# Iterative DFS

- Use a stack to allow for backtracking during DFS.
- Initialize stack by placing a start vertex $v$.
- As long as stack is nonempty pop the last vertex and mark it visited if it s not visited.
- Then push all the other end vertices of the edges incident on the current vertex.

# Iterative DFS

```
IterativeDFS($G$, $v$) {
// Initialization
index = 0;
T = $\Phi$;
makeNull(S);  // Define an empty stack
for all ($v \in V$)
    mark[$v$] = unvisited;
choose($s$); // Start vertex
S.push($s$);
// Remaining part in next slide
}
```

# Iterative DFS

```
while (!isEmpty(S)) {
    $v$ = S.pop();
    if (marked[$v$] == "unvisited") {
        mark[$v$] = "visited";
        dfn[v] = ++index;
        for all $w \in \mathrm{ADJ}_G(v)$ {
            if (marked[$w$]=="unvisited") {
                S.push($w$);
            }
        }
    }
}
```

# Iterative DFS

- Can have multiple copies vertices on the stack.
- But total number of iterations of stack loop cannot exceed number edges.
- Thus size of stack cannot exceed $|E|$.
- Try out how you can avoid having multiple copies a vertex in the stack.

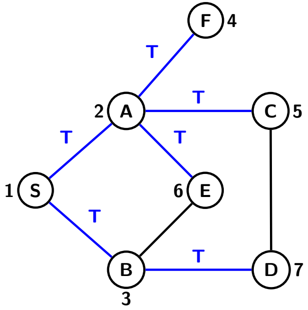
# Breadth First Search

```
 // Initialization
index = 0;
T = $\Phi$;
Q = NULL;
for all ($v \in V$)
    mark[$v$] = "unvisited";
choose($s$); // Start vertex
bfn[$s$] = ++index;
ENQUEUE(Q, $s$);
// Remaining part in next slide
```

# Breadth First Search

```
while (!isEmpty(Q)) {
    $v$ = DEQUEUE(Q);
    mark[$v$] = "visited";
    for all ($w \in$ ADJ$_G$($v$)) {
         if (mark[$w$] == "unvisited") {
             mark[$w$] = "visited";
             T = T $\cup$ $\{(v, w)\}$;
             bfn[$v$] = ++index;
             ENQUEUE(Q, $w$);
         }
    }
}
```

# Breadth First Search



| v | w | Action | Queue |
|---|---|---|---|
| - | - | bfn(S) = 1 | {S} |
| S | A | bfn(A) = 1 | {A} |
|  | B | bfn(B) = 2 | {A,B} |
| A | F | bfn(F) = 4 | {B,F} |
|  | C | bfn(C) = 5 | {B,F,C} |
|  | E | bfn(E) = 6 | {B,F,C,E} |
| B | D | BFN(D) = 7 | {F,C,E,D} |
|  | E | None | {F,C,E,D} |
|  | S | None | {F,C,E,D} |
| F | A | None | {C, E, D} |
| C | A | None | {E, D} |
|  | D | None | {E, D} |
| E | A | None | {D} |
|  | B | None | {D} |
| D | C | None | {} |
|  | B | None | {} |

# Classification of Edges by BFS

- There can be no back edges or forward edges in BFS of undirected graphs.
- For each tree edge $(u, v)$, dist[$v$] = dist[$u$] + 1
- For each cross edge $(u, v)$ dist[$u$] = dist[$v$] or dist[$v$] = dist[$u$] + 1