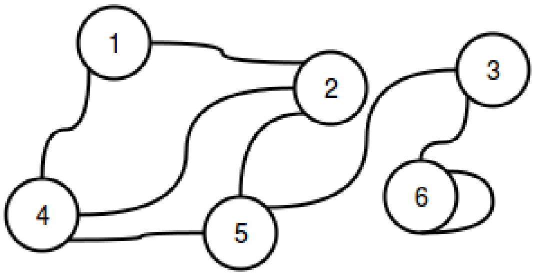
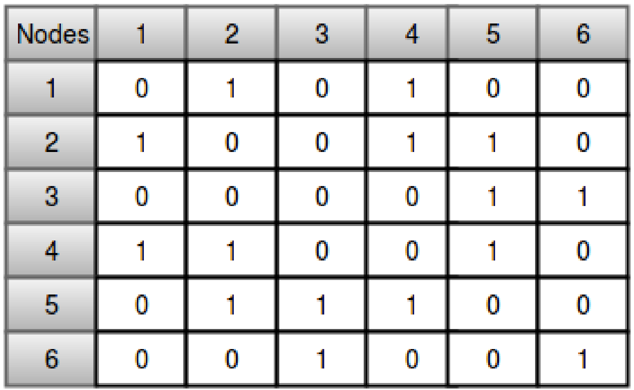
এই পর্বে গ্রাফ স্টোর করার ২য় পদ্ধতি অ্যাডজেসেন্সি লিস্ট নিয়ে লিখব। এ পদ্ধতিতে গ্রাফ স্টোর করে কম মেমরি ব্যবহার করে আরো efficient কোড লেখা যায়। এ ক্ষেত্রে আমরা ডায়নামিক্যালি মেমরি অ্যালোকেট করব,ভয়ের কিছু নেই সি++ এর standard template library(STL) ব্যবহার করে খুব সহজে কাজটা করা যায়। আগের লেখার শেষের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

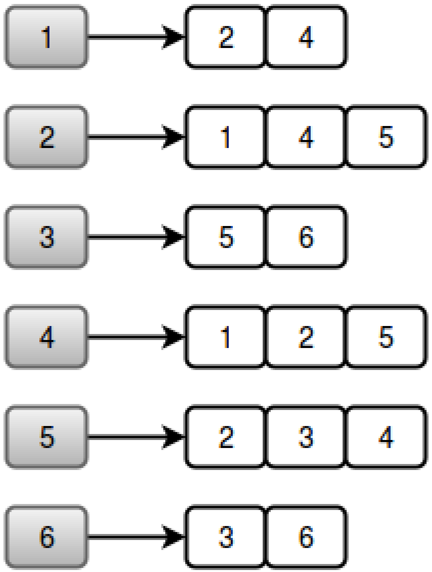
অ্যাডজেসেন্সি লিস্ট শুনতে যতটা ভয়ংকর শুনায়,ব্যাপারটি আসলে তেমনই সহজ। আমরা আবার আগের পোস্টের ছবিটিতে ফিরে যাই:





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

এবার বাজার করার লিস্টের মত একটি লিস্ট বানাই:



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 numNodes, numEdges;

  scanf("%d%d", &numNodes, &numEdges);
  for (int i = 1; i <= numEdges; 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:

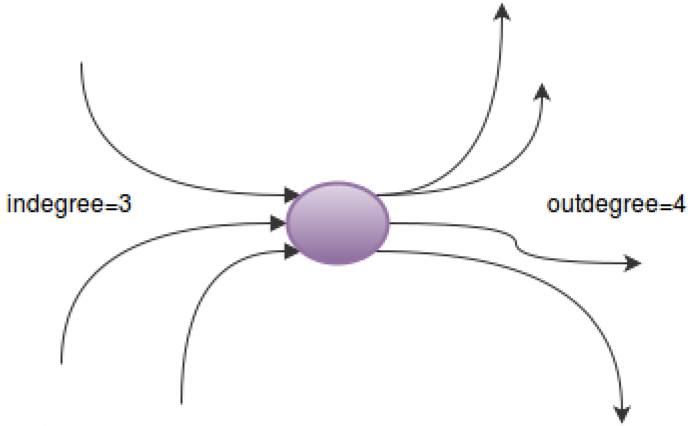
ধরো তুমি ১ নম্বর নোডের সাথে যুক্ত সবগুলো নোডের নম্বর চাও, তুমি তাহলে edges[1] এর সাইজ পর্যন্ত লুপ চালাবে এভাবে:

```
size=edges[1].size();
for(int i=0; i < size ; i++)
printf("%d ",edges[1][i]);
```

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, 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