমে আমরা সোর্স ধরলাম ১. ১ কে তাহলে visited চিহ্নিত করি।

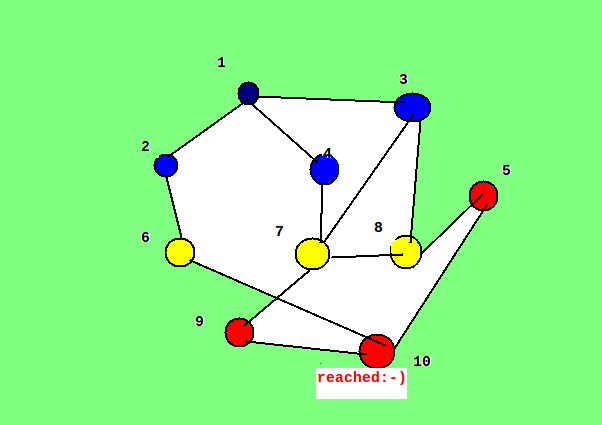
১ থেকে শর্টেস্ট পথে বা ১দুরত্বে যাওয়া যায় ২,৩,৪ নম্বর নোডে। এবার সেগুলোকে আমরা visited চিহ্নিত করি এবং সেগুলো নিয়ে কাজ করি। নিচের ছবি দেখ:



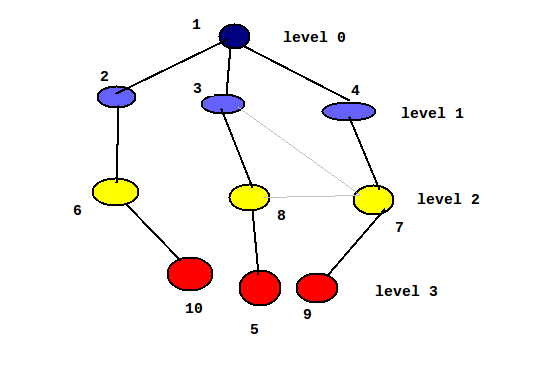
লালনোডগুলো নিয়ে আমরা এখন কাজ করবো। রঙিন সবগুলো নোড visited,**এক নোডে ২বার কখনো যাবোনা**। ২,৩,৪ থেকে শর্টেস্ট পথে যাওয়া যায় ৬,৭,৮ এ। সেগুলো visited চিহ্নিত করি:



লক্ষ কর যে নোডকে যত নম্বর লেভেলে পাচ্ছি,সোর্স থেকে তার শর্টেস্ট পথের দৈর্ঘ্য ঠিক তত। যেমন ২নম্বর লেভেলে ৮কে পেয়েছি তাই ৮ এর দুরত্ব ২। ছবিগুলোকে একেকটা লেভেলের একেক রং দেয়া হয়েছে। আর লাল নোড দিয়ে বুঝানো হয়েছে আমরা এখন ওগুলো নিয়ে কাজ করছি। আমরা ১০ এ পৌছাইনি তাই পরের নোডগুলো ভিজিট করে ফেলি:



আমরা দেখতে পাচ্ছে ২টি লেভেল পার হয়ে ৩ নম্বর লেভেলে আমরা ১০ কে পাচ্ছি। তাহলে ১০ এর শর্টেস্ট পথ ৩। লেভেল বাই লেভেল গ্রাফটাকে explore করে আমরা শর্টেস্ট পথ বের করলাম। এটা ঠিক বাইনারি tree এর lever order traversal এর মত। আসলে এখানে আমরা একটা ট্রি তৈরি করে ফেলেছি। যেসব edge গুলো আমরা ব্যবহার করিনি সেগুলোকে বাদ দিয়ে ছবিটিকে একটু ঘুরিয়ে নিচের মত করে আকতে পারি:



লক্ষ্য কর গ্রাফদুটি একই,খালি নোডগুলো ঘুরানো হয়েছে। যেসব edge ব্যবহার করিনি সেগুলো হালকা করে দিয়েছি,এই edge গুলো বাদ দিলে গ্রাফটি একটি tree হয়ে যায়। tree তে দুটি নোডের মধ্য একটি মাত্র পথ থাকে।

লেখাটি আর বড় করবোনা,KolourPaint এ ছবি আকঁতে আকঁতে আমি টায়ার্ড। যেহেতু তুমি এখন adjacency list,matrix এ গ্রাফ স্টোর করতে পারো,নিজে কিছুক্ষন bfs এর কোড লেখার চেষ্টা কর। কোড খুব সহজ,সোর্স থেকে পরের লেভেলের সবগুলো নোডে যাবে,সেখান থেকে তার পরের লেভেলে যাবে,কোনো নোডে ২বার যাবেনা। উপরের গ্রাফটি স্টোর করে কোনো নোডকে সোর্স ধরে শর্টেস্ট পাথ বের কর।

কোডটি যদি লিখে ফেলতে পারো তাহলে প্র্যাকটিসের জন্য নিচের দুটি প্রবলেম ঝটপট সলভ করে ফেলো:  
[A Node Too Far](http://uva.onlinejudge.org/external/3/336.html)(uva- 336)(শর্টেস্ট পাথ প্রবলেম)  
[Bicoloring](http://uva.onlinejudge.org/external/100/10004.html)(uva-10004)(বাইপারটাইট চেকিং)

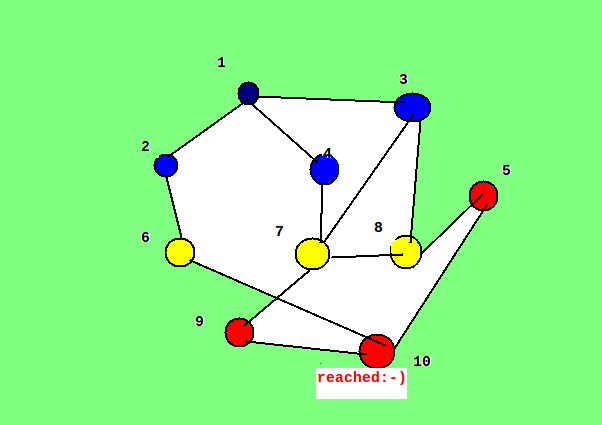
Top of Form

[গ্রাফ থিওরিতে হাতেখড়ি-৫(কোডিং বিএফএস)](http://www.shafaetsplanet.com/planetcoding/?p=639)

[গতপর্বে](http://www.shafaetsplanet.com/planetcoding/?p=604) আমরা দেখেছি bfs কিভাবে কাজ করে। এবার আমরা bfs এর কোড লিখতে শিখব। [আগের পর্বটি](http://www.shafaetsplanet.com/planetcoding/?p=604) পড়া না থাকলে বা মাথায় না থাকলে আরেকবার ভালো করে দেখতে অনুরোধ করছি,কারণ পুরোনো কথা এ পর্বে পুনরাবৃত্তি করা হবেনা।

আমরা এখন জানি যে bfs এ আমাদের কাজ হলো গ্রাফটিকে একটি একটি লেভেলে ভাগ করা,যে যত নম্বর লেভেলে আছে সোর্স থেকে তার দূরত্ব ঠিক তত। আমরা ২ভাবে bfs লেখা দেখব। সাধারণত আমরা ১টি queue ব্যবহার করে bfs লিখি। তবে আমি প্রথমে শিখাবো ২টি ভেক্টর বা সাধারণ অ্যারে ব্যবহার করে bfs,এ পদ্ধতিতে বুঝতে খুব সহজ। তারপর আমরা দেখবো queue দিয়ে স্ট্যান্ডার্ড পদ্ধতিতে কিভাবে লিখতে হয়। তুমি চাইলে scroll করে ২য়টি আগে দেখতে পারো।

[আগের পোস্টের](http://www.shafaetsplanet.com/planetcoding/?p=604) গ্রাফটি ছিল:



আমাদের সোর্স হলো ১। আমরা দুটি ভেক্টর বা অ্যারে নিবো(ভেক্টর নেয়া অনেক বেশি সুবিধাজনক)। মনে করি ভেক্টর দুটি হলো V1,V2. এখন সোর্স ১ কে V1 এ পুশ করি।

V1>>1  
V2>>empty

এবার ১ থেকে যেসব যায়গায় যাওয়া যায় সেগুলো কে V2 তে পুশ করি। তাহলে V1 আর V2 তে এখন আছে:

V1>>1  
V2>>2,3,4

পরের ধাপে V1 কে খালি করে V2 এর সব নোড আবার V1 এ নিয়ে আসি এবং V2 খালি করে ফেলি:

V1>>2,3,4  
V2>>empty

এবার ২,৩,৪ থেকে যেসব যায়গাতে যাওয়া যায় সেগুলো V2 এ পুশ করি:

V1>>2,3,4  
V2>>6,7,8

পরবর্তিতে 6,7,8 নিয়ে কাজ করতে হবে তাই V1 কে খালি করে V2 কে V1 এ নিয়ে আসি:

V1>>6,7,8  
V2>>empty

এবার 6,7,8 থেকে যেসব যায়গাতে যাওয়া যায় সেগুলো V2 এ পুশ করি:

V1>>6,7,8  
V2>>9,10,5

আবার 9,10,5 কে প্রসেস করার জন্য V1 এ নিয়ে আসি:

V2>>9,10,5  
V2>>empty

এবার 9,10,5 থেকে যেসব যায়গাতে যাওয়া যায় সেগুলো V2 এ পুশ করি:

V1>>9,10,5  
V2>>empty

9,10,5 থেকে আর কোথাও যাওয়া যায়না,তাই V2 তে কোনো নোড নেই। তাই আবার V1 কে খালি করে V2 কে V1 এ নেয়া দরকার নেই,কারণ প্রসেস করার মত আর নোড নেই। আমরা এখন লুপ ব্রেক করতে পারি।  
আমাদের কাজ শেষ,আমরা গ্রাফটিকে লেভেল অর্ডারে সার্চ বা ব্রেডথ ফার্সট সার্চ করে ফেলেছি। মনোযোগ দিয়ে পড়লে এতক্ষনে অবশ্যই বুঝ ফেলেছো যে আমরা প্রতিবার V1 এ গ্রাফের একটি করে লেভেল পেয়েছি। উপরে সবুজ রং এর লেখাগুলো লক্ষ করো,প্রতি লুপে আমরা একটি করে লেভেল পেয়েছি।  
বুঝে থাকলে এখন কোড লেখা তোমার জন্য খুব সহজ হবার কথা। তারপরেও আমি কোড লিখে দেখাচ্ছি:

vector<int>G[100];

void bfs(int n,int source)

{

vector<int>V1,V2;

V1.push\_back(source);

int taken[100]={0};

taken1=1;

printf("Level 0::%d\n",source);

for(int loop=1;loop<n;loop++) //n-1 এর থেকে বেশি লেভেল থাকা সম্ভব না,তাই n-1 টি লুপ যথেষ্ট

{

printf("Level %d::",loop);

for(int i=0;i<V1.size();i++)

{

int u=V1[i];

for(int j=0;j<G[u].size();j++)

{

int v=G[u][j]; //u থেকে v তে edge পেয়েছি

if(!taken[v]) //একই নোডে বারবার যাবোনা

{

printf("%d ",v);

taken[v]=1;

V2.push\_back(v); //V1 থেকে যেসব নোডে যাওয়া যায় সেগুলো V2 তে পুশ করছি

}

}

}

if(V2.empty()) {puts("EMPTY");break;}//প্রসেস করার মত নোড নেই

else //V2 তে প্রসেস করার মত নোড আছে তাই সেগুলোকে V1 এ নিয়ে যাবো

{

puts("");

V1.clear();

V1=V2;

V2.clear();

}

}

}

নিজে গ্রাফ ইনপুট নিয়ে ফাংশনটি বসিয়ে কোডটি চালিয়ে দেখ। যদি না পারো,প্রথম দিকের পর্বগুলো দেখে বেসিক জিনিসগুলো আগে বুঝে নাও। কোড দেখে না বুঝলে ক্ষতি নেই,আইডিয়া বুঝলে কোড লিখতে বেশি সময় লাগবেনা। আর আইডিয়া না বুঝলে খালি কোড দেখে লাভ নেই :-) ।

দুটি ভেক্টর ব্যবহার করে bfs লেখার সুবিধাগুলা হলো সহজে প্রতিটা লেভেল কে আলাদা করা যায়,বোঝা সহজ,কোড ডিবাগ করে সুবিধা হয়,বেশ কিছু প্রবলেমে advantage দেয়।

তবে সব বইয়ে বা ল্যাবে queue দিয়ে bfs করানো হয়,সেটাই স্ট্যান্ডার্ড উপায়। এটা বোঝা খুবই জরুরি কারণ queue দিয়ে করলে complexity কিছুটা কম হয়,এছাড়া সামনে যখন dijkstra,bellman শিখবে তখন কাজে লাগবে। queue এর আইডিয়াটি বলছি।

মনে করি Q নামের একটি কিউ আছে। প্রথম সেটায় সোর্স কে প্রবেশ করাবো।

Q>>1

এবার Q এর সামনের(front) নোড থেকে যেসব জায়গায় যাওয়া যায় সেগুলো Q তে পুশ করাবো। Q যেহেতু FIRST IN FIRST OUT তাই নতুন নোড আগের নোডগুলোর পিছে গিয়ে দাড়াবে:

Q>>2,3,4,1

১ এর কাজ শেষ,ওকে এবার ফেলে দিতে(pop) পারি:

Q>>2,3,4

এবার 4 থেকে যেসব যায়গায় যাওয়া যায় সেগুলো পিছে দাড় করিয়ে ৪ কে ফেলে দেই:

Q>>7,8,2,3,4

Q>>7,8,2,3

এবার ৩ নিয়ে একই কাজ করি। যতক্ষণনা queue খালি হয় ততক্ষণ একই কাজ চালিয়ে যাই। প্রতিটি নোডের দুরত্ব হবে সে যে নোড থেকে এসেছে(parent node) তার থেকে ১ বেশি। সোর্স নোডের দুরত্ব ০।

vector<int>G[100];

void bfs(int n,int src)

{

queue<int>Q;

Q.push(src);

int taken[100]={0},distance[100];

taken [src]=1;

distance [src]=0;

while(!Q.empty())

{

int u=Q.front(); //Q এর সামনের নোড নিয়ে আমরা কাজ করবো

for(int i=0;i<G[u].size();i++)

{

int v=G[u][i];

if(!taken[v])

{

distance[v]=distance[u]+1; //সোর্স থেকে v এর দুরত্ব u এর থেকে ১ বেশি

taken[v]=1;

Q.push(v);

}

}

Q.pop();

}

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

printf("%d to %d distance %d\n",src,i,distance[i]);

}

এবার আমরা কমপ্লেক্সিটি হিসাব করি। প্রতিটা নোড একবার Q তে পুশ হচ্ছে এবং প্রতিটা edge নিয়ে আমরা একবার কাজ করছি। তাহলে bfs এর complexity হলো O(E+V) যেখানে V হলো নোড সংখ্যা এবং E হলো edge সংখ্যা।

bfs এর প্রবলেমে প্রায়ই তোমাকে path প্রিন্ট করতে বলবে। এটা খুব সহজ,একটি অ্যারেতে কোন নোডের প্যারেন্ট নোড কে সেটা সেভ করে রাখো,পরে গন্তব্য থেকে parent খুজতে খুজতে সোর্সে চলে আসো। অর্থাৎ u থেকে v তে গেলে parent[v]=u এভাবে সেভ করে রাখো।

bfs এর একটি সাধারণ প্রয়োগ হলো bipartite checking বা গ্রাফ bi-colorable কিনা চেক করা। একটি গ্রাফ bicolorable হবে যদি গ্রাফটি ২টি রঙ দিয়ে কালারিং করা যায় এবং পাশাপাশি দুটি নোডের রঙ একই না হয়। এটা খুব সহজ,একটু চিন্তা করলে নিজেই পারবে।

অ্যালগোরিদম শেখার পর প্রবলেম সলভিং তোমাকে নিজেকেই করতে হবে। কোনো অ্যালগোরিদম শেখার পর সেই অ্যালগোরিদম সংক্রান্ত ৯৫% প্রবলেমে তুমি অ্যালগোরিদমটা সরাসরি ব্যবহার করতে পারবে না। প্রবলেম অনুযায়ী মডিফাই করতে হবে। অ্যালগোরিদম শেখা সহজ কাজ,অল্প কিছুদিনে শেখা যায়,কিন্তু সেই অ্যালগোরিদম দিয়ে প্রবলেম সলভ করতে দরকার অনেক বেশি দক্ষতা,প্র্যাকটিস,চিন্তা-ভাবনা,অভিজ্ঞতা।  
তাই যারা bfs ভালোভাবে জানতে চাও তারা নিচের প্রবলমেগুলো অবশ্যই সলভ করতে চেষ্টা করবে:

[Bicoloring](http://uva.onlinejudge.org/external/100/10004.html)(uva-1004)(Bipartite checking)  
[A Node Too Far](http://uva.onlinejudge.org/external/3/336.html)(uva- 336)(Shortest path)  
[Risk](http://uva.onlinejudge.org/external/5/567.html)(uva- 567)( (Shortest path)  
[Bombs! NO they are Mines!!](http://uva.onlinejudge.org/external/106/10653.html) (uva- 10653)( (bfs in 2d grid)  
[Knight Moves](http://uva.onlinejudge.org/external/4/439.html)(uva- 439)( (bfs in 2d grid)  
[We Ship Cheap](http://uva.onlinejudge.org/external/7/762.html)(uva- 762)( (Printing path)  
[Word Transformation](http://uva.onlinejudge.org/external/4/429.html)(uva- 429)( (strings)

### ১০টি মন্তব্য

**সিয়াম**

কোন নোড এ তো লুপ নাই যদি লুপ থাকে তা হলে কি coding same থাকবে…?

**শাফায়েত**

হ্যা একই থাকবে।

[সঞ্জয় দেবনাথ](http://uhunt.felix-halim.net/id/157834)

int v=G[u][j];  
এই লাইনে ভাইয়া বুজতে পারতেছিনা । G কে তো 1D vector array হিসাবে declare করা হল কিন্তু এই লাইনে এটাকে আবার ২ডি অ্যারে হিসাবে কিভাবে লিখলেন ? একটু বুজাইয়া বললে উপকৃত হতাম ।

G[u].size() এটা কিসের সাইজ বের করবে ?

[সিয়াম](http://codingaquarium.wordpress.com/)

@সঞ্জয় দেবনাথ,  
একটু খেয়াল করেন আমরা normal vector এর বেলায় variable এর element গুলো access করি এভাবে v[i] যেখানে v একটা ভেক্টর object আর [i] হল index এখন যদি একটা vector object এর array declare করি তাহলে i তম object এর j তম variable এর access হবে v[i][j]  
আশা করি বুঝতে পারছেন না পারলে বা না বুঝলে জানাবেন  
আর G[u].size() এটা দ্বারা u তম object এর সাইজ বুঝানো হয়েছে। আশা করি বুঝতে পেরেছেন :)  
না বুঝলে জানাবেন।

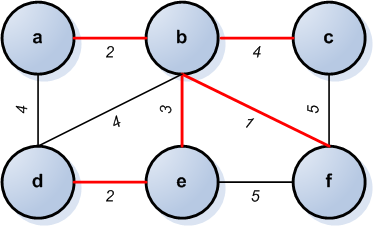
[গ্রাফ থিওরিতে হাতেখড়ি ৬: মিনিমাম স্প্যানিং ট্রি(প্রিমস অ্যালগোরিদম)](http://www.shafaetsplanet.com/planetcoding/?p=692)

মিনিমাম স্প্যানিং ট্রি গ্রাফ থিওরির খুব গুরুত্বপূর্ণ একটি অংশ। এই টিউটোরিয়ালে গ্রাফ থেকে মিনিমাম স্প্যানিং ট্রি নির্ণয়ের একটি অ্যালগোরিদম নিয়ে আলোচনা করবো।  
একটি গ্রাফ থেকে কয়েকটি নোড আর edge নিয়ে নতুন একটি গ্রাফ তৈরি করা হলে সেটাকে বলা হয় subgraph। স্প্যানিং ট্রি হলো এমন একটি সাবগ্রাফ যেটায়:

\* মূল গ্রাফের সবগুলো নোড আছে।  
\* সাবগ্রাফটি একটি ট্রি। ট্রিতে কখনো সাইকেল থাকেনা,edge থাকে n-1 টি যেখানে n হলো নোড সংখ্যা।

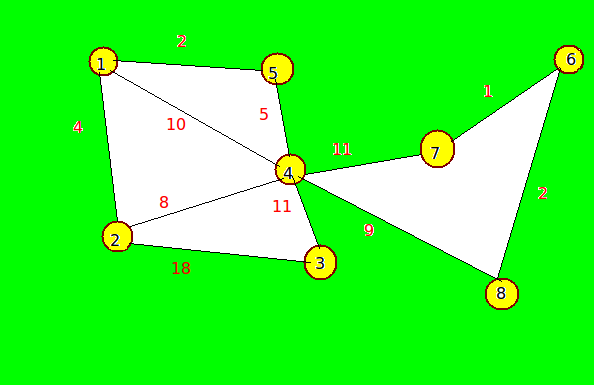
একটি গ্রাফের অনেকগুলো স্প্যানিং ট্রি থাকতে পারে,যে ট্রি এর edge গুলোর weight যোগফল সব থেকে কম সেটাই মিনিমাম স্প্যানিং ট্রি।  
(n টি নোডকে সর্বনিম্ন n-1 টি edge দিয়ে একসাথে যুক্ত করা সম্ভব,তাই n-1 টির বেশি এজ আমরা নিবোনা)

মনে করি নিচের গ্রাফের প্রতিটি নোড হলো একটি করে বাড়ি। আমাদের বাড়িগুলোর মধ্যে টেলিফোন লাইন বসাতে হবে। আমরা চাই সবথেকে কম খরচে লাইন বসাতে। edge গুলোর weight লাইন বসানোর খরচ নির্দেশ করে:



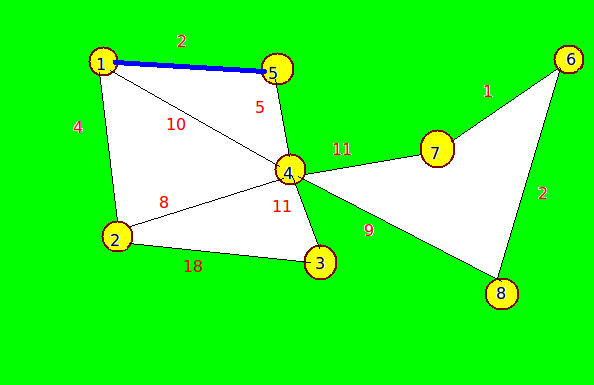
আমরা অনেক ভাবে লাইন বসাতে পারতাম। ছবিতে লাল এজ দিয়ে টেলিফোন লাইন বসানোর একটি উপায় দেখানো হয়েছে। টেলিফোন লাইনগুলো একটি সাবগ্রাফ তৈরি করেছে যেটায় অবশ্যই n-1 টি এজ আছে,কোনো সাইকেল নেই কারণ অতিরিক্ত edge বসালে আমাদের খরচ বাড়বে,কোনো লাভ হবেনা। মিনিমাম স্প্যানিং ট্রি বের করার সময় আমরা এমন ভাবে এজগুলো নিবো যেন তাদের weight এর যোগফল মিনিমাইজ হয়।

এখন নিচের গ্রাফ থেকে কিভাবে আমরা mst বের করব?

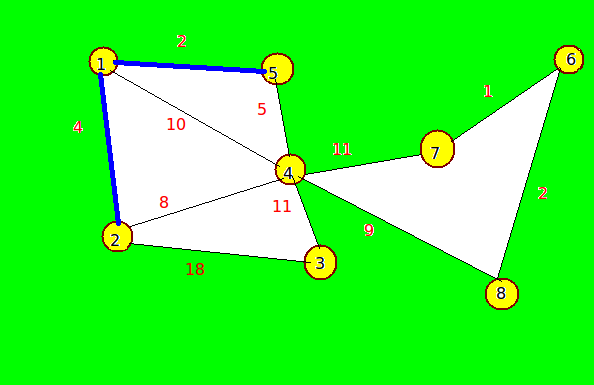


greedy অ্যাপ্রোচে খুব সহজে mst বের করা যায়। mst নির্ণয়ের প্রচলিত দুটি অ্যালগোরিদমই greedy। আমরা এখন prim’s অ্যলগোরিদম কিভাবে কাজ করে দেখব।

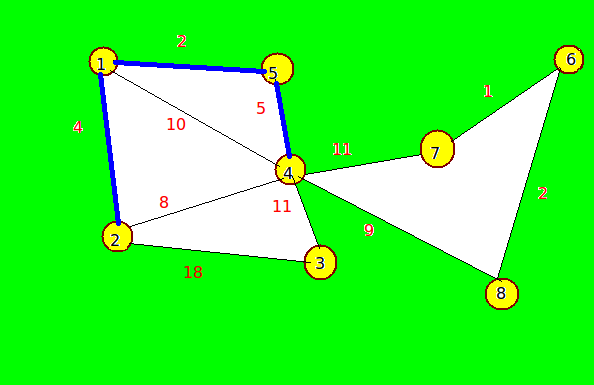
আমরা প্রথমে যেকোনো একটি সোর্স নোড নিব। ধরি সোর্স হলো ১। ১ থেকে যতগুলো এজ আছে সেগুলোর মিনিমাম টিকে আমরা সাবগ্রাফে যোগ করব। নিচের ছবিতে নীল এজ দিয়ে বুঝানো হচ্ছে এজটি সাবগ্রাফে যুক্ত করা হয়েছে:



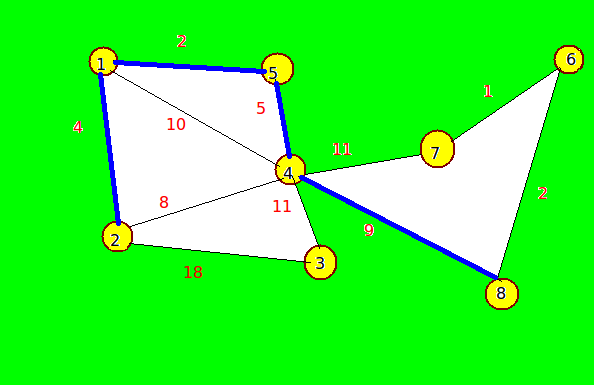
এবার সোর্স ১ এবং ৫ নম্বর নোড থেকে মোট যত এজ আছে(আগের এজগুলো সহ) তাদের মধ্যে মিনিমাম টি নিব:



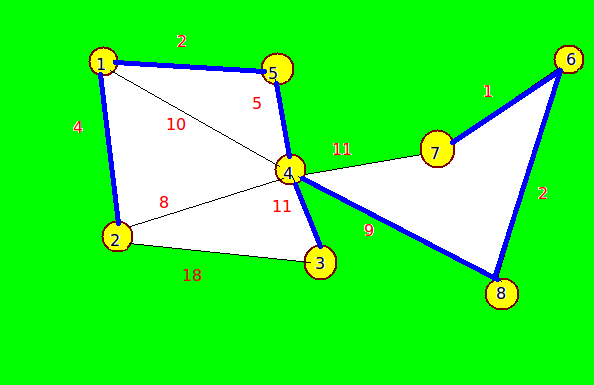
এবার নিব ১,২ এবং ৫ নম্বর নোড থেকে মোট যত এজ আছে(আগের এজগুলো সহ) তাদের মধ্যে মিনিমাম:



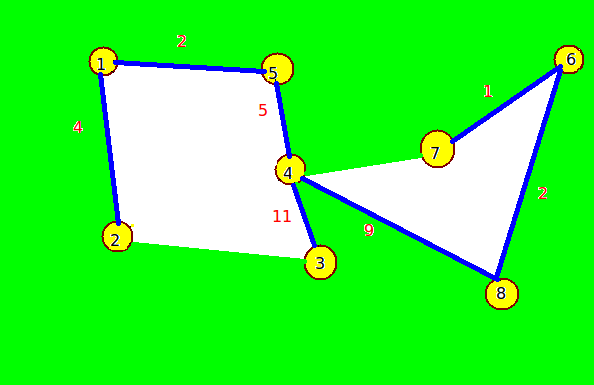
**পরের ধাপটি গুরুত্বপূর্ণ।** ১,২,৫,৪ থেকে যত এজ আছে তাদের মধ্য মিনিমাম হলো ২-৪, কিন্তু ২ নম্বর নোড এবং ৪ নম্বর নোড দুইটাই অলরেডি সাবগ্রাফের অংশ,তারা আগে থেকেই কানেক্টেড,এদের যোগ করলে সাবগ্রাফে সাইকেল তৈরি হবে,তাই ২-৪ এজটি নিয়ে আমাদের কোনো লাভ হবেনা। আ**মরা এমন প্রতিবার এজ নিব যেন নতুন আরেকটি নোড সাবগ্রাফে যুক্ত হয়।** তাহলে ৪-৮ হবে আমাদের পরের চয়েস।



এভাবে শেষ পর্যন্ত আমরা পাবো:



নীলরং এর এই সাবগ্রাফটাই আমাদের মিনিমাম স্প্যানিং ট্রি। বাকি এজগুলো মুছে দিলে থাকে:



তাহলে টেলিফোন লাইন বসাতো মোট খরচ: ৪+২+৫+১১+৯+২+১=৩৪। একটি গ্রাফে অনেকগুলো মিনিমাম স্প্যানিং ট্রি থাকতে পারে,তাই অনলাইন জাজে mst সংক্রান্ত প্রবলেম অনেক ক্ষেত্রেই special judge দিয়ে কাজ করে,অর্থাত সঠিক আউটপুটগুলোর যেকোনো একটা প্রিন্ট করলেই চলে।

আমাদের সুডোকোড হবে এরকম:

[?](http://www.shafaetsplanet.com/planetcoding/?p=692)

|  |  |
| --- | --- |
| কোড: | |
| 3  4  5  6  7  8  9  10 | \* Input: A non-empty connected weighted graph with vertices V and edges E (the weights can be negative).  \* Initialize: Vnew = {x}, where x is an arbitrary node (starting point) from V, Enew = {}  \* Repeat until Vnew = V:       o Choose an edge (u, v) with minimal weight such that u is in Vnew and v is not       (if there are multiple edges with the same weight, any of them may be picked)       o Add v to Vnew, and (u, v) to Enew  \* Output: Vnew and Enew describe a minimal spanning tree  (Wiki) |

এখন মাথায় প্রশ্ন আসতে পারে কি ভাবে prim’s mst ইম্প্লিমেন্ট করব? বারবার লুপ চালিয়ে naive অ্যাপ্রোচে কোড লিখলে তোমার কোড টাইম লিমিটের মধ্যে রান না করার সম্ভাবনাই বেশি।

আমাদের সমস্যার সমাধান হলো “priority queue”। আশা করি এই ডাটা স্ট্রাকচারটি সম্পর্কে সবাই জানো।  
প্রতিটি নোড ভিজিট করার সাথে সাথে নতুন এজগুলোকে priority queue তে ঢুকিয়ে রাখবে। priority queue তে এজগুলোকে weight অনুযায়ী সর্টেড রাখবে। তাহলে প্রতিবার queue এর top এ পাবে এখন পর্যন্ত নেয়া নোডগুলো থেকে বের হওয়া সবথেকে কম weight এর এজ,সেটাকে সাবগ্রাফে যোগ করবে।

নিজে [priority queue](http://www.cplusplus.com/reference/stl/priority_queue/) না বানিয়ে stl বা java collections ব্যবহার কর। সি++ এ [অপারেটর ওভারলোডিং](http://www.cprogramming.com/tutorial/operator_overloading.html) করে priority queue সর্ট করা সব থেকে সহজ। আমরা প্রথমে ডাটা নামক একটি structure ডিক্লেয়ার করবো:

[?](http://www.shafaetsplanet.com/planetcoding/?p=692)

|  |  |
| --- | --- |
| কোড: | |
| 3  4  5  6  7  8 | struct data  {  int u,v,cost;      bool operator < ( const data& p ) const { //overloading operator          return cost > p.cost;      }  }; |

এরপর [priority queue](http://www.cplusplus.com/reference/stl/priority_queue/) ডিক্লেয়ার করে সেটায় ‘data’ টাইপের ভ্যারিয়েবল পুশ করলে কিউ তে অটোমেটিক সর্ট হয়ে যাবে। যেহেতু এটা অ্যালগোরিদমের টিউটোরিয়াল,সি++ এর না,তাই এসব নিয়ে আর বেশি লিখবোনা,কোথায় সমস্যা হলে মন্তব্য অংশে বা আমার সাথে যোগাযোগ করে জানাতে পারো, আর তার আগে [এই লিংকটা একবার দেখো](http://comsci.liu.edu/~jrodriguez/cs631sp08/c++priorityqueue.html)।

mst নির্ণয়ের জন্য আরেকটি অ্যালগোরিদম আছে যা [kruskal’s mst](http://www.shafaetsplanet.com/planetcoding/?p=825) নামে পরিচিত যা কন্টেস্টেন্টদের মধ্যে বেশি জনপ্রিয়।[kruskal](http://www.shafaetsplanet.com/planetcoding/?p=825) শিখতে হলে তোমাকে ডিসজয়েন্ট সেট নামের একটি চমতকার ডাটা স্ট্রাকচার এর সাথে অবশ্যই পরিচিত হতে হবে। তারপর kruskal’s mst শেখা খুব সহজ হবে, prim’s শিখলেও kruskal অবশ্যই শিখতে হবে। ডিসজয়েন্ট সেট শেখার জন্য নিচের টিউটোরিয়ালগুলো দেখো:

[আমার লেখা টিউটোরিয়াল](http://www.shafaetsplanet.com/planetcoding/?p=763)  
[topcoder tutorial](http://www.topcoder.com/tc?module=Static&d1=tutorials&d2=disjointDataStructure)  
[Wikipedia](http://en.wikipedia.org/wiki/Disjoint-set_data_structure)  
Coreman’s introduction to algorithm

অ্যালগোরিদমটা ইমপ্লিমেন্ট করার পর অবশ্যই নিচের সমস্যা গুলো সমাধানের চেষ্টা করবে, আবারো বলবো প্রবলেম সল্ভ না করলে অ্যালগোরিদম শেখা অর্থহীন কারণ মাথা খাটিয়ে প্রবলেম সলভিং হলো স্কিল ডেভেলপ করার সব থেকে ভালো উপায়।

<http://uva.onlinejudge.org/external/5/544.html>(Straight forward)  
<http://uva.onlinejudge.org/external/9/908.html>  
<http://uva.onlinejudge.org/external/100/10034.html>(Straight forward)  
<http://uva.onlinejudge.org/external/112/11228.html>  
<http://uva.onlinejudge.org/external/104/10462.html>(2nd best mst)  
spoj:  
<http://www.spoj.pl/problems/MST/>(Straight forward)

Top of Form

### ১৩টি মন্তব্য

**মুন্না**

৮ ডিসেম্বর ২০১১

“priority queue ডিক্লেয়ার করে সেটায় ‘data’ টাইপের ভ্যারিয়েবল পুশ করলে Q তে অটোমেটিক সর্ট হয়ে যাবে।” …ভাইয়া এখানে একটু সমস্যা হচ্ছে ,একটু যদি ডিটেইলস লিখতেন যে কিভাবে data টাইপের stucture দিয়ে minimum priority queue তৈরি করতে হয়,খুব উপকার হত।আর আপনার পুরো কোড টা একটু দেখতে চাই…প্লিজ।  
আপনাকে অভিবাদন এত সুন্দর একটা টিউটরিয়াল লেখার জন্য।

**শাফায়েত**

কোডে সামান্য ভুল আছে,return cost < p.cost; এর জায়গায় return cost > p.cost; হবে।

আমরা ইন্টিজার,ডাবলে সহজেই +,< ইত্যাদি অপারেটর ব্যবহার করতে পারি। কিন্তু স্ট্রাকচারগুলো কাস্টম ডাটাটাইপ,এগুলোর জন্য অপারেটর নিজে ডিফাইন করে দিতে হয়।  
struct data  
{ int u,v,cost;};

ডাটা টাইপের দুটি ভ্যারিয়েবল তুমি তুলনা করবে কিভাবে? u,v বা cost এর ভিত্তিতে বা এ সবগুলোর ভিত্তিতে ছোট-বড় তুলনা করা যেতে পারে তবে তোমাকে সেটা ডিফাইন করে দিতে হবে। উপরের কোডে ঠিক এভাবেই ডিফাইন করা হয়েছে ওভারলোডিং এর মাধ্যমে,এ ক্ষেত্রে cost অনুযায়ী তুলনা করা হয়েছে। প্রায়োরিটি কিউ যখন তোমার ডাটা গুলোকে সর্ট করবে তখন সে "<" ব্যবহার করে তুলনা করবে,তাই আমরা "<" এর আচরণ ডিফাইন করে দিয়েছি।  
অনেক প্রোগ্রামার ওভারলোডিং এর বিরুদ্ধে কারণ কোড readability হারায়,এ কারণে জাভাসহ অনেক হাইলেভেল ল্যাংগুয়েজ এটি সাপোর্ট করেনা। উইকিতে একটা ছোট আর্টিকেল আছে,পুরোটা পড়ে ফেলো:  
<http://en.wikipedia.org/wiki/Operator_overloading>

**[মুন্না](http://en.wikipedia.org/wiki/Operator_overloading)**

[ধন্যবাদ উত্তর দেয়ার জন্য । আরেকটা সমস্যায় পরেছি তা হচ্ছে…](http://en.wikipedia.org/wiki/Operator_overloading)

[“পরের ধাপটি গুরুত্বপূর্ণ। ১,২,৫,৪ থেকে যত এজ আছে তাদের মধ্য মিনিমাম হলো ২-৪, কিন্তু ২ নম্বর নোড এবং ৪ নম্বর নোড দুইটাই অলরেডি সাবগ্রাফের অংশ,তারা আগে থেকেই কানেক্টেড,এদের যোগ করলে সাবগ্রাফে সাইকেল তৈরি হবে,তাই ২-৪ এজটি নিয়ে আমাদের কোনো লাভ হবেনা। আমরা এমন প্রতিবার এজ নিব যেন নতুন আরেকটি নোড সাবগ্রাফে যুক্ত হয়।”](http://en.wikipedia.org/wiki/Operator_overloading)

[আমি যে কোড টা করেছি তাতে সাইকেল তৈরী হয়ে যাচ্ছে [:(](http://en.wikipedia.org/wiki/Operator_overloading) কোন উপায় খুজে পাচ্ছিনা… একটু হেল্প চাই…প্লিজ। আর আপনার প্রিম’স অ্যালগরিদম এর কোড টা একটু দেখতে চাই,আমাকে মেইল করে দিলে খুব ভাল হয়।](http://en.wikipedia.org/wiki/Operator_overloading)

[অসংখ্য ধন্যবাদ সময় দেয়ার জন্য।](http://en.wikipedia.org/wiki/Operator_overloading)

**[শাফায়েত](http://en.wikipedia.org/wiki/Operator_overloading)**

[আমি প্রিম মাত্র একবার ইম্প্লিমেন্ট করেছি,এই টিউটোরিয়ালটা লেখার সময়,কোডটা এখন নেই। তুমি যখনই কোনো এজ নিচ্ছো তখন দুপাশের নোডদুটিকে ভিজিটেড করে দাও,যখন নতুন এজ কিউ থেকে নিবে তখন আগে চেক করবে যে এজের দুপাশের দুটি নোডই ভিজিটেড নাকি,যদি দুটিই ভিজিটেড হয় তাহলে সেই এজ নেবার কোনো দরকার নেই। না পারলে তোমার কোডটি পোস্ট করতে পারো,দেখার চেষ্টা করবো।](http://en.wikipedia.org/wiki/Operator_overloading)

**[Ahbab](http://en.wikipedia.org/wiki/Operator_overloading)**

[bool operator p.cost;](http://en.wikipedia.org/wiki/Operator_overloading)

[ভাইয়া, এই লাইন ২ টার কাজ বুজতে পারতেছি না। প্লিজ একটু ব্যাখ্যা করে দেন।](http://en.wikipedia.org/wiki/Operator_overloading)

**[শাফায়েত](http://en.wikipedia.org/wiki/Operator_overloading)**

[এটাকে অপারেটর ওভারলোডিং বলে, উপরের একটা কমেন্টে ব্যাখ্যা করেছি দেখো।](http://en.wikipedia.org/wiki/Operator_overloading)

**[Ahbab](http://en.wikipedia.org/wiki/Operator_overloading)**

[bool operator p.cost;](http://en.wikipedia.org/wiki/Operator_overloading)

[এই ২ টা লাইনে](http://en.wikipedia.org/wiki/Operator_overloading)

**[ইমতিয়াজ হাসান](http://en.wikipedia.org/wiki/Operator_overloading)**

[২৭ ডিসেম্বর ২০১২](http://en.wikipedia.org/wiki/Operator_overloading)

[আগের কমেন্টটা ডিলিট করে দিবেন প্লিজ।  
ভাইয়া priority queue নিয়েই যত্তো সমস্যা।](http://en.wikipedia.org/wiki/Operator_overloading)

[?](http://www.shafaetsplanet.com/planetcoding/?p=692)

|  |  |
| --- | --- |
| কোড: | |
| 3  4  5  6  7  8 | struct data  {  int u,v,cost;      bool operator < ( const data& p ) const { //overloading operator          return cost > p.cost;      }  }; |

নরমাল সর্ট করার সময় যেভাবে compare ফাংশান দিয়ে কাজ করি সেটা কীভাবে করবো? ওভারলোডিং করতে আমার সমস্যা হয় কীনা!

[?](http://www.shafaetsplanet.com/planetcoding/?p=692)

|  |  |
| --- | --- |
| কোড: | |
| 3  4  5  6  7  8  9  10 | struct data{      int a,b,c;  };  data arr[size];  bool comp(data a,data b){      return a.c>b.c;  }  sort(arr,arr+size,comp); |

আর ভেরিয়্যাবল ডিক্লেয়ার করবো কীভাবে?

[?](http://www.shafaetsplanet.com/planetcoding/?p=692)

|  |  |
| --- | --- |
| কোড: | |
| 3 | priority\_queue < data > pqi; |

compare function এর কী ব্যাবস্থা করবো?

[?](http://www.shafaetsplanet.com/planetcoding/?p=692)

|  |  |
| --- | --- |
| কোড: | |
| 3 | priority\_queue < data ,comp > pqi;  ??? |

আর পুশ কি এইভাবে করতে হবে ভেক্টরের নিয়মে?

[?](http://www.shafaetsplanet.com/planetcoding/?p=692)

|  |  |
| --- | --- |
| কোড: | |
| 3  4  5  6  7  8  9  10  11  12 | struct TT{      int a,b;  };  vector v;  for(i=0;i<5;i++){      TT t;      t.a=10\*i;      t.b=10+i;      v.push\_back(t);  } |

**শাফায়েত**

[তুমি এই](http://en.wikipedia.org/wiki/Operator_overloading)[লিংকটা দেখো](http://abuasifkhan.blogspot.com/2012/05/stl-max-priorityqueue-min-priorityqueue.html), আমার মনে হয় এটা দেখার পরে তোমার সমস্যা মিটে যাবে, এরপরেও সমস্যা থাকলে জানাও।

**ইমতিয়াজ হাসান**

ধন্যবাদ ভাইয়া।

[অনিন্দ্য সুন্দর পাল](http://binaryrongo.wordpress.com/)

“যেহেতু এটা অ্যালগোরিদমের টিউটোরিয়াল,সি++ এর না,তাই এসব নিয়ে আর বেশি লিখবোনা,কোথায় সমস্যা হলে মন্তব্য অংশে বা আমার সাথে যোগাযোগ করে জানাতে পারো, আর তার আগে [এই লিংকটা একবার দেখো।](http://abuasifkhan.blogspot.com/2012/05/stl-max-priorityqueue-min-priorityqueue.html)”  
ভাইয়া, এই লিংকটা কাজ করছে না।

**শাফায়েত**

লিংকটা সরিয়ে ফেলা হয়েছে। অন্য একটা লিংক বসিয়ে দিলাম।

**মুহিম মুক্তাদির**

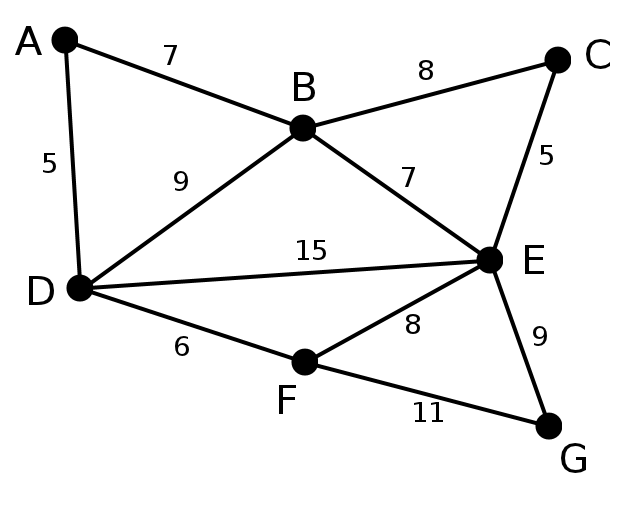
ভাইয়া, আমি ইউভিএ এর ৫৪৪ নাম্বার প্রব্লেম টা implement করেছি কিন্তু verdict- TLE . নিচে আমার কোড, দয়া করে সাহায্য করলে উপক্রিত হতাম ঃ  
<http://ideone.com/JRDg5A>

[গ্রাফ থিওরিতে হাতেখড়ি ৭: মিনিমাম স্প্যানিং ট্রি(ক্রুসকাল অ্যালগোরিদম)](http://www.shafaetsplanet.com/planetcoding/?p=825)

[আগের পোস্টে](http://www.shafaetsplanet.com/planetcoding/?p=692) আমরা প্রিমস অ্যালগোরিদম ব্যবহার করে [mst](http://www.shafaetsplanet.com/planetcoding/?p=692) নির্ণয় করা দেখেছি। mst কাকে বলে সেটাও আগের পোস্টে বলা হয়েছে। এ পোস্টে আমরা দেখবো mst বের করার আরেকটি অ্যালগোরিদম যা ক্রুসকালের অ্যালগোরিদম নামে পরিচিত। এটি mst নির্ণয়ের সবথেকে সহজ অ্যালগোরিদম। তবে তোমাকে অবশ্যই ডিসজয়েন্ট সেট ডাটা স্ট্রাকচার সম্পর্কে জানতে হবে,না জানলে [এই পোস্টটি](http://www.shafaetsplanet.com/planetcoding/?p=763) অবশ্যই দেখে আসো।

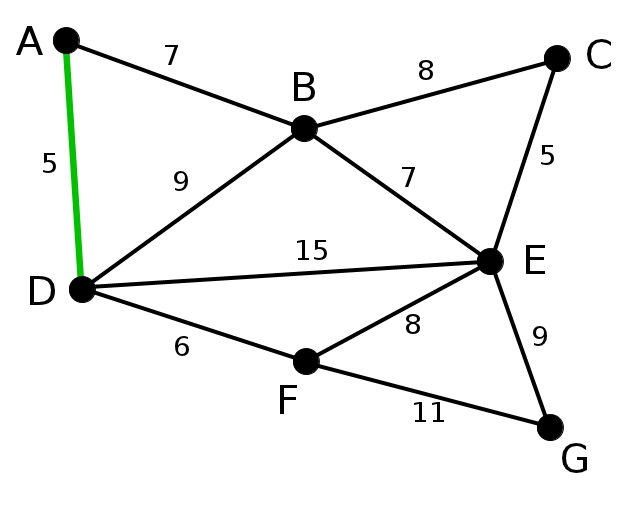
এই পোস্টে নিজের আকা ছবি ব্যবহার করবোনা। উইকিতে ক্রুসকাল নিয়ে খুব সুন্দর করে লেখা আছে,আমি ওখানকার ছবিগুলোই ব্যবহার করে সংক্ষেপে অ্যালগোটা বুঝানোর চেষ্টা করবো।

নিচের গ্রাফটি দেখো:

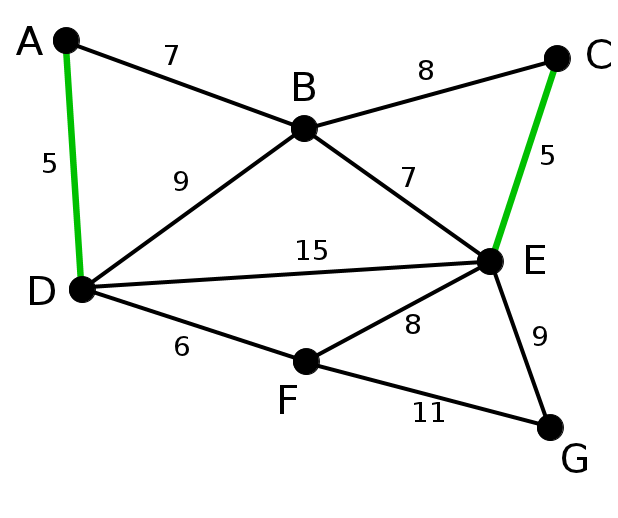


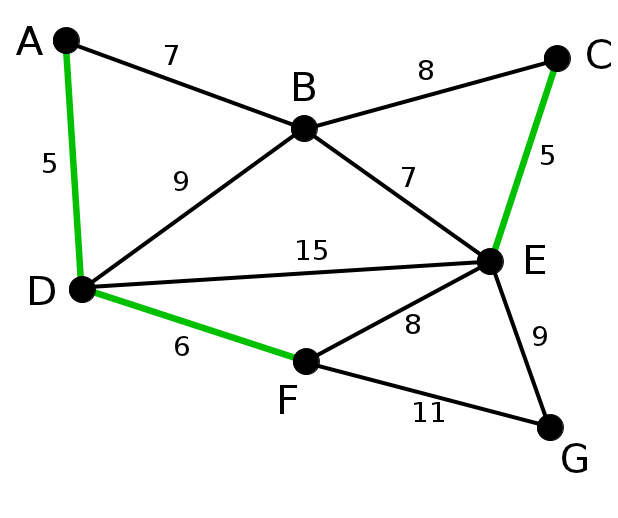
প্রথমে আমাদের ট্রিতে একটি এজও নেই। আমরা মুল গ্রাফের এজগুলোকে cost অনুযায়ী সর্ট করে ফেলবো। সব থেকে কম cost এর এজ আগে নিবো,বেশি cost এর এজ পরে নিবো। দুটি এজের cost সমান হলে যেকোনো একটি আগে নিতে পারি। তারপর একটি করে এজ নিবো আর দেখবো এজের দু প্রান্তের নোডগুলোর মধ্যে ইতোমধ্যে কোনো পথ আছে নাকি,যদি থাকে তাহলে এজটি নিলে সাইকেল তৈরি হবে,তাই এজটা আমরা নিবোনা। বুঝতেই পারছো প্রিমসের মত এটিও একটি greedy অ্যালগোরিদম।

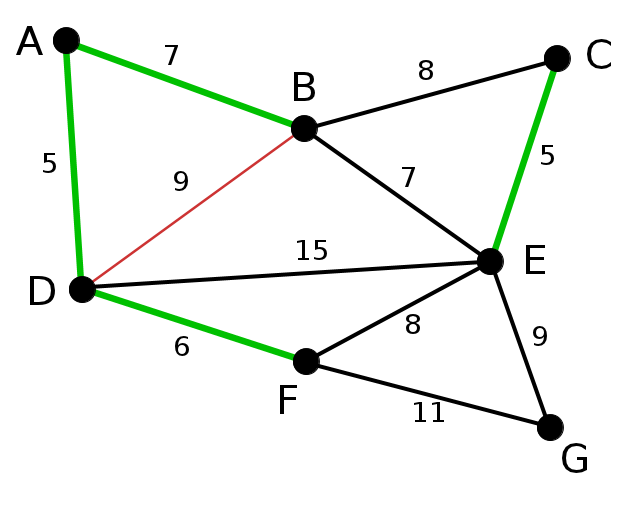
উপরে AD আর CE হলো সবথেকে কম cost এর এজ। আমরা AD কে সাবগ্রাফের অন্তর্ভূক্ত করলাম।

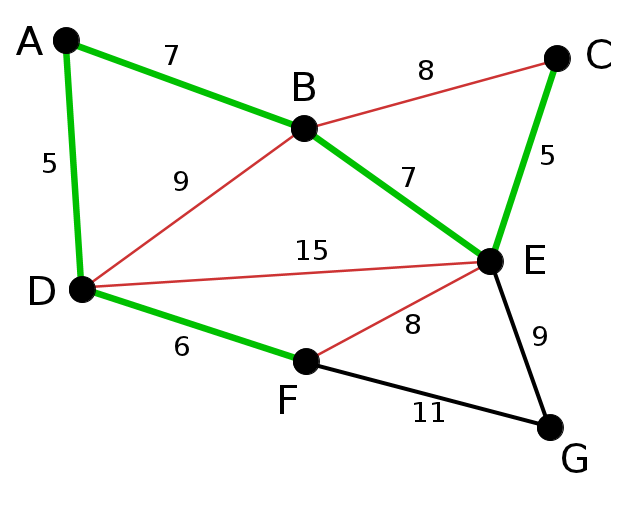


একই ভাবে এরপ CE তারপর DF,AB এবং BE কে যুক্ত করবো:



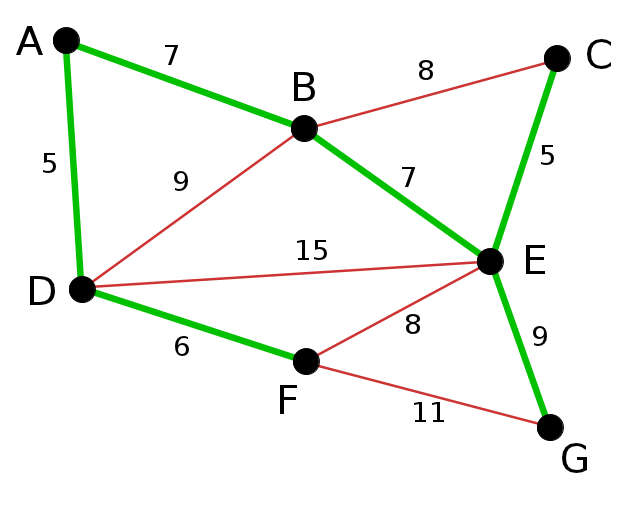






এরপর সবথেকে ছেোট এজ হলো EF, এটাকে আমরা নিতে পারবোনা কারণ EF নিলে একটি সাইকেল তৈরি হয়ে যাবে,E থেকে F তে যাবার রাস্তা আগে থেকেই আছে,তাই এজটি নেয়ার কোনো দরকার নেই। এভাবে BC,DB সহ লাল রঙের এজগুলো বাদ পড়বে কারণ এরা সাইকেল তৈরি করে।

সবশেষে EG যোগ করলে আমরা mst পেয়ে যাবো।



এখন আমরা ইম্প্লিমেনটেশনে আসি। আমাদের প্রথম কাজ হলো সর্ট করা। পরের কাজ হলো একটি একটি এজ নিয়ে চেক করা যে দু প্রান্তের নোড দুটির মধ্য পথ আছে নাকি,অর্থাত তারা একই কম্পোনেন্টের ভিতর আছে নাকি। এটা চেক করতে লাগবে ডিসজয়েন্ট সেট। ডিসজয়েন্ট সেট নিয়ে[টিউটোরিয়ালে](http://www.shafaetsplanet.com/planetcoding/?p=763) দেখিয়েছিলাম কিভাবে দুটি নোড একই সাবগ্রাফে আছে নাকি বের করতে হয়। তুমি সেই কাজটিই এখানে করবে। তারপর একই সাবগ্রাফে না থাকলে আগের মত Union ফাংশন কল দিয়ে তাদের একসাথে নিয়ে আসবে আর এজটি একটি ভেক্টর বা অ্যারেতে সেভ করে রাখবে।

নিচে একটা ইমপ্লিমেন্টেশন দিলাম, আশা করি এটা কপি না করে নিজে বুঝে লিখবে:

[?](http://www.shafaetsplanet.com/planetcoding/?p=825)

|  |  |
| --- | --- |
| কোড: | |
| 3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | struct edge  {      int u,v,w;      bool operator < ( const edge& p ) const      {          return w < p.w;      }  };  int pr[MAXN];  vector<edge>e;  int find(int r)  {     return (pr[r]==r) ? r:  find(pr[r]);  }  int mst(int n)  {      sort(e.begin(),e.end());      for(int i=1;i<=n;i++)pr[i]=i;        int count=0,s=0;      for(int i=0;i<(int)e.size();i++)      {          int u=find(e[i].u);          int v=find(e[i].v);          if(u!=v)          {              pr[u]=v;              count++;              s+=e[i].w;              if(count==n-1) break;          }      }      return s;  }    int main(){  //  READ("in");      int n,m;      cin>>n>>m;      for(int i=1;i<=m;i++)      {          int u,v,w;          cin>>u>>v>>w;          edge get;          get.u=u; get.v=v; get.w=w;          e.push\_back(get);      }      cout<<mst(n)<<endl;      return 0;    } |

**কমপ্লেক্সিটি অ্যানালাইসিস:**  
মনে করি m হলো এজ সংখ্যা। এজগুলোকে সর্ট করতে হবে, সেটার কমপ্লেক্সিটি mlogm, এরপরে শুধু এজগুলোর উপর লিনিয়ার লুপ চালাতে হবে। তাহলে মোট কমপ্লেক্সিটি O(mlogm)।

[mst](http://www.shafaetsplanet.com/planetcoding/?p=692) সম্পর্কিত অনেকগুলো সহজ প্রবলেম দিয়েছি প্রিমস এর টিউটোরিয়ালে,ওগুলো সলভ করে প্র্যাকটিস করতে পারো। আরেকটু ভালো প্রবলেম করতে চাইলে দেখো:

<http://uva.onlinejudge.org/external/103/10369.html>  
<http://uva.onlinejudge.org/external/117/11733.html>

Top of Form

**Ronok1307**

Great tutorial. But I have one problem. In your Union function you say that it’s ok to use either **par[u]=v** or **par[v]=u**. But for an ACM problem, specifically problem 544, while using Disjoint Set for Kruskal, I am getting WA with one and AC for the other. What could be the problem?

**শাফায়েত**

আমি ঠিক নিশ্চিত না সমস্যা কোথায়। যেকোনো একটা লিখলেই কাজ করার কথা। আমি এইমাত্র [৯০৮](http://uva.onlinejudge.org/external/9/908.html)নম্বরের কোড ট্রাই করে দেখলাম, দুটাতেই accepted হয়েছে। কোডটা [দিয়ে দিলাম এখানে](http://paste.ubuntu.com/1040671/)।

**হাসান**

একই ভাবে এরপ CE তারপর DF,AB কে যুক্ত করবো:  
এরপর সবথেকে ছেোট এজ হলো BD, এটাকে আমরা নিতে পারবোনা কারণ BD নিলে একটি সাইকেল তৈরি হয়ে,B থেকে D তে যাবার রাস্তা আগে থেকেই আছে,তাই এজটি নেয়ার কোনো দরকার নেই।

AB এর পরতো সবচেয়ে ছোট edge তো BE, তাহলে BD নিয়ে কাজ করবো কেনো?

**শাফায়েত**

একটু টাইপিং মিসটেক আছে, আমি ঠিক করে দিচ্ছি।

[Kfoozminus](https://sites.google.com/site/mathprogrammingbooks/)

২১ জানুয়ারি ২০১৩

ভাইয়া একটা ব্যাপার… 29 নম্বর লাইনে pr[u]=v না লিখে pr[u]=pr[v] লিখলে find() এর কাজটা আরো কমে যেত না?

[গ্রাফ থিওরিতে হাতেখড়ি ৮:টপোলোজিকাল সর্ট](http://www.shafaetsplanet.com/planetcoding/?p=848)

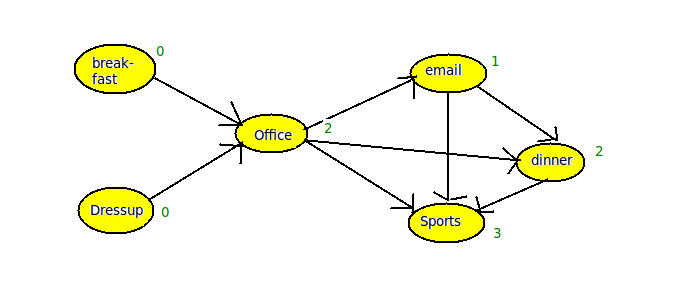
মনে কর তোমার হাতে কিছু কাজের একটা তালিকা আছে,কাজগুলো অবশ্যই শেষ করতে হবে। কাজগুলো হলো অফিসে যাওয়া,সকালে নাস্তা করা,টিভিতে খেলা দেখা,কিছু ই-মেইলের উত্তর দেয়া ,বন্ধুদের সাথে ডিনার করা ইত্যাদি। কাজগুলো কিন্তু আপনি যেকোনো অর্ডারে করতে পারবেনা,কিছু শর্ত মানতে হবে। যেমন অফিসে যাবার আগে নাস্তা করতে হবে,খেলা দেখার আগে অফিসে যেতে হবে,ডিনারে বসার আগে ইমেইলের উত্তর দিতে হবে।  
তুমি শর্তগুলোর তালিকা করে ফেললে:

১. সকালের নাস্তা —> অফিস (ক—>খ এর মানে হলো ‘খ’ কাজটি করার আগে ‘ক’ কাজটি করতে হবে)  
২. সুট-টাই পড়া —-> অফিস  
৩. অফিস —-> ইমেইল  
৪. অফিস —-> ডিনার  
৫. অফিস —-> খেলা  
৬. ইমেইল —> ডিনার  
৭. ইমেইল —> খেলা  
৮. ডিনার —> খেলা

তুমি এখন কোন কাজ কখন করবে? উল্টাপাল্টা অর্ডারে করলে আপনার কাজ ভন্ডুল হয়ে যাবে,ইমেইল না করে খেলা দেখতে বসলে তুমি ক্লায়েন্ট হারাবে,তাই অর্ডারিং খুব জরুরি।

এটা একটি “টাস্ক শিডিউলিং” প্রবলেম। কোন কাজের পর কোন কাজ করতে হবে সেটা আমাদের বের করতে হবে। অর্থাত এটা এক ধরণের সর্টিং যাকে টপোলোজিকাল সর্টিং বলে। আমরা এ টিউটোরিয়ালে এজ সরিয়ে বা indegree কমিয়ে টপসর্ট বের করবো।

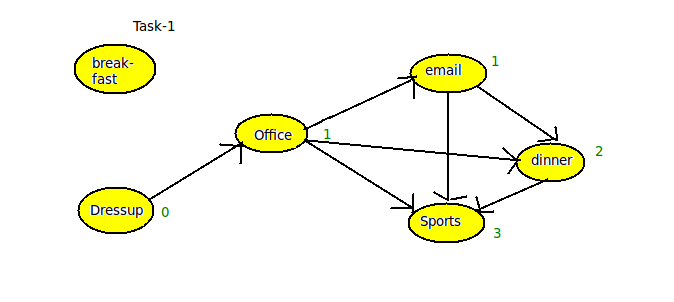
আমরা প্রথমেই সমস্যাটাকে নিচের গ্রাফ দিয়ে মডেলিং করবো:



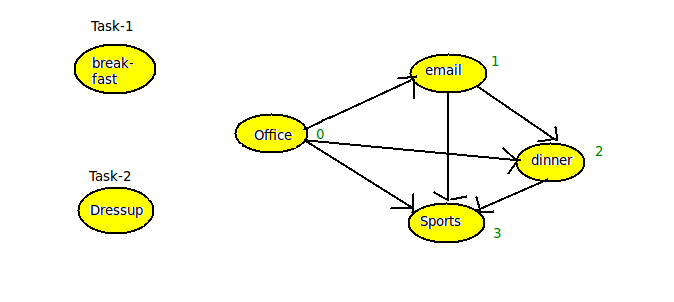
উপরের ছবিতে প্রতিটা কাজ একটি করে নোড দিয়ে দেখানো হয়েছে। ব্রেকফাস্ট থেকে অফিসের দিকে তীরচিহ্ন দিয়ে বুঝানো হচ্ছে যে অফিসে আসার আগে ব্রেকফাস্ট করতে হবে। উপরের ৮টি শর্ত ছবিতে ৮টি ডিরেক্টেড এজ দিয়ে দেখানো হয়েছে।

প্রতিটি নোডের পাশে ছোট করে কিছু সংখ্যা দেখতে পাচ্ছো। যেমন অফিসের সাথে ০,ডিনারের সাথে ২ ইত্যাদি। এগুলো দিয়ে বুঝাচ্ছে একটি কাজ অন্য কয়টি কাজের উপর নির্ভরশীল। যেমন ডিনারের আগে তোমাকে অফিস,ইমেইল এই ২টা কাজ করতে হবে,ডিনার নোডটিতে ২টি তীরচিহ্ন প্রবেশ করেছে,আর আমরা পাশে লিখে দিয়েছি “২”। ঠিক এভাবে ইমেইলের পাশে লেখা হয়েছে ১। এ সংখ্যাগুলোকে indegree বলা হয়।

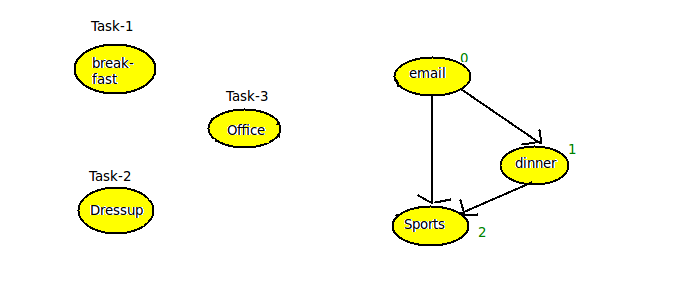
লক্ষ্য কর ব্রেকফাস্ট এবং ড্রেসআপ কোনো কাজের উপর নির্ভরশীল নয়,তাই তাদের পাশে ০ লেখা হয়েছে। তারমানে আমরা এ দুটি কাজের**যেকোনোটা** দিয়ে দিন শুরু করতে পারি। মনে করি তুমি নাস্তা আগে খেতে চাও। নাস্তা খেয়ে নেবার পর যেসব কাজ ব্রেকফাস্টের উপর নির্ভরশীল ছিল তারা আর সেটার উপর নির্ভরশীল থাকলোনা,গ্রাফটা হয়ে গেল এরকম:



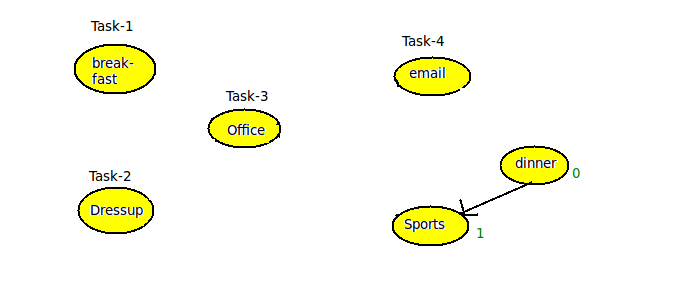
ব্রেকফাস্ট থেকে অফিসের তীরচিহ্ন সরিয়ে দিয়েছি। এখন অফিস আর মাত্র ১টি কাজের উপর নির্ভরশীল(আগে ছিল ২টির উপর)। এবার তোমাকে ড্রেসআপ করতে হবে,কারণ এখন একমাত্র এই কাজটিই কারো উপর নির্ভরশীল নয়:



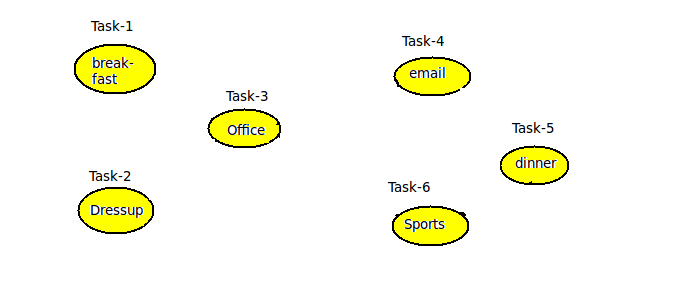
ড্রেসআপ থেকে অফিসের তীরচিহ্ন সরিয়ে দেয়া হয়েছে,আর কোনো কাজ নেই,এবার তুমি অফিসে যাবার জন্য প্রস্তুত। অফিসে যাবার উপর যারা নির্ভরশীল তাদের তীরচিহ্নগুলো এখন সরিয়ে দিতে পারি:



এখন ইমেইল “নোড” এর নির্ভরশীলতা ০ হয়ে গিয়েছে:



সবশেষে ডিনার করে খেলা দেখতে বসা:



এখন তুমি কাজের অর্ডারিং পেয়ে গিয়েছো,নাস্তা করা,অফিসের পোষাক পড়া,অফিসে যাওয়া,ইমেইল করা,ডিনার করা,খেলা দেখা।

এটাকেই বলা হয় টপোলোজিক্যাল সর্ট(topological sort) বা টপসর্ট। মূলত কাজের অর্ডারিং খুজে বের করতে এই অ্যালগোরিদমটি ব্যবহার করা হয়। কম্পিউটার তার অভ্যন্তরে বিভিন্ন কাজের অর্ডারিং ঠিক করতে টপসর্ট ব্যবহার করে। অনেক রিয়েল লাইফ প্রয়োগ থাকায় টপসর্ট কম্পিউটার সায়েন্সে খুবই গুরুত্বপূর্ণ একটি টপিক। টপসর্ট বের করার আরেকটি পদ্ধতি আছে যার নাম “ডেপথ ফার্স্ট সার্চ”,এটা নিয়ে [আলোচনা করেছি এই লেখায়](http://www.shafaetsplanet.com/planetcoding/?p=973)।

উপরের অ্যালগোরিদমটি O(n^2) এ কাজ করে। একটি গ্রাফের অনেকগুলো সর্টেড অর্ডার থাকতে পারে,যেমন উপরের সমস্যায় তুমি নাস্তা খাবার আগে জামা-কাপড় পরতে পারতে। তোমাকে লেক্সিকোগ্রাফিকালি ছোটটা প্রিন্ট করতে বলতে পারে অথবা যেটা ইনপুটে আগে আছে সেটাকে আগে প্রিন্ট করতে বলতে পারে,আশা করি এসব কন্ডিশন সহজে হ্যান্ডল করতে পারবে।

ইমপ্লিমেন্টেশন নিয়ে তেমন কিছু বলার নেই। সবগুলো নোডের indegree বের করবে। তারপর যার indegree শূন্য তার সাথে যাদের এজ আছে তাদের indegree ১ কমিয়ে দিবে,তারপর আবার খুজবে কার indegree এখন শূন্য। এক নোডকে কখনো ২বার নিবেনা। ছবিতে এজ উঠিয়ে দেখিয়েছি বুঝানোর জন্য,তোমার ম্যাট্রিক্স থেকে এজ উঠানোর দরকার নেই,indegree কমালেই চলবে।

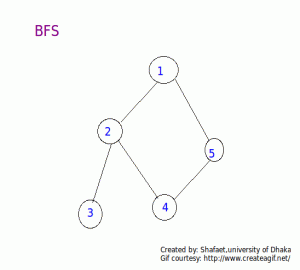
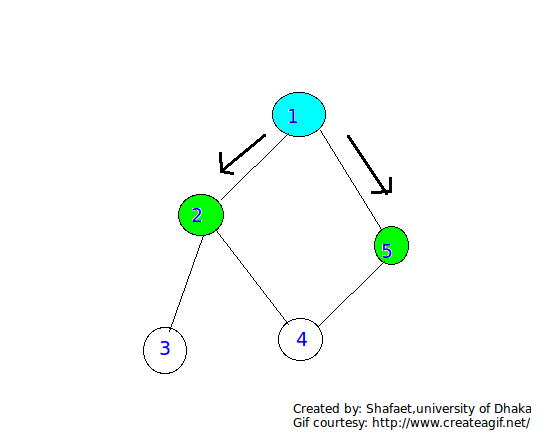
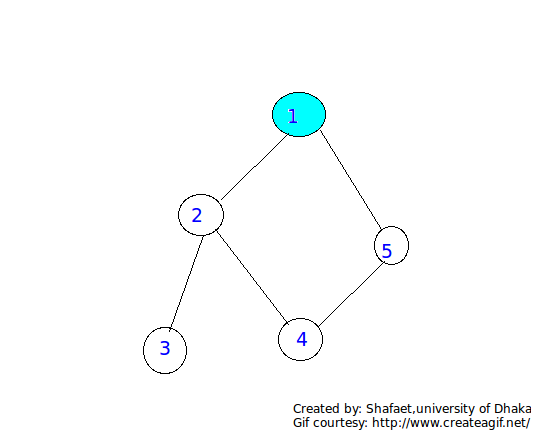
অনেক সময় বলতে পারে যতরকম ভাবে topsort করা যায় সবগুলো বের করতে। তখন তোমাকে [backtracking](http://www.shafaetsplanet.com/planetcoding/?p=1266) এর সাহায্য নিতে হবে।

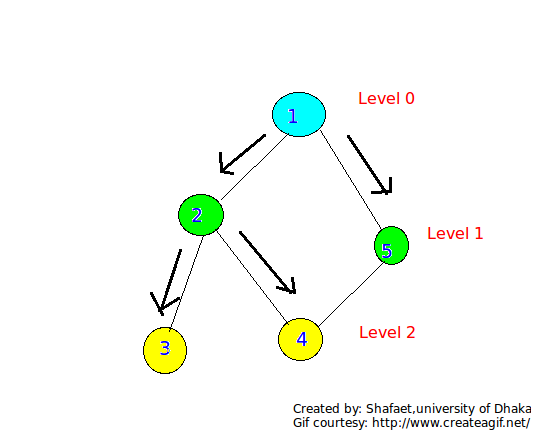
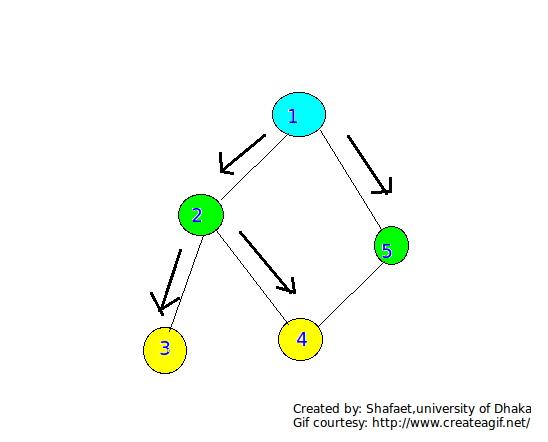
নিচের সমস্যাগুলো সমাধান করে ফেলো:  
<http://uva.onlinejudge.org/external/103/10305.html>(Easy,straight-forward,special judge)  
<http://uva.onlinejudge.org/external/110/11060.html>(Easy)  
<http://uva.onlinejudge.org/external/1/124.html>(Medium,All possible topsort)  
<http://uva.onlinejudge.org/external/4/452.html>(Medium)

[গ্রাফ থিওরিতে হাতেখড়ি ৯: ডেপথ ফার্স্ট সার্চ এবং আবারো টপোলোজিকাল সর্ট](http://www.shafaetsplanet.com/planetcoding/?p=973)

বেশ কিছুদিন পর [সিরিজের](http://www.shafaetsplanet.com/planetcoding/?cat=35)পরের পর্ব লিখতে বসলাম। গ্রাফের টিউটোরিয়াল লিখতে অনেক ছবি আকতে হয় দেখে অনেক সময় চলে যায়। এখন থেকে স্থির ছবি ব্যবহার না করে gif অ্যানিমেশন ব্যবহার করবো বলে ঠিক করেছি,এতে বুঝাতে সুবিধা হবে,অনেক কম কথা বলতে হবে। এটা এই**সিরিজের প্রথম অ্যানিমেটেড টিউটোরিয়াল**।(তবে তুমি যদি লেখার উপরের আইকনে চাপ দিয়ে pdf হিসাবে ডাউনলোড করো তাহলে অ্যানিমেশন কাজ করবেনা)।

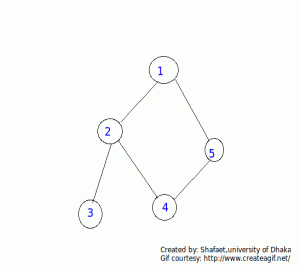
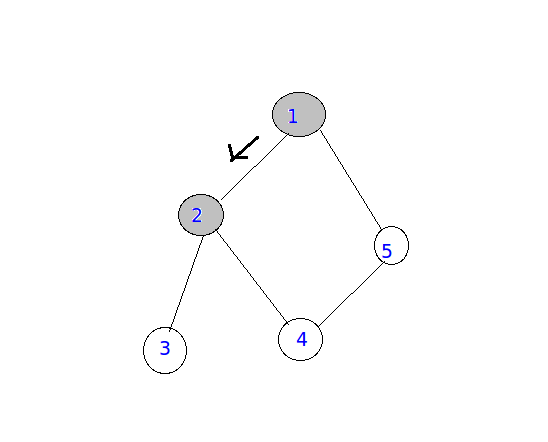
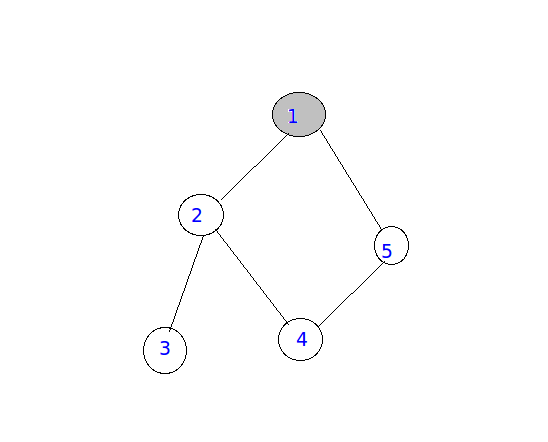
আগের পর্বগুলো পড়ে থাকলে হয়তো ডেপথ ফার্স্ট সার্চ বা dfs এতদিনে নিজেই শিখে ফেলেছো। তারপরেও এই টিউটোরিয়ালটি পড়া দরকার কিছু কনসেপ্ট জানতে।

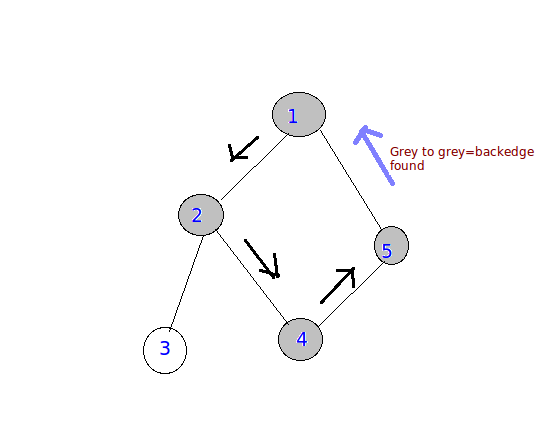
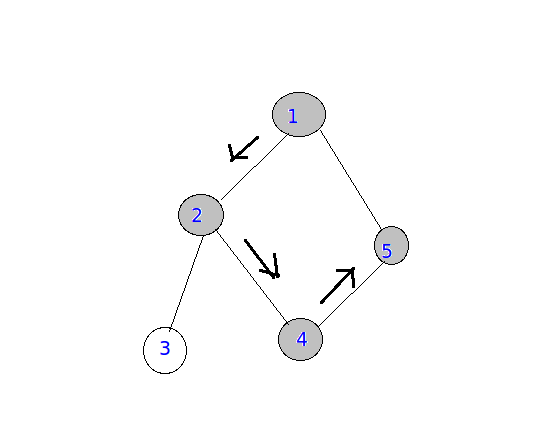
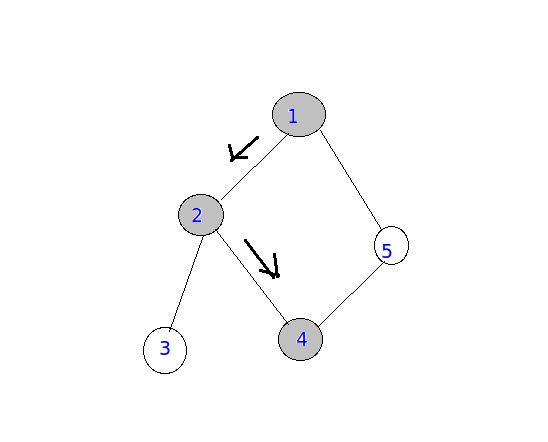
bfs এ আমরা গ্রাফটাকে লেভেল বাই লেভেল সার্চ করেছিলাম,নিচের অ্যানিমেশনের মতো করে:  
[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2012/03/bfs_anim12.gif)

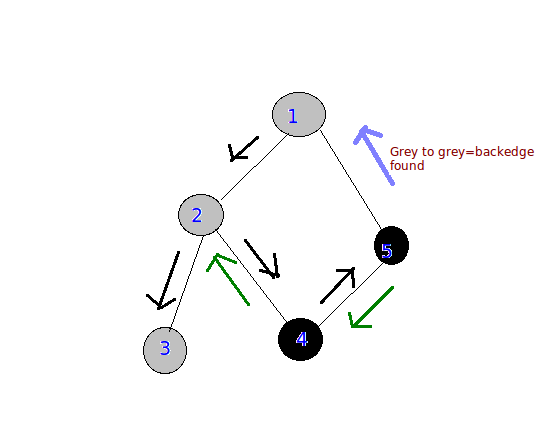
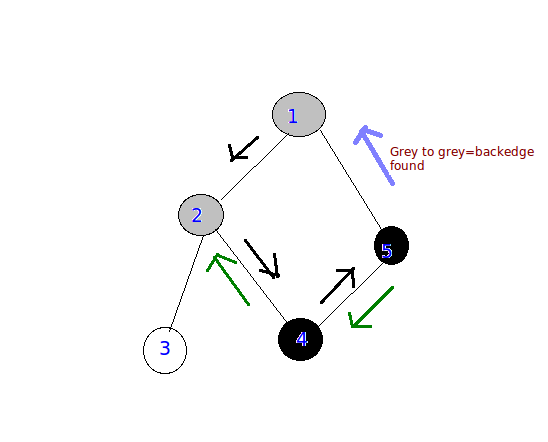
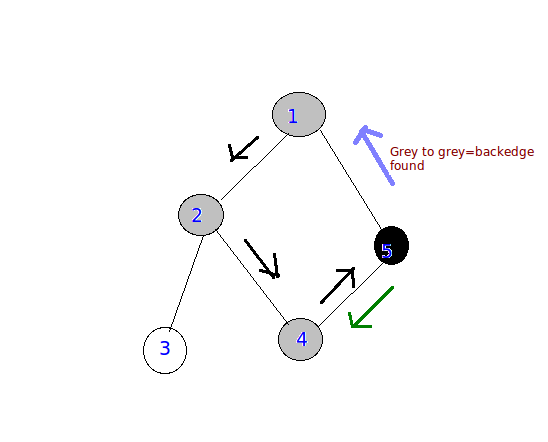
  
(যদি অ্যানিমেশন দেখতে সমস্যা হয় তাহলে সবগুলো ছবি ডাউনলোড করতে পারো [এখান থেকে](http://shafaetsplanet.com/uploads/dfstutorialpic/dfstutorialpic.zip) )

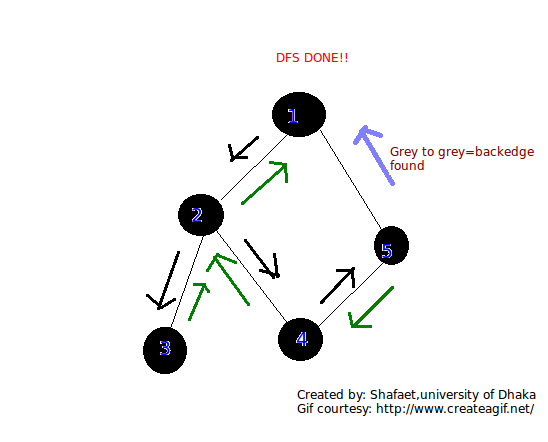
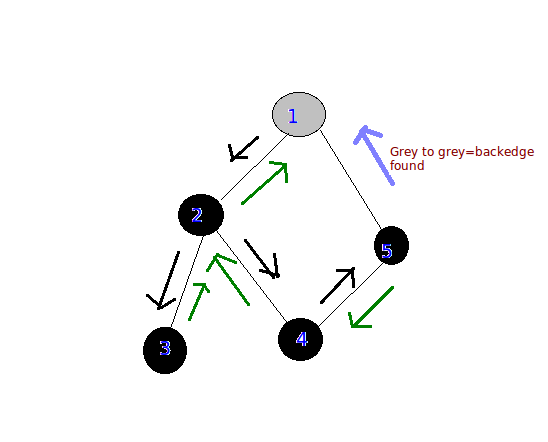
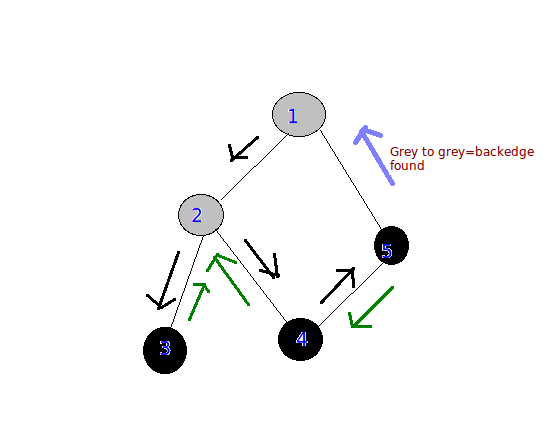
এবার আমরা কোনো নোড পেলে সাথে সাথে সে নোড থেকে আরো গভীরে চলে যেতে থাকবো,যখন আর গভীরে যাওয়া যাবেনা তখন আবার আগের নোডে ফিরে এসে অন্য আরেক দিকে যেত চেষ্টা করবো,এক নোড কখনো ২বার ভিজিট করবোনা। আমরা নোডের ৩টি রং(কালার) দিবো:

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 অ্যানিমেশন দেখতে পারি:  
[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2012/03/dfs_anim1.gif)

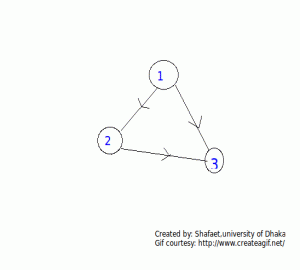
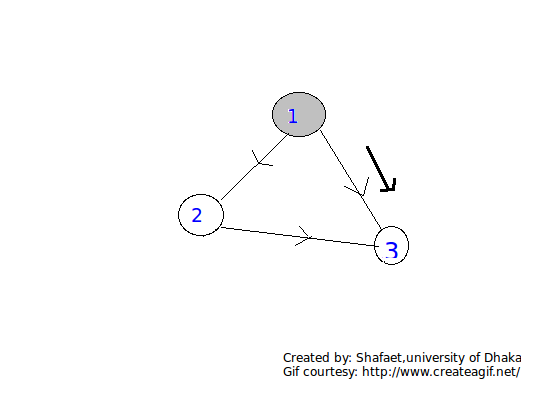
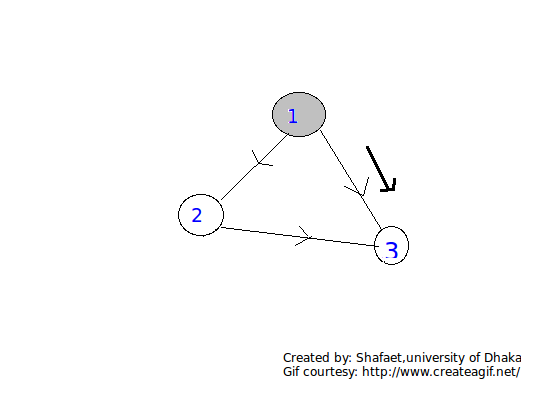


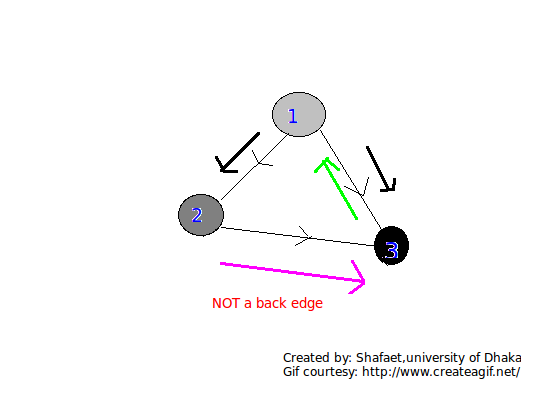
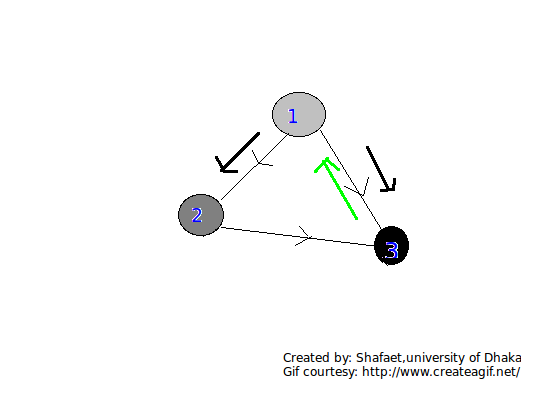
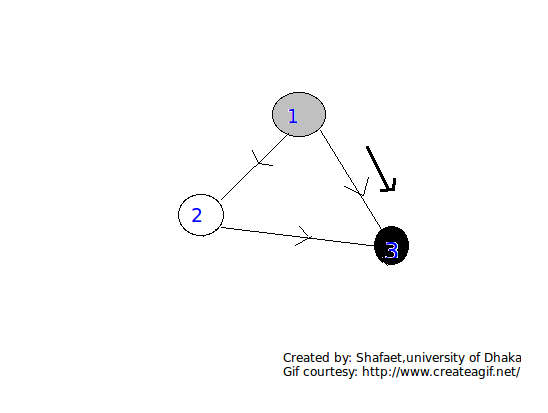


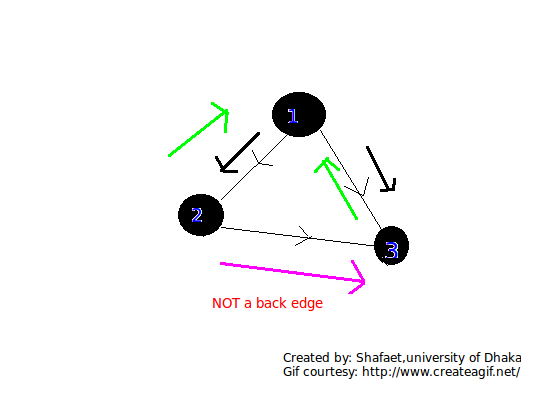
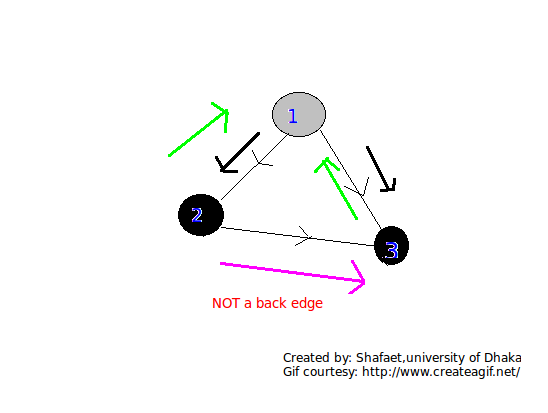


আশা করি ডিএফএস কিভাবে কাজ করে এটা পরিস্কার,খুব সহজ জিনিস এটা। এবার আমরা একটা খুব গুরুত্বপূর্ণ টার্ম শিখবো,সেটা হলো ব্যাকএজ(backedge)। অ্যানিমেশনে লক্ষ করো ৫-১ কে ব্যাকএজ বলা হয়েছে। এর কারণ হলো তখনও ১ এর কাজ চলছে,৫ থেকে ১ এ যাওয়া মানে এমন একটা নোড ফিরে যাওয়া যাকে নিয়ে কাজ এখনো শেষ হয়নি,তারমানে অবশ্যই 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 adjacency list 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**।

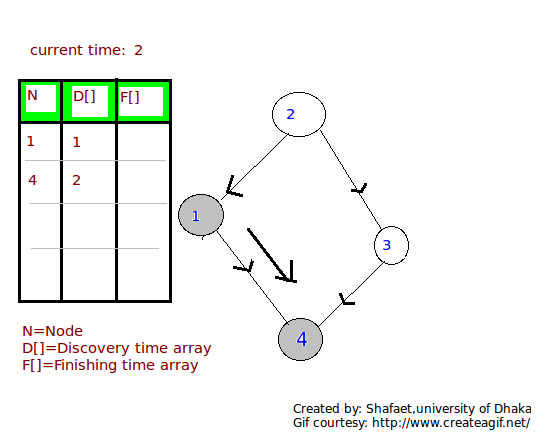
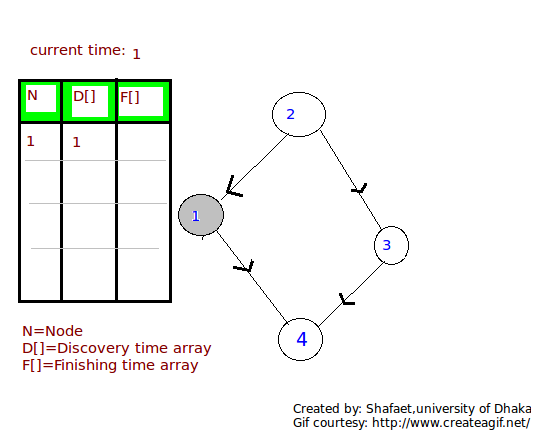
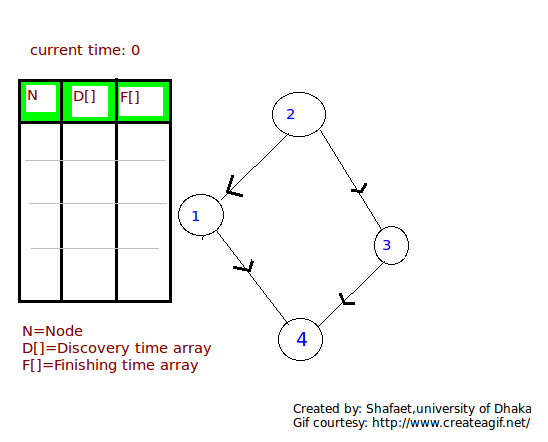
আনডিরেক্টেড গ্রাফের ক্ষেত্রে আগে ভিজিট করা কোনো নোডে ফিরে গেলেই সেটা ব্যকএজ,কালার চেক না করলেও হয়। তবে ডিরেক্টেড গ্রাফের ক্ষেত্রে অবশ্যই করতে হবে। আরেকটি ছোট অ্যানিমেশন:  
[](http://www.shafaetsplanet.com/planetcoding/wp-content/uploads/2012/03/dfs2.gif)

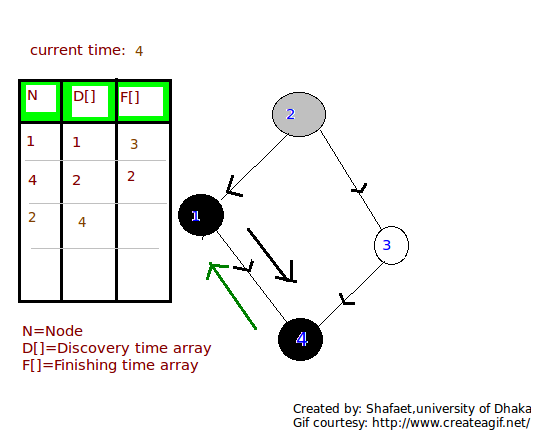
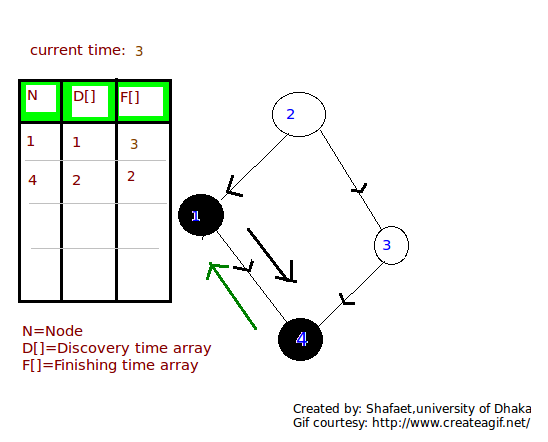
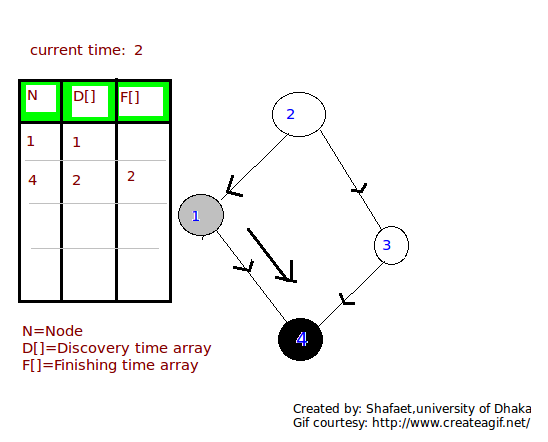


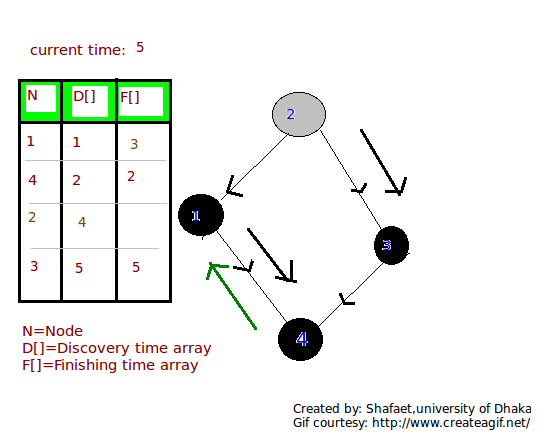
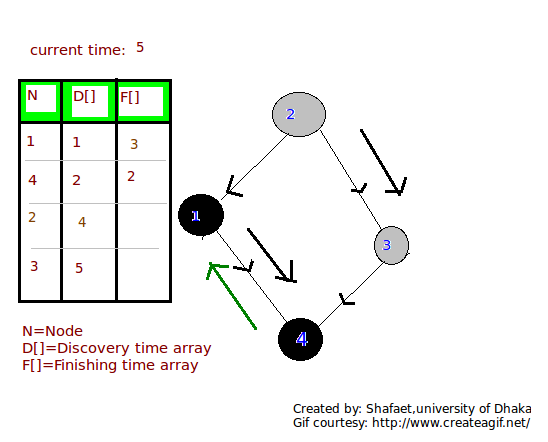
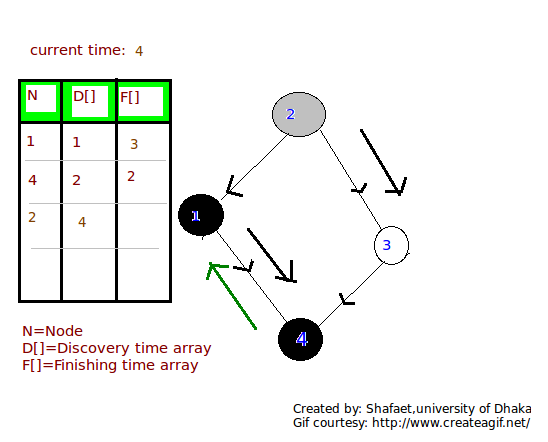


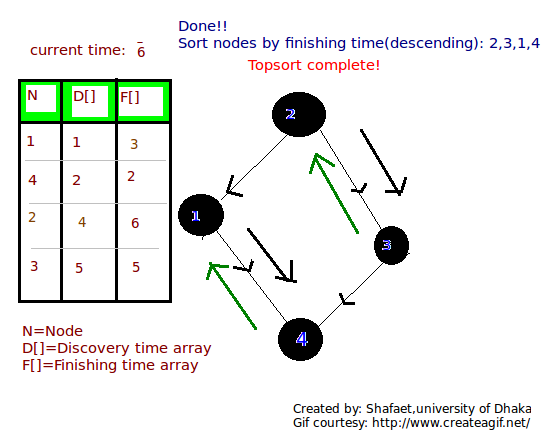
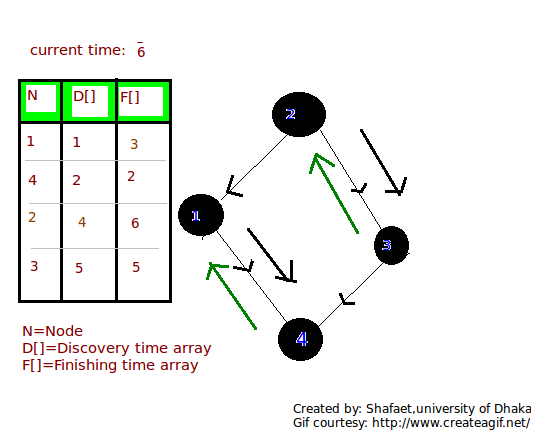
২-৩এর এজটাকে ব্যাকএজ বলা যাচ্ছেনা,কারণ ৩ এর কাজ আগেই শেষ হয়ে গেছে। dfs এর কমপ্লেক্সিটি O(V+E)।

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](http://www.shafaetsplanet.com/planetcoding/?p=848),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।









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](http://uva.onlinejudge.org/external/115/11504.html)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:

[?](http://www.shafaetsplanet.com/planetcoding/?p=973)

|  |  |
| --- | --- |
| কোড: | |
| 3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | DFS (V, E)         for each vertex u in V[G]          do color[u] ← WHITE                  π[u] ← NIL         ▷ π[] is parent array       time ← 0       for each vertex u in V[G]          do if color[u] ← WHITE                  then DFS-Visit(u)              ▷ build a new DFS-tree from u        DFS-Visit(u)         color[u] ← GRAY                         ▷ discover u       time ← time + 1                         ▷ stopwatch       d[u] ← time       for each vertex v adjacent to u     ▷ explore (u, v)          do if color[v] ← WHITE                  then π[v] ← u             ▷ saving parent to print path                          DFS-Visit(v)       color[u] ← BLACK       f[u] ← time                                 ▷ we are done with u  #<http://www.personal.kent.edu/>~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/depthSearch.htm |

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>

যদি অ্যানিমেশন দেখতে সমস্যা হয় তাহলে সবগুলো ছবি ডাউনলোড করতে পারো [এখান থেকে](http://shafaetsplanet.com/uploads/dfstutorialpic/dfstutorialpic.zip) ।

এরপরে [এই আর্টিকেলটা](http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/depthSearch.htm) পড়ে ফেলো বিস্তারিত জানার জন্য,আমার লেখা পড়ে তুমি বেসিকটা শিখতে পারবে,বিস্তারিত জানতে এবং কঠিন প্রবলেম সলভ করতে আরো অনেক কিছু জানতে হবে।Top of Form

### ৭টি মন্তব্য

**ফারসান**

ভাইয়া শেষ animation এর ১-২ এর edge direction টা মনে হয় উলটা হবে…

**শাফায়েত**

আমারতো মনে হচ্ছে ঠিকই আছে..

**Sumit**

৩ আগস্ট ২০১৩

I am finding a problem on pseudo code.I think I need to increase time by 1 befor recording finishing time.I am also in a confushion.

[গ্রাফ থিওরিতে হাতেখড়ি-১০ (ডায়াক্সট্রা ডায়াক্সট্রা!)](http://www.shafaetsplanet.com/planetcoding/?p=1500)

আমরা শুরুতেই শিখেছি কিভাবে শর্টেস্ট পাথে এক জায়গা থেকে আরেক জায়গায় যেতে হয়। সেজন্য আমরা শিখেছি [ব্রেথড ফার্স্ট সার্চ](http://www.shafaetsplanet.com/planetcoding/www.shafaetsplanet.com/planetcoding/?p=604) নামের একটি সার্চিং অ্যালগোরিদম। অ্যালগোরিদমটি চমৎকার কিন্তু সমস্যা হলো সে ধরে নেয় প্রতিটি রাস্তা দিয়ে যেতে সমান সময় লাগে, মানে সব এজ এর কস্ট সমান। প্র্যাকটিকাল লাইফে বেশিভাগ ক্ষেত্রেই এটা অচল হয়ে পড়ে, তখন আমাদের দরকার পরে ডায়াক্সট্রা। প্রথমে নাম শুনে আমার ধারণা হয়েছিলো ডায়াক্সট্রা খুবই ভয়ংকর কোনো জিনিস কিন্তু বিশ্বাস করো বিএফএস লেখার মতোই সহজ ডায়াক্সট্রা লেখা, আমি তোমাদের দেখানোর চেষ্টা করবো কিভাবে বিএফএসকে কিছুটা পরিবর্তন করে একটা প্রায়োরিটি কিউ যোগ করে সেটাকে ডায়াক্সট্রা বানিয়ে ফেলা যায়।

ডায়াক্সট্রা শুরু করার আগে আমরা পাথ রিল্যাক্সেশন(relax) নামের একটা ছোট্ট জিনিসের সাথে পরিচিত হই। ধরো সোর্স থেকে প্রতিটা নোডের ডিসটেন্স রাখা হয়েছে d[] অ্যারেতে। যেমন d[3] মানে হলো সোর্স থেকে বিভিন্ন এজ পার হয়ে ৩ এ আসতে মোট d[3] ডিসটেন্স লেগেছে। যদি ডিসটেন্স জানা না থাকে তাহলে ইনফিনিটি অর্থাৎ অনেক বড় একটা মান রেখে দিবো। আর cost[u][v] তে রাখা আছে u-v এজ এর কস্ট।

ধরো তুমি বিভিন্ন জায়গা ঘুরে ফার্মগেট থেকে টিএসসি তে গেলে ১০ মিনিটে, আবার ফার্মগেট থেকে কার্জন হলে গেলে ২৫ মিনিটে। তাহলে তোমার কাছে ইনফরমেশন আছে:

d[টিএসসি]=১০, d[কার্জনহল]=২৫

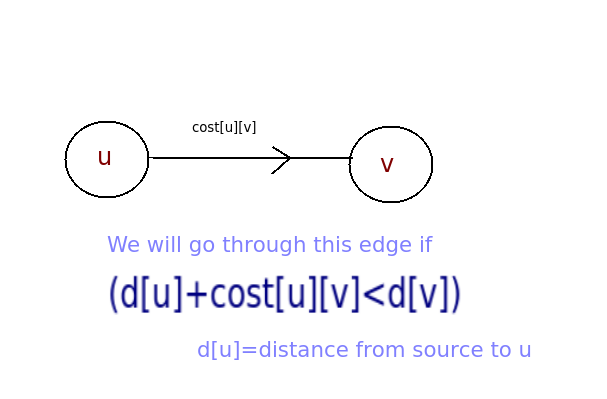
এখন তুমি দেখলে টিএসসি থেকে ৭ মিনিটে কার্জনে চলে যাওয়া যায়,

cost[টিএসসি][কার্জন]=৭

তাহলে তুমি ২৫ মিনিটের জায়গায় মাত্র ১০+৭=১৭ মিনিটে কার্জনহলে যেতে পারবে। যেহেতু তুমি দেখেছো:

d[টিএসসি]+cost[টিএসসি][কার্জন] < d[কার্জনহল]

তাই তুমি এই নতুন রাস্তা দিয়ে কার্জন হলে গিয়ে d[কার্জনহল]=d[টিএসসি]+cost[টিএসসি][কার্জন] বানিয়ে দিতেই পারো!!

[](http://www.shafaetsplanet.com/planetcoding/?attachment_id=1502)

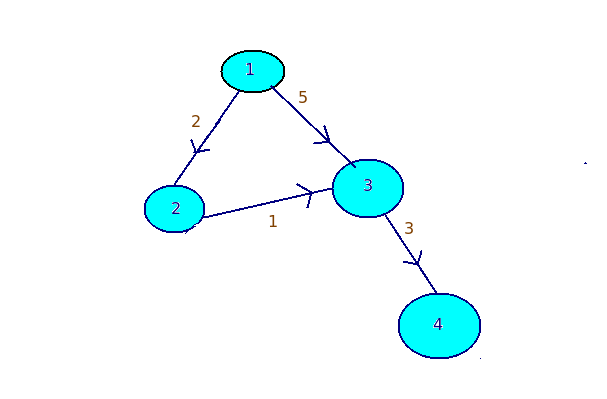
উপরের ছবিটা সেটাই বলছে। আমরা u থেকে v তে যাবো যদি **d[u]+cost[u][v] < d[v]** হয়। আর d[v] কে আপডেট করে d[v]=d[u]+cost[u][v] বানিয়ে দিবো। ভবিষ্যতে যদি কার্জনহলে অন্য রাস্তা দিয়ে আরো কম সময়ে যেতে পারি তখন সেই রাস্তা এভাবে কম্পেয়ার করে আপডেট করি দিবো। ব্যাপারটা অনেকটা এরকম:

[?](http://www.shafaetsplanet.com/planetcoding/?p=1500)

|  |  |
| --- | --- |
| কোড: | |
| 3  4  5  6 | if(d[u]+cost[u][v] < d[v])  {      d[v]=d[u]+cost[u][v];  } |

উপরের অংশটা যদি বুঝে থাকো তাহলে ডায়াক্সট্রা বোঝার ৬০% কাজ হয়ে গেছে। না বুঝে থাকলে আবার পড়ো।

বিএফএস নিশ্চয়ই তুমি ভালো করে বুঝো।  [বিএফএস](http://www.shafaetsplanet.com/planetcoding/www.shafaetsplanet.com/planetcoding/?p=604) এ আমাদের একটা নোডে কখনো দুইবার যাওয়া দরকার হয়নি, আমরা প্রতিবার দেখেছি একটা নোড ভিজিটেড কিনা, যদি ভিজিটেড না হয় তাহলে সেই নোডকে কিউতে পুশ করে দিয়েছি এবং ডিসটেন্স ১ বাড়িয়ে দিয়েছি। ডায়াক্সট্রাতে আমরা একই ভাবে কিউ তে নোড রাখবো তবে ভিজিটেড দিয়ে আপডেট না করে নতুন এজকে “রিল্যাক্স” বা আপডেট করবো উপরের পদ্ধতিতে। নিচের গ্রাফটা দেখো:

[](http://www.shafaetsplanet.com/planetcoding/?attachment_id=1501)

ধরে নেই সোর্স হলো ১ নম্বর নোড। তাহলে

d[1]=0, d[2]=d[3]=d[4]=infinity(a large value)

ইনফিনিটি কারণ ২,৩,৪ এর দূরত্ব আমরা এখনো জানিনা, আর সোর্সের দূরত্ব অবশ্য শূন্য। এখন তুমি আগের বিএফএস এর মতোই সোর্স থেকে যতগুলো নোডে যাওয়া যায় সেগুলা আপডেট করার চেষ্টা করো, আপডেট করতে পারলে কিউতে পুশ করো। যেমন ১-২ এজটা ধরে আমরা আগাবো কারণ d[1]+2 < d[2] এই শর্তটা পূরণ হচ্ছে। তখন d[2] হয়ে যাবে ২, একই ভাবে ১ থেকে ৩ এ গেলে d[3] হয়ে যাবে ৫।

কিন্তু ৫ তো ৩ নম্বরনোডে যাওয়ার শর্টেস্ট ডিসটেন্স না! আমরা বিএফএস এ দেখেছি একটা নোড একবারের বেশি আপডেট হয়না, সেই প্রোপার্টি এখানে কাজ করছেনা। ২ নম্বর নোড থেকে ২-৩ এজ ধরে এগিয়ে আবার আপডেট করলে তখন d[3] তে d[2]+1=3 পাবো। তাহলে আমরা দেখলাম এক্ষেত্রে একটা নোড অনেকবার আপডেট হতে পারে। (প্রশ্ন: সর্বোচ্চ কত বার?)

আমরা তাহলে আগের বিএফএস এর কোডের আপডেট অংশ একটু পরিবর্তন করি যাতে একটা নোড বার বার আপডেট হতে পারে:

[?](http://www.shafaetsplanet.com/planetcoding/?p=1500)

|  |  |
| --- | --- |
| কোড: | |
| 3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | vector<int>G[100];  int cost[100][100];    void shortest\_path(int n,int src)  {      int d[100];      for(int i=1;i<=n;i++)      d[i]=1000000000;      queue<int>Q;      Q.push(src);      d[src]=0;      while(!Q.empty())      {          int u=Q.front();          for(int i=0;i<(int)G[u].size();i++)          {              int v=G[u][i];                if(d[u]+cost[u][v]<d[v]) // update/relaxation              {                  d[v]=d[u]+cost[u][v];                  Q.push(v);              }          }          Q.pop();      }  } |

আমরা ঠিক আগের বিএফএস এর কোডেই জাস্ট কস্ট বসায় বারবার আপডেট করছি! এই কোড তোমাকে সোর্স থেকে প্রতিটা নোডের শর্টেস্ট পাথ বের করে দিবে কিন্তু কমপ্লেক্সিটির দিক থেকে এটা খুবই বাজে! এজন্য আমাদের লাগবে একটা প্রায়োরিটি কিউ।

বিএফএস এ আমরা যখন ১ নোড থেকে ৫টা নোডে যাচ্ছি তখন সেই ৫টা নোড থেকে আবার নতুন করে কাজ করার সময় "আগে আসলে আগে পাবেন" ভিত্তিতে কাজ করছি। যেমন উপরের গ্রাফে ১ থেকে আগে ৩ এবং তারপর ২ এ গেলে আগে ৩ নিয়ে কাজ করছি। ভালো করে দেখো এখানে কি সমস্যাটা হচ্ছে। ৩ নিয়ে আগে কাজ করলে আমরা ৫ এর ডিসটেন্সকে আপডেট করে দিচ্ছি ডিসটেন্স ৫+২=৭ হিসাবে। পরবর্তীতে যখন ২ দিয়ে ৩ কে আবার আপডেট করা হচ্ছে তখন ৩ এর ডিসটেন্স হয়ে গিয়েছে ৩, এবার ৫ এর ডিসটেন্সকে আবার আপডেট করছি ৩+২=৫ হিসাবে। ৫ কে মোট দুইবার আপডেট করা লাগলো।

বিজ্ঞানী ডায়াক্সট্রা চিন্তা করলেন যদি এই "আগে আসলে আগে পাবেন" ভিত্তিতে কাজ না করে সবথেকে কাছের নোডগুলোকে আগে প্রসেস করি তাহলে অনেক কমবার আপডেট করা লাগে। আমরা যদি ২ কে নিয়ে আগে কাজ করতাম তাহলে ৩ আগেই আপডেট হয়ে যেত এবং ৫ কে একবার আপডেট করেই শর্টেস্ট ডিসটেন্স পেয়ে যেতাম! একটু হাতে কলমে সিমুলেট করে দেখো। আইডিয়াটা হলো যেকোনো সময় কিউ তে যতগুলো নোড থাকবে তাদের মধ্যে যেটা সোর্স থেকে সবথেকে কাছে সেটা নিয়ে আগে কাজ করবো। এজন্যই আমরা কিউ এর জায়গায় বসিয়ে দিবো একটি প্রায়োরিটি কিউ যে কিউতে নোড পুশ করার সাথে সাথে কাছের নোডটাকে সামনে এনে দিবে। পার্থক্য হলো আগে খালি নোড নাম্বার পুশ করেছি, এখন বর্তমান ডিসটেন্স অর্থাত d[u] এর মানটাও পুশ করতে হবে।

নিচে একটা সম্পূর্ণ ডায়াক্সট্রার কোড দিলাম যেটা ১ থেকে n তম নোডে যাবার শর্টেস্ট পাথ বের করে এবং পাথটাও প্রিন্ট করে:

[?](http://www.shafaetsplanet.com/planetcoding/?p=1500)

|  |  |
| --- | --- |
| কোড: | |
| 3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | #define mx 100002  vector<int>g[mx],cost[mx];  struct node  {      int u,w;      node(int a,int b){u=a; w=b;}      bool operator < ( const node& p ) const {      return w > p.w;   }  };  int d[mx],par[mx];  int dijkstra(int n)  {      memset(d,63,sizeof(d));  //অনেক বড় একটা মান বসাতে!      memset(par,-1,sizeof(par));      priority\_queue<node>q;      q.push(node(1,0));      d[1]=0;      while(!q.empty())      {          node top=q.top(); q.pop();          int u=top.u;            if(u==n) return d[n];          for(int i=0;i<(int)g[u].size();i++)          {              int v=g[u][i];              if(d[u]+cost[u][i]<d[v])              {                  d[v]=d[u]+cost[u][i];                  par[v]=u;                  q.push(node(v,d[v]));              }          }      }      return -1;  }  int main(){      int n,e;      cin>>n>>e;      for(int i=0;i<e;i++)      {          int u,v;          int w;          cin>>u>>v>>w;          g[u].push\_back(v);          g[v].push\_back(u);          cost[u].push\_back(w);          cost[v].push\_back(w);        }      int ret=dijkstra(n);      if(ret==-1) puts("No path!");      else      {          int u=n;          vector<int>out;          while(u!=-1)          {              out.push\_back(u);                u=par[u];            }          reverse(out.begin(),out.end());          for(int i=0;i<(int)out.size();i++)              cout<<out[i]<<" ";          puts("");        }  } |

এখানে কয়েক জায়গায় একটু ভিন্নতা আছে। একটা হলো cost[][] অ্যারেতে কস্ট না রেখে গ্রাফের অ্যাডসেন্সি লিস্টের পাশাপাশি কস্ট এর লিস্ট রেখেছি, অ্যারে রাখলে নোড বেশি হলে সমস্যা করবে। শুরুতেই node নামের একটা স্ট্রাকচার বানিয়ে নিয়েছি যেখানে নোডের নম্বরের সাথে কস্ট টাও রেখে দেয়া হয়েছে। কস্ট অনুসারে অপারেটর ওভারলোড করেছি যাতে প্রায়োরিটি কিউতে যার কস্ট কম সেটা আগে আসে। বাকি কাজ আগের মতোই।

একটা ব্যাপার লক্ষ্য করো, টার্গেট নোড পপ হবার সাথে সাথে আমরা কস্ট রিটার্ণ করে দিয়েছি। এটার কারণ কিউতে বাকি যত নোড আছে সবার কস্ট আরো বেশি বা সমান তাই ভবিষ্যতে টার্গেট নোডের আর কখনো আপডেট হবার সম্ভাবনা নেই।

নেগেটিভ এজ থাকলে কি ডায়াক্সট্রা অ্যালগোরিদম কাজ করবে? যদি নেগেটিভ সাইকেল থাকে তাহলে ইনফিনিট লুপে পড়ে যাবে, বারবার আপডেট করে কস্ট কমাতে থাকবে। যদি নেগেটিভ এজ থাকে কিন্তু সাইকেল না থাকে তাহলেও কাজ করবেনা। তবে তুমি যদি টার্গেট পপ হবার সাথে সাথে রিটার্ণ করে না দাও তাহলে কাজ করবে কিন্তু সেটা তখন আর মূল ডায়াক্সট্রা অ্যালগোরিদম থাকবেনা।

ডায়াক্সট্রা অ্যালগোরিদমের কমপ্লেক্সিটি কত? বিএফএস এর কমপ্লেক্সিটি ছিলো log(V+E) যেখানে V হলো নোড সংখ্যা আর E হলো এজ সংখ্যা। এখানেও আগের মতোই কাজ হবে তবে প্রায়োরিটি কিউ তে প্রতিবার সর্ট করতে logV কমপ্লেক্সিটি লাগবে। মোট: log(VlogV+E) ।

নেগেটিভ সাইকেল নিয়ে কাজ করতে হলে আমাদের জানতে হবে বেলম্যান ফোর্ড অ্যালগোরিদম। সেখানেও এজ রিল্যাক্স করে আপডেট করা হয়, একটা নোডকে সর্বোচ্চ n-1 বার আপডেট করা লাগতে পারে সেই প্রোপার্টি কাজে লাগানো হয়। বেলম্যান নিয়ে আলোচনা হবে অন্য আরেকদিন, তুমি চাইলে এখনই গুগল করে শিখে নিতে পারো। আর ডায়াক্সট্রা ভালো করে শিখতে নিচের প্রবলেমগুলো ঝটপট করে ফেলো:

[Dijkstra?](http://codeforces.com/contest/20/problem/C)  
[Not the Best](http://www.lightoj.com/volume_showproblem.php?problem=1099)  
[New Traffic System](http://www.lightoj.com/volume_showproblem.php?problem=1281)

হ্যাপি কোডিং!

[(গ্রাফ থিওরি নিয়ে সবগুলো লেখা)](http://www.shafaetsplanet.com/planetcoding/?tag=%E0%A6%97%E0%A7%8D%E0%A6%B0%E0%A6%BE%E0%A6%AB-%E0%A6%A5%E0%A6%BF%E0%A6%93%E0%A6%B0%E0%A6%BF)