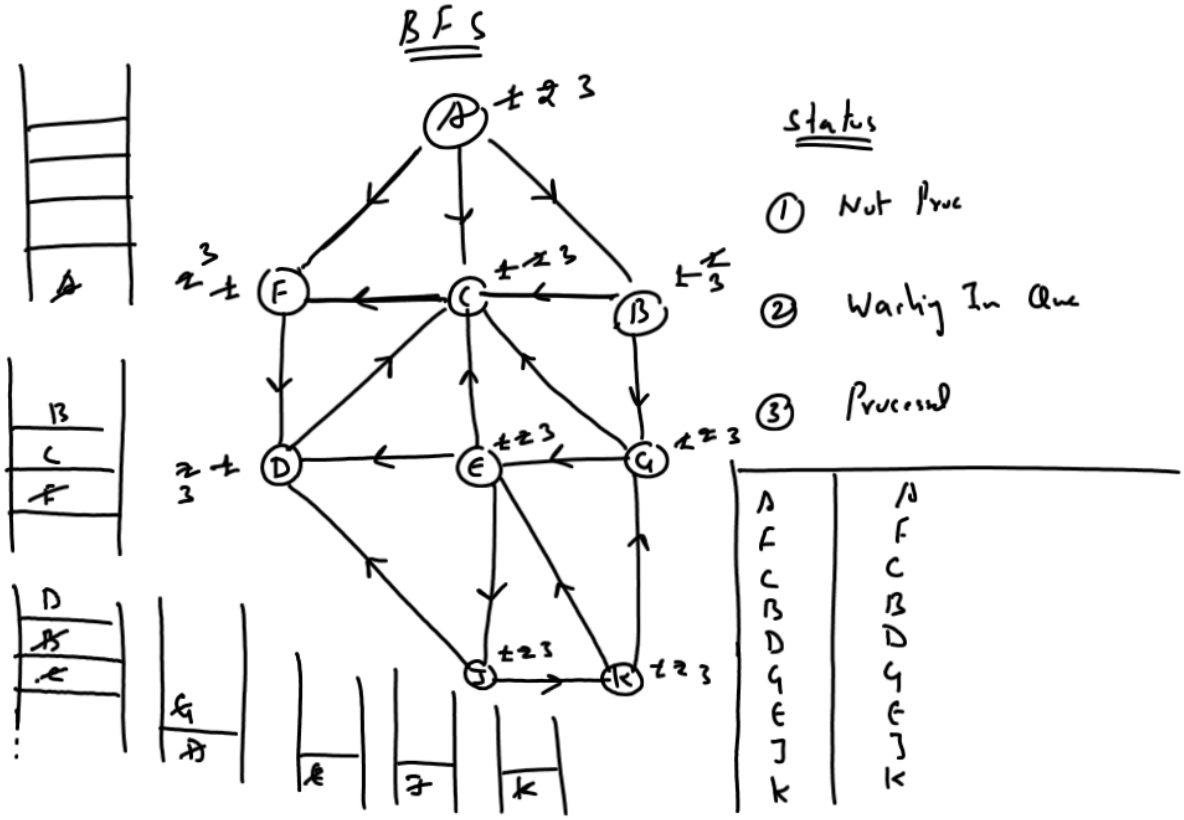
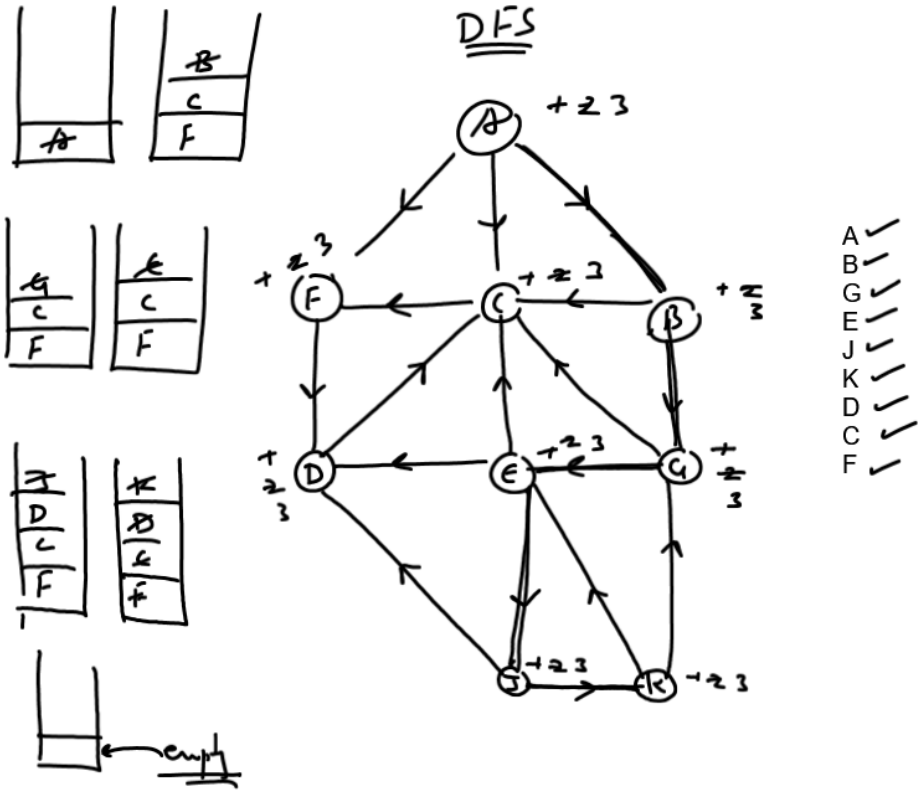
GRAPH TRAVERSAL TECHNIQUES

1. BFS (Breadth First Search) : uses Queue data structure
2. DFS (Depth First Search) uses Stack data structure





Algorithm For DFS
==========

1. Set the STATUS of each VERTEX to 1 , indcating that they have not yet been processed.

2. Find the SOURCE vertex , PUSH it in the STACK and set its STATUs to 2 , indicating that it is in the WAITING STATE.

3. POP the top node from the STACK and do the following:

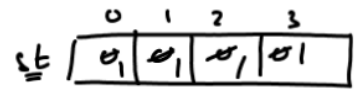
    a. Change its STATUS to 3 and print it
    b. Find all the adjacent nodes of the current node whose STATUS is 1.
    c. PUSH these nodes in the STACK and set their STATUS to 2

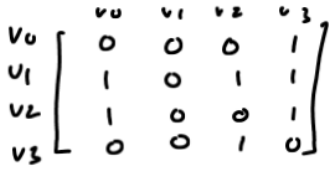
4. Repeat step 3 until STACK becomes empty.

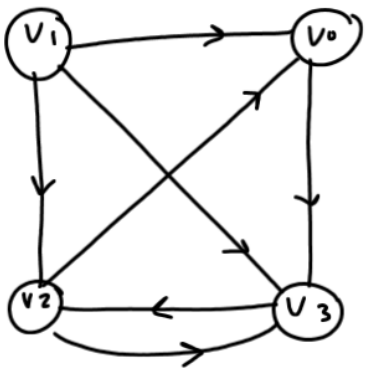
5. Finish and return

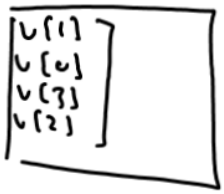
```
struct Stack
{
   int arr[10];
   int tos;
};
void push(struct Stack *,int);
int pop(struct Stack*);
void dfs(int[ ][4],int);
int main()
{
   int adj[4][4];
   int i,j,s;
   for(i=0;i<4;i++)
   {
        for(j=0;j<4;j++)
        {
           printf("Is there a path from v[%d] to v[%d],Y-1,N-0:",i,j);
           scanf("%d",&adj[i][j]);
        }
 }
 printf("Enter source vertex:");
 scanf("%d",&s);
 dfs(adj,s);
 return 0;
}
```

```
   void dfs(int adj[4][4],int v)
   {
     struct Stack S;
     int status[4]={0};
     int i;
     S.tos=-1;
     push(&S,v);
     while(S.tos!=-1)
     {
       v=pop(&S);
       if(status[v]==0)
         {
            printf("\nv[%d]",v);
            status[v]=1;
         }
       for(i=3;i>=0;i--)
       {
            if(adj[v][i]==1 && status[i]==0)
                  push(&S,i);
       }
   }
```









SHORTEST PATH