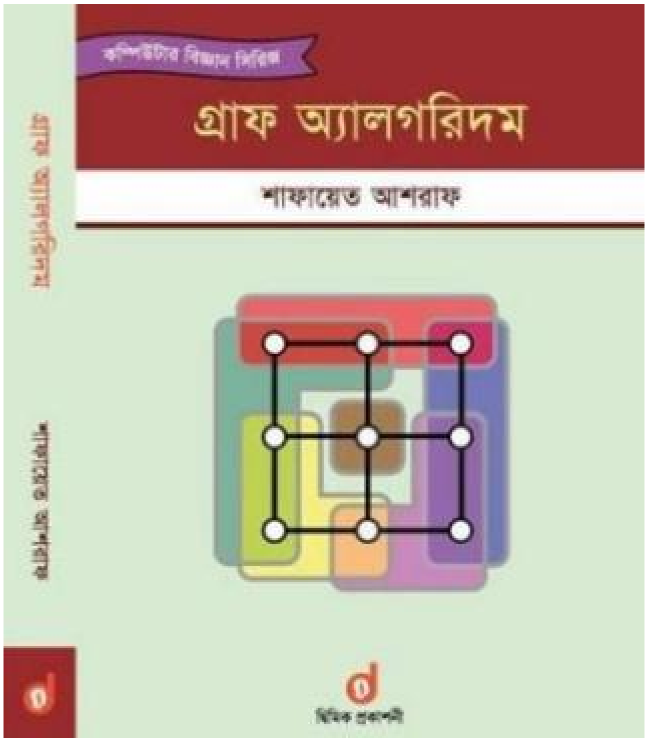
শাফায়েতের ব্লগ

প্রোগ্রামিং, অ্যালগরিদম, ব্যাকএন্ড ইঞ্জিনিয়ারিং

Home
অ্যালগরিদম নিয়ে যত লেখা!
আমার সম্পর্কে...

গ্রাফ থিওরিতে হাতেখড়ি – ২ (ভ্যারিয়েবলে গ্রাফ স্টোর-১)

ডিসেম্বর ২৬, ২০১০ by *Shafaet Ashraf*



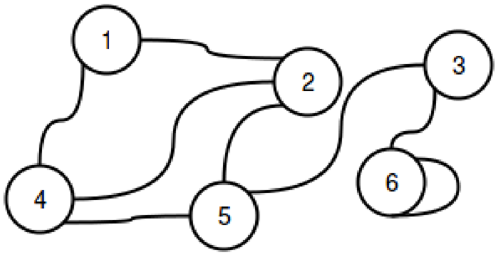
আগের পোস্টে আমরা দেখেছি গ্রাফ থিওরি কি কাজে লাগে,আর এলিমেন্টারি কিছু টার্ম শিখেছি। এখন আমরা আরেকটু ভিতরে প্রবেশ করবো। প্রথমেই আমাদের জানা দরকার একটা গ্রাফ কিভাবে ইনপুট নিয়ে স্টোর করে যায়। অনেকগুলো পদ্ধতির মধ্যে দুটি খুব কমন:

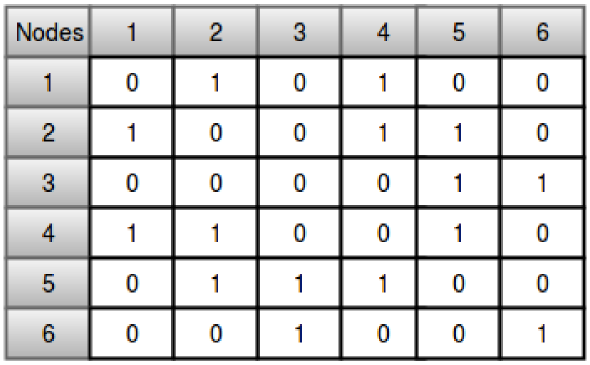
top

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 শুধুমাত্র ২-ডি অ্যারে বুঝানো হয়েছে, তাই ঘাবড়ে যাবার কিছু নেই!



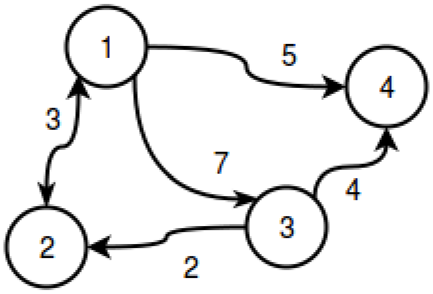


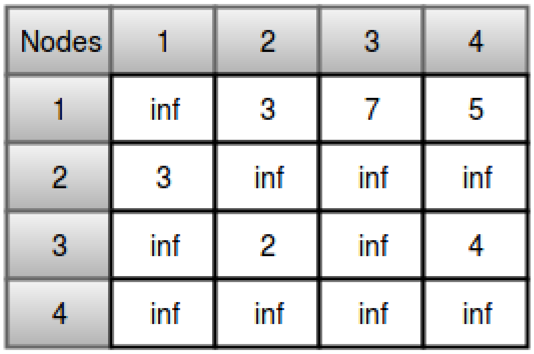
Nodes	1	2	3	4	5	6
1	0	1	0	1	0	0
2	1	0	0	1	1	0
3	0	0	0	0	1	1
4	1	1	0	0	1	0
5	0	1	1	1	0	0
6	0	0	1	0	0	1

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 0 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 ডিরেক্টেড ওয়েটেড গ্রাফের অ্যাডজেসেন্সি ম্যাট্রিক্সের উদাহরণ দেখানো হয়েছে।





Nodes	1	2	3	4
1	inf	3	7	5
2	3	inf	inf	inf
3	inf	2	inf	4
4	inf	inf	inf	inf

যেসব নোড এর ভিতর কোনো এজ নাই তাদেরকে এখানে ইনফিনিটি বা অনেক বড় একটা সংখ্যা দিয়ে 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 অ্যারে 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*

top

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।

অ্যাডজেসেন্সি ম্যাট্রিক্স ব্যবহার করার সমস্যা:

মেমরি একটা বিশাল প্রবলেম, এজ যতগুলোই থাকুকনা কেন তোমার লাগছে $N*N$ সাইজের ম্যাট্রিক্স যেখানে N হলো নোড সংখ্যা। 10000 টা নোড হলো $N*N$ম্যাট্রিক্সের সাইজ দাড়াবে $4*1000*1000$ বাইট বা প্রায় 381 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।

top