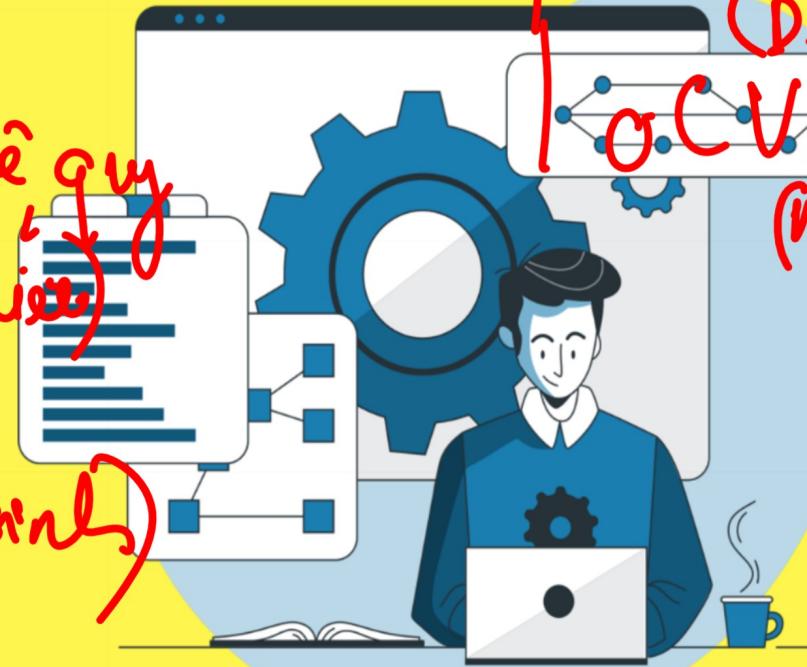




Chỉ thuật để dùng để tìm kiếm duyệt đồ thị:

DFS
BFS

→ Giải thuật đệ quy
Cây 2 chiều
queue ($O^2 \text{sinh}$)



{ OCV+E)
(DFK)
DCV+E)
(DSC)
OCV.V)
(MTC)



1. Thuật toán DFS - Depth First Search:



Thuật toán tìm kiếm theo chiều sâu là một giải thuật tìm kiếm, duyệt đồ thị sử dụng đệ quy. Thuật toán ưu tiên đi sâu nhất có thể, ở đây ta thống nhất khi mở rộng đồ thị thì sẽ ưu tiên mở rộng đỉnh có số thứ tự nhỏ hơn trước.

DFS

Đệ quy ~ luồng

Độ phức tạp

- $O(V + E)$ nếu sử dụng danh sách kề.
- $O(V * V)$ nếu sử dụng ma trận kề. (A_{nm})
- $O(V * E)$ nếu sử dụng danh sách cạnh.

E : Edge : Cây
V : Vertice : Chỗ

Mã giả:

```
DFS(u){  
    <Tham dinh u>  
    visited[u] = true; //Đánh dấu đã thăm u  
    //Duyệt danh sách kề của u  
    for(int v : adj[u]){  
        if(!visited[v]){  
            DFS(v);  
        }  
    }  
}
```



1. Thuật toán DFS - Depth First Search:

Code

```
int n, m; // đỉnh, cạnh
vector<int> adj[1005]; // danh sách kề
bool visited[1005];

void nhap(){
    cin >> n >> m;
    for(int i = 0; i < m; i++){
        int x, y; cin >> x >> y;
        adj[x].push_back(y);
        adj[y].push_back(x);
    }
    memset(visited, false, sizeof(visited));
}

void DFS(int u){
    cout << u << ' ';
    visited[u] = true;
    for(int v : adj[u]){
        if(!visited[v]){
            DFS(v);
        }
    }
}
```

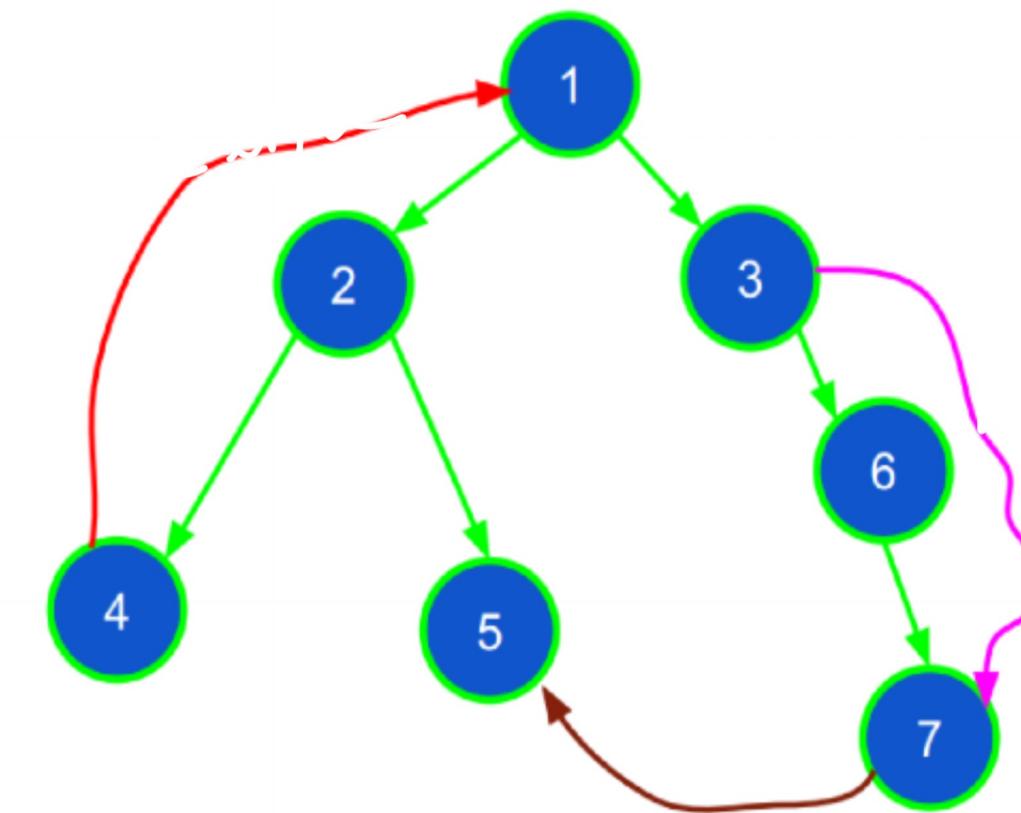
1. Thuật toán DFS - Depth First Search:



Khi thuật toán DFS mở rộng đồ thị sẽ tạo ra các loại cạnh khác nhau.

Các loại cạnh trên DFS:

- **Tree Edge:** là cạnh mà theo đó từ một đỉnh ta đến thăm một đỉnh mới.
- **Back Edge:** Cạnh ngược là cạnh đi từ con cháu (descendant) đến tổ tiên (ancestor)
- **Forward Edge:** Cạnh tới là cạnh đi từ tổ tiên tới hậu duệ.
- **Cross Edge:** Cạnh vòng là cạnh nối 2 đỉnh không có quan hệ họ hàng.





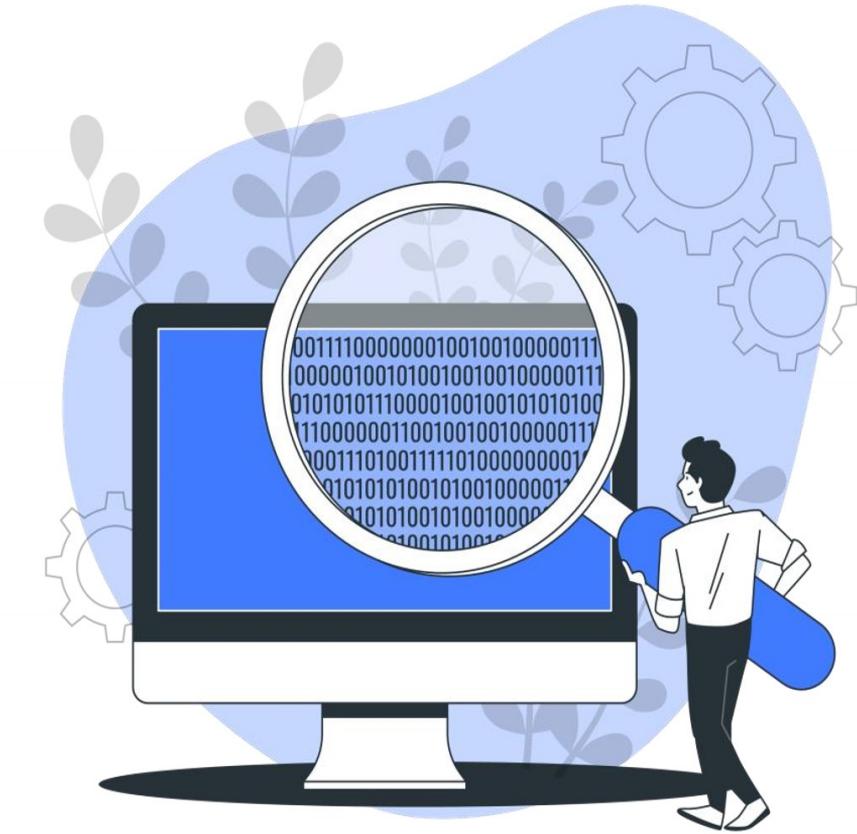
2. Thuật toán BFS - Breadth First Search:



Thuật toán tìm kiếm theo chiều rộng ưu tiên mở rộng các đỉnh gần đỉnh nguồn hơn. Trên đồ thị không có trọng số thì thuật toán BFS cho đường đi ngắn nhất giữa 2 đỉnh.

Độ phức tạp

- $O(V + E)$ nếu sử dụng danh sách kề.
- $O(V * V)$ nếu sử dụng ma trận kề.
- $O(V * E)$ nếu sử dụng danh sách cạnh.





2. Thuật toán BFS - Breadth First Search:



Mã giả:

```
BFS(u){  
    //STEP 1: Khởi tạo  
    queue = Ø; //Tạo một hàng đợi rỗng  
    push(queue, u); // Đẩy đỉnh u vào hàng đợi  
    visited[u] = true; // Đánh dấu đỉnh u đã được thăm  
    //STEP 2: Lặp khi mà hàng đợi vẫn còn phần tử  
    while (queue != Ø){  
        v = queue.front(); // Lấy ra đỉnh ở đầu hàng đợi  
        queue.pop(); //Xóa đỉnh khỏi đầu hàng đợi  
        <Thăm đỉnh v>  
        // Duyệt các đỉnh kề với v mà chưa được thăm và đẩy vào hàng đợi  
        for(int x: ke[v]){  
            if(!visited[x]){ // Nếu x chưa được thăm  
                push(queue, x);  
                visited[x] = true;  
            }  
        }  
    }  
}
```



2. Thuật toán BFS - Breadth First Search:

Code

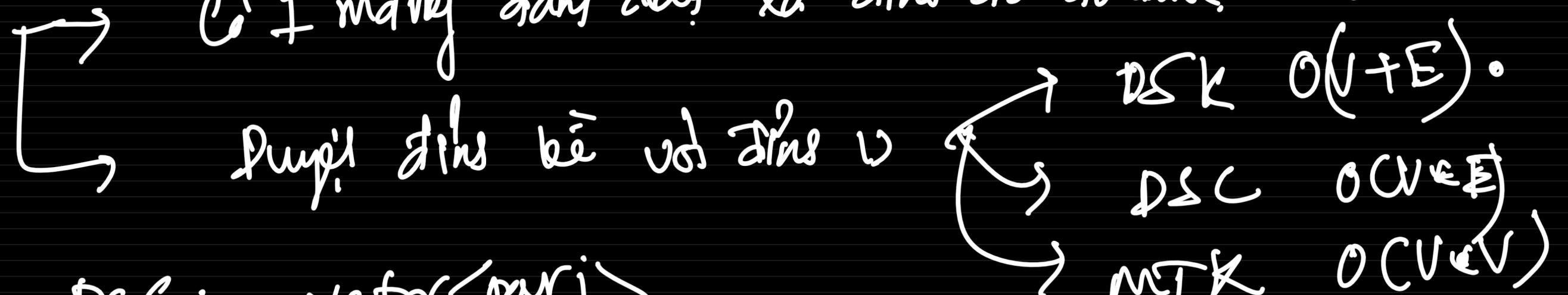
```
int n, m; // đỉnh, cạnh
vector<int> adj[1005]; // danh sách kề
bool visited[1005];

void nhap(){
    cin >> n >> m;
    for(int i = 0; i < m; i++){
        int x, y; cin >> x >> y;
        adj[x].push_back(y);
        adj[y].push_back(x);
    }
    memset(visited, false, sizeof(visited));
}
```

```
void BFS(int u){
    queue<int> q;
    q.push(u);
    visited[u] = true;
    while(!q.empty()){
        int x = q.front(); q.pop();
        cout << x << ' ';
        for(int y : adj[x]){
            if(!visited[y]){
                q.push(y);
                visited[y] = true;
            }
        }
    }
}
```

* think for DFS

→ Có $\frac{1}{2}$ many start day là \rightarrow sinh đồ từ đồ họa



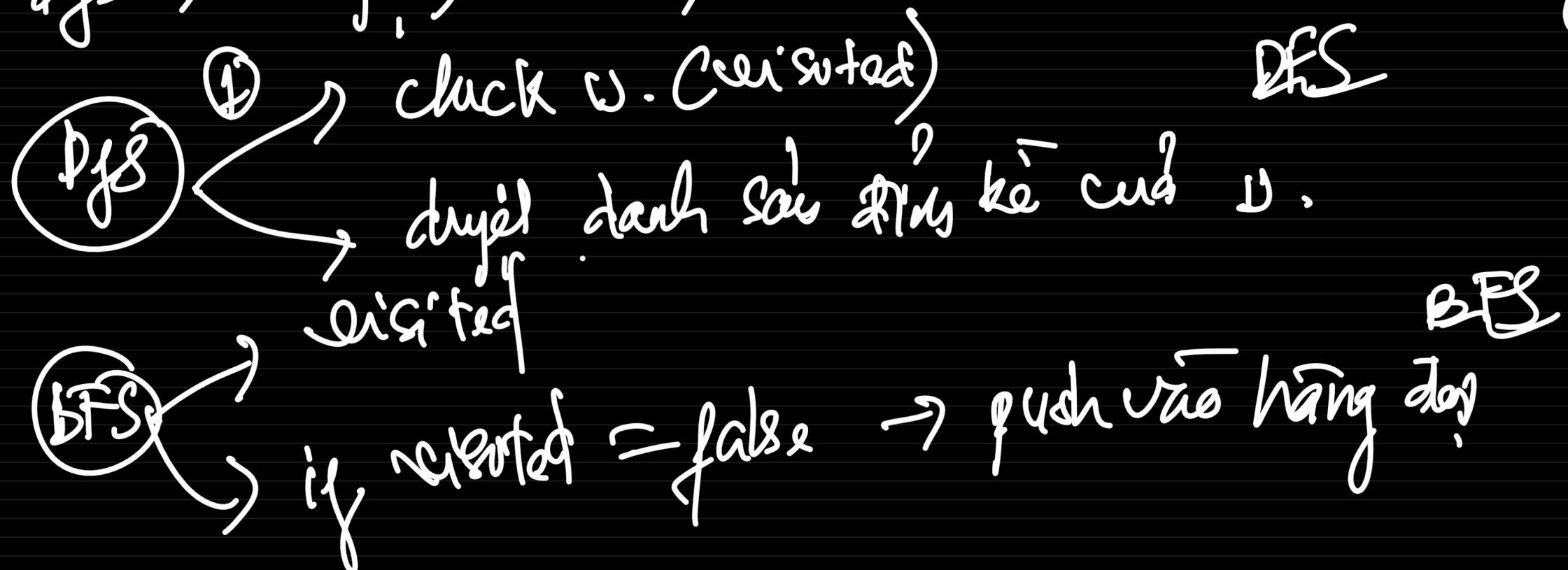
DSC: \rightarrow vector <pari>

\Rightarrow lấy qua file ch phay M và vector

MTK \Rightarrow duyệt cây $a[W][i] \Rightarrow i$ \rightarrow định \bar{d}

① Tìm số thời phay ller không cói \bar{d} thi \Rightarrow dfs
↳ hay cói cói \bar{d} cói \bar{d} thi \Rightarrow step = 1 \Rightarrow AVL.

DFS → dryi, search, tìm kiếm thanh phao kia thang.



DFS và BFS có bát thu đều có visited (true/false)
và dryi als kíp cua dñus U.

