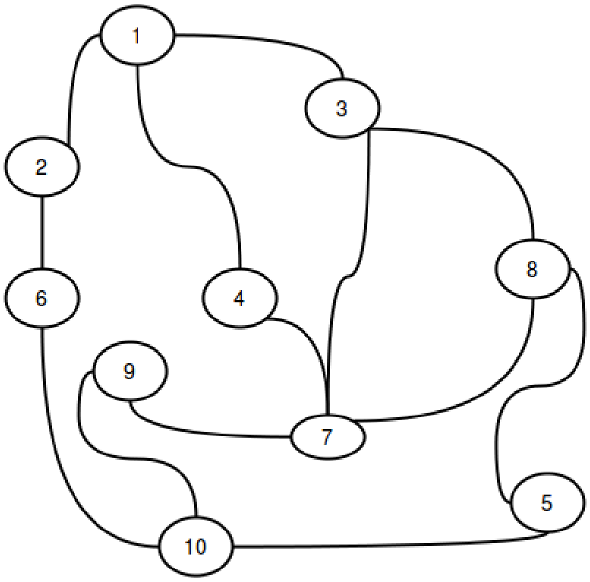
আগের পর্বগুলোতে আমরা দেখেছি কিভাবে ম্যাট্রিক্স বা লিস্ট ব্যবহার করে গ্রাফ স্টোর করতে হয়। এবার আমরা প্রথম অ্যালগোরিদম দেখবো এর দিকে যাবো। শুরুতেই আমরা যে অ্যালগোরিদমটা শিখব তার নাম ব্রেডথ ফার্স্ট সার্চ(breadth first search,bfs)।
বিএফএস এর কাজ হলো গ্রাফে একটা নোড থেকে আরেকটা নোডে যাবার শর্টেস্ট পাথ বের করা। বিএফএস কাজ করবে শুধুমাত্র আন-ওয়েটেড গ্রাফের ক্ষেত্রে, তারমানে সবগুলো এজের কস্ট হবে ১।

বিএফএস অ্যালগোরিদমটা কাজ করে নিচের ধারণারগুলোর উপর ভিত্তি করে:

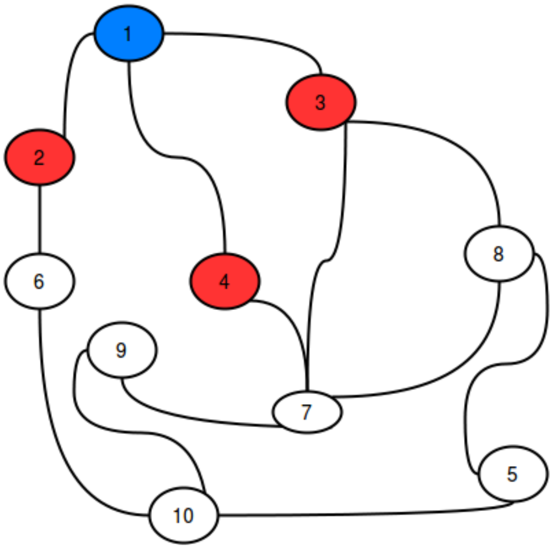
১. কোনো নোডে ১ বারের বেশি যাওয়া যাবেনা
২. সোর্স নোড অর্থাৎ যে নোড থেকে শুরু করছি সেটা ০ নম্বর লেভেলে অবস্থিত।
৩. সোর্স বা ‘লেভেল ০’ নোড থেকে সরাসরি যেসব নোডে যাওয়া যায় তারা সবাই ‘লেভেল ১’ নোড।
৪. ‘লেভেল ১’ নোডগুলো থেকে সরাসরি যেসব নোডে যাওয়া যায় তারা সবাই ‘লেভেল ২’ নোড। এভাবে লেভেল এক এক করে বাড়তে থাকবে।
৫. যে নোড যত নম্বর লেভেলে,সোর্স থেকে তার শর্টেস্ট পথের দৈর্ঘ্য তত।

উপরে লেখাগুলো পুরোপুরি না বুঝলে আমরা একটা উদাহরণ দেখে বাকিটা পরিস্কার করব।

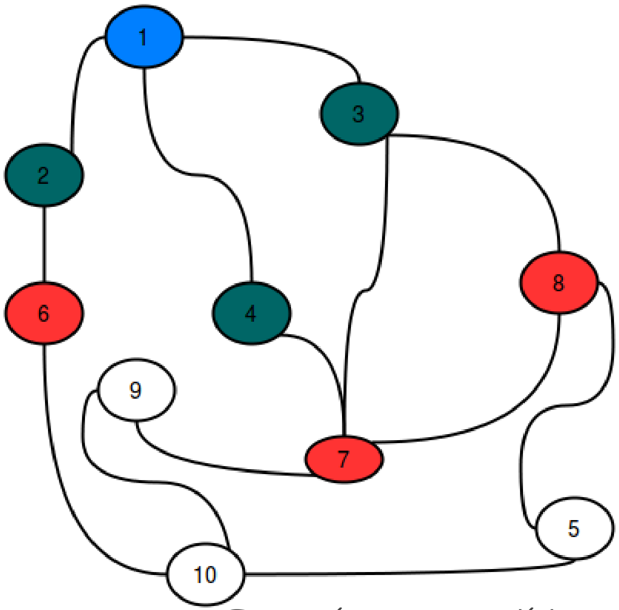


ধর তুমি ১ নম্বর শহর থেকে ১০ নম্বর শহরে যেতে চাও। প্রথমে আমরা সোর্স ধরলাম ১ নম্বর নোডকে। ১ তাহলে একটা ‘লেভেল ০’ নোড। ১ কে ভিজিটেড চিহ্নিত করি।

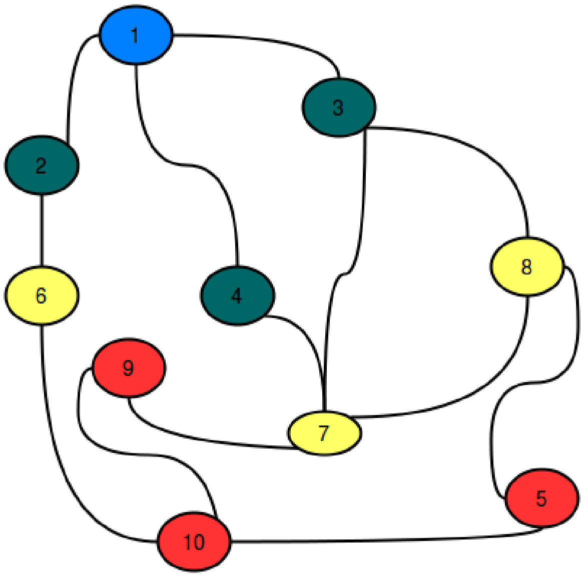
১ থেকে সরাসরি যাওয়া যায় ২,৩,৪ নম্বর নোডে। তাহলে ২,৩,৪ হলো ‘লেভেল ১’ নোড। এবার সেগুলোকে আমরা ভিজিটেড চিহ্নিত করি এবং সেগুলো নিয়ে কাজ করি। নিচের ছবি দেখ:



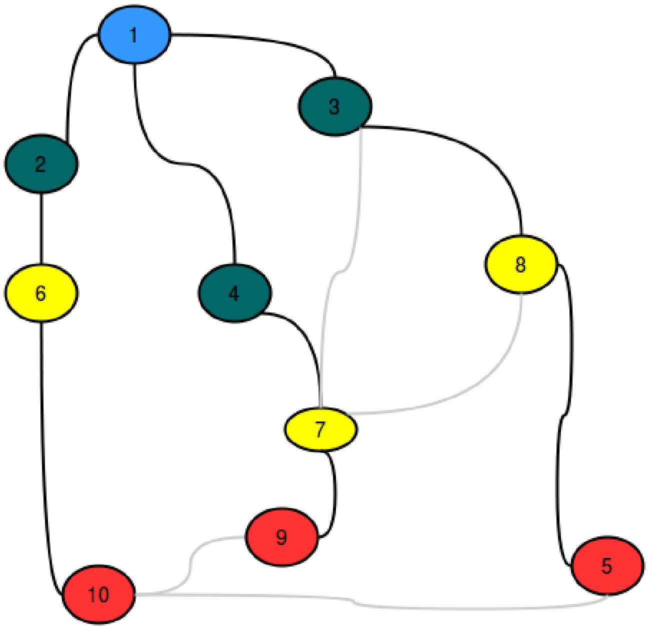
লাল নোডগুলো নিয়ে আমরা এখন কাজ করবো। রঙিন সবগুলো নোড ভিজিটেড, **এক নোডে ২বার কখনো যাবোনা**। ২,৩,৪ থেকে শর্টেস্ট পথে যাওয়া যায় ৬,৭,৮ এ। সেগুলো ভিজিটেড চিহ্নিত করি:



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 এককেটা 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 সার্চ করে আমরা শর্টেস্ট পথ বের করলাম। যেসব এজ গুলো আমরা ব্যবহার করিনি সেগুলোকে বাদ দিয়ে ছবিটিকে নিচের মত করে আকতে পারি:



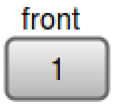
যেসব এজ ব্যবহার করিনি সেগুলো হালকা করে দিয়েছি,এই এজ গুলো বাদ দিলে গ্রাফটি একটি ট্রি হয়ে যায়। এই ট্রি টাকে বলা হয় বিএফএস ট্রি।

তারমানে আমাদের কাজ গুলো সোর্স থেকে লেভেল ১ নোডগুলোতে যাওয়া, তারপর লেভেল ১ এর নোডগুলো থেকে লেভেল ২ নোডগুলো খুজে বের করা, এভাবে যতক্ষন না গন্তব্যে পৌছে যাচ্ছি অথবা সব নোড ভিজিট করা শেষ হয়ে গিয়েছে ততক্ষণ কাজ চলতে থাকবে।

কিউ ডাটা স্ট্রাকচারটার সাথে আশা করি সবাই পরিচিত। কিউ হলো হুবুহু বাসের লাইনের মতো ডাটা স্ট্রাকচার। যখন একটা সংখ্যা কিউতে যোগ করা হয় তখন সেটা আগের সবগুলো সংখ্যার পিছে

গিয়ে দাড়ায়, যখন কোন একটা সংখ্যা বের করে ফেলা হয় তখন সবার প্রথমের সংখ্যাটা নেয়া হয়। একে বলা ফার্স্ট ইন ফার্স্ট আউট। আমরা বিএফএস এ কিউ কাজে লাগাতে পারি। লেভেল ১ থেকে যখন কয়েকটা নতুন লেভেল ২ নোড পাবো সেগুলোকে কিউতে বা লাইনে অপেক্ষা করিয়ে রাখবো, আর সবসময় প্রথম নোডটা নিয়ে কাজ করবো। তাহলে বড় লেভেলের নোডগুলো সবসময় পিছের দিকে থাকবে, আমরা ছোট লেভেলগুলো নিয়ে কাজ করতে করতে আগাবো। উপরের গ্রাফের জন্য এটা আমরা সিমুলেট করে দেখি:

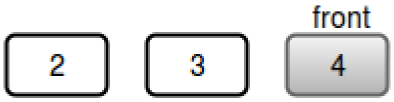
প্রথমে কিউতে সোর্স পুশ করবো:



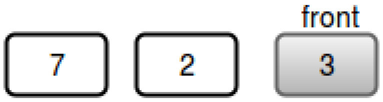
১ এর লেভেল হবে ০ বা লেভেল[১] = ০। এবার বিএফএস শুরু করবো।

প্রথমে কিউ এর সবার সামনের নোডটাকে নিয়ে কাজ করবো। সবার সামনে আছে ১, সেখান থেকে যাওয়া যায় ৪, ৩, ২ এ। ৪, ৩, ২ এ এসেছি ১ থেকে, তাহলে লেভেল[৪] = লেভেল[১] + ১ = ১, লেভেল[৩] = লেভেল[১] + ১ = ১, লেভেল[২] = লেভেল[১] + ১ = ১।

১ কে ফেল দিয়ে এদেরকে কিউতে পুশ করে রাখি:



এবার ৪ নিয়ে কাজ করি। ৪ থেকে যাওয়া যায় ৭ এ। তাহলে আমরা বলতে পারি লেভেল[৭]=লেভেল[৪]+১=২। ৪ কে ফেলে দিয়ে ৭ কে কিউতে পুশ করি:



৩ থেকে ৭,৮ এ যাওয়া যায়। ৭ কে এরই মধ্যে নিয়েছি, শুধু ৮ পুশ করতে হবে। লেভেল[৮]=লেভেল[৩]+১=২।

এভাবে যতক্ষণনা কিউ খালি হচ্ছে ততক্ষণ কাজ চলতে থাকবে। লেভেল[] অ্যারের মধ্যে আমরা পেয়ে যাবো সোর্স থেকে সবগুলো নোডের দূরত্ব!

সুডোকোড:

```
procedure BFS(G,source):
    Q=queue(), level[]=infinity
    Q.enqueue(source)
    level[source]=0
    while Q is not empty
       u ← Q.pop()
       for all edges from u to v in G.adjacentEdges(v) do
           if level[v] = infinity:
                    level[v] = level[u] + 1;
                     Q.enqueue(v)
            end if
        end for
       end while
    Return distance;
```

ঠিক যেভাবে সিমুলেট করেছি সেভাবেই কোডটা লিখেছি, আশা করি বুঝতে সমস্যা হচ্ছেনা।

শুধু পাথের দৈর্ঘ্য যথেষ্ট না, পাথটাও দরকার হতে পারে। লক্ষ্য করো আমরা uu থেকে vv তে যাবার সময় parent[v]=uparent[v]=u করে দিচ্ছি। আমরা প্রতিটা নোডের জন্য জানি কোন নোড থেকে সেই নোডে এসেছি। তাহলে আমরা যে নোডের জন্য পাথ বের করতে চাই সেই নোড থেকে তার প্যারেন্ট নোডে যেতে থাকবো যতক্ষণনা সোর্সে পৌছে যাই। খুবই সহজ কাজ, পাথ বের করার কোড করা তোমার উপর ছেড়ে দিলাম।

কমপ্লেক্সিটি:

প্রতিটা নোডে একবার করে গিয়েছি, প্রতিটা এজ এ একবার গিয়েছি।

তাহলে কমপ্লেক্সিটি হবে O(V+E)O(V+E) যেখানে VV হলো নোড সংখ্যা এবং EE হলো এজ সংখ্যা।

কখনো কখনো ২-ডি গ্রিডে বিএফএস চালানো লাগতে পারে। যেমন একটা দাবার বোর্ডে একটি ঘোড়া আর একটা রাজা আছে। ঘোড়াটা মিনিমাম কয়টা মুভে রাজার ঘরে পৌছাতে পারবে? অথবা একটা ২-ডি অ্যারেতে কিছু সেল ব্লক করে দেয়া হয়েছে, এখন কোনো সেল থেকে আরেকটি সেলে মিনিমাম মুভে পৌছাতে হবে, প্রতি মুভে শুধুমাত্র সামনে-পিছে-বামে-ডানে যাওয়া যায়। আগে নোডকে আমরা প্রকাশ করছিলাম একটা মাত্র সংখ্যা দিয়ে, এখন নোডকে প্রকাশ করতে হবে দুটি সংখ্যা দিয়ে, রো(row) নাম্বার, এবং কলাম নাম্বার। তাহলে আমরা নোড রিপ্রেজেন্ট করার জন্য সি তে একটা স্ট্রাকচার বানিয়ে নিতে পারি এরকম:

struct node{int r,c;};

অথবা আমরা সি++ এর “পেয়ার” ব্যবহার করতে পারি।

pair<int,int>

এ ক্ষেত্রে ভিজিটেড, প্যারেন্ট, লেভেল অ্যারেগুলো হবে ২ ডিমেনশনের,
যেমন visited[10][10]visited[10][10] ইত্যাদি। কিউতে নোডের বদলে স্ট্রাকচার পুশ করবো। আর কোন একটা ঘর থেকে অন্য ঘরে যাবার সময় চেক করতে হবে বোর্ডের বাইরে চলে যাচ্ছে কিনা। একটা স্যাম্পল সি++ কোড দেখি:

```
#define pii pair<int,int>
int fx[]={1,-1,0,0}; //ডিরেকশন অ্যারে
int fy[]={0,0,1,-1};
int cell[100][100]; //cell[x][y] যদি -১ হয় তাহলে সেলটা ব্লক
int d[100][100],vis[100][100]; //d means destination from source.
int row,col;
void bfs(int sx,int sy) //Source node is in [sx][sy] cell.
{
        memset(vis,0,sizeof vis);
        vis[sx][sy]=1;
        queue<pii>q; //A queue containing STL pairs
        q.push(pii(sx,sy));
        while(!q.empty())
        {
                pii top=q.front(); q.pop();
                for(int k=0;k<4;k++)
                {
                        int tx=top.uu+fx[k];
                        int ty=top.vv+fy[k]; //Neighbor cell [tx][ty]
                        if(tx>=0 and tx<row and ty>=0 and ty<col and cell[tx][ty]!=-1 and vis[tx][ty]==0) //Check if the neighbor is valid and not visited before.
                        {
                                vis[tx][ty]=1;
                                d[tx][ty]=d[top.uu][top.vv]+1;
                                q.push(pii(tx,ty)); //Pushing a new pair in the queue
                        }
                }
        }
}
```

তুমি যদি ডিরেকশন অ্যারের ব্যাপারটা না বুঝো তাহলে এই লেখাটা পড়লে আরো কিছু ডিটেইলস জানতে পারবে।

বিএফএস শুধুমাত্র আন-ওয়েটেড গ্রাফে কাজ করে, ওয়েটেড গ্রাফে শর্টেস্ট পাথ বের করতে ডায়াক্সট্রা অ্যালগোরিদম ব্যবহার করতে পারো। গ্রাফে নেগেটিভ সাইকেল থাকলে বেলম্যান ফোর্ড ব্যবহার করতে হবে।

প্র্যাকটিসের জন্য প্রবলেম:

Bicoloring(Bipartite checking)
A Node Too Far(Shortest path)
Risk(Shortest path)
Bombs! NO they are Mines!!(bfs in 2d grid)
Knight Moves(bfs in 2d grid)
We Ship Cheap(Printing path)
Word Transformation(strings)