Depth First Search

Sutanu Gayen

August 2024

1 Pseudocode

Algorithm 1: DFS($G, s, start, finish, \pi, color, clock$)

Input: The adjacency list of a (Directed) graph $G = (V, E)$, A source vertex s, Four arrays $start$, $finish$, $\pi, color$ each indexed by $v \in V$, An integer $clock$

Output: None. The arrays and clock will be updated.

```
S ← an empty stack of MAX value 2(|V| + |E|)
S.Push(s)
while S is not empty do
    u ← S.Peek()
    if color[u] = white then
        start[u] ← clock
        clock++
        color[u] ← gray
        for each (u, w) ∈ E do
            if color[w] = white then
                S.Push(w)
                π[w] ← u
                // (u, w) is a tree/forward edge
            else if color[w] = gray then
                // (u, w) is a backward edge
            else if color[w] = black then
                // (u, w) is a cross/forward edge
    else if color[u] = gray then
        color[u] ← black
        finish[u] ← clock
        clock++
        S.pop()
    else if color[u] = black[a] then
        S.pop()
```

[a] I missed this check in class. You must do this to take care of the case when a vertex appears more than once in the stack.

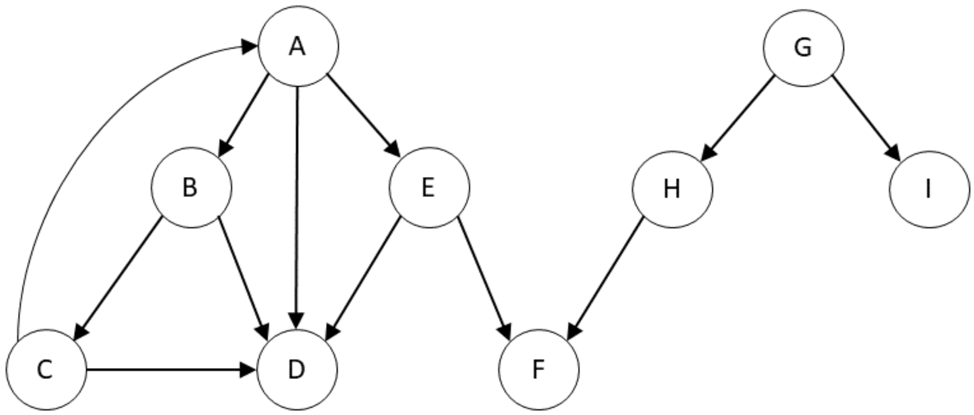

Figure 1: Input Graph

Algorithm 2: DFS-EXPLORE(G)

Input: The adjacency list of a (Directed) graph $G = (V, E)$

Output: Three arrays $start, finish, \pi$ each of size $|V|$

$clock \leftarrow 0$
$start \leftarrow$ all -1 array of size $|V|$
$finish \leftarrow$ all -1 array of size $|V|$
$\pi \leftarrow$ all $NULL$ array of size $|V|$
$color \leftarrow$ all $white$ array of size $|V|$
for $each\ v \in V$ **do**
    if $start[v] = -1$ **then**
        $DFS(G, v, start, finish, \pi, color, clock)$
return $(start, finish, \pi)$

Notes.

1. This is a modification of the basic DFS with two important differences:

   (a) children are popped before the parent is popped

   (b) first and last time seeing are timestamped

   Once you keep these two facts in mind, you can develop the pseudocode with a little creativity.

2. It's possible that a vertex is pushed more than once into the stack. Only the topmost pushing will go through line numbers 5-14. This is also according to the DFS logic that we go deeper and deeper and then backtrack. For any subsequent presence in the stack, line numbers 20-21 will simply throw out the vertex without doing anything.

   Nevertheless, the combination of while and for loop (Line numbers 10-14) at most run once for each edge. Thus the final running time is still $O(n + m)$.

3. π value at line number 12 may keep on updating. Thus tree edges can only be identified after DFS completes. Final value of π will be set via the deepest path which is according to the DFS logic.

2 Exercise

Question: Consider the graph in Figure 1.

1. Write the adjacency list of this graph. A alphabetically higher ordered vertex needs to be written later in the list.
2. Suppose you are given as input the adjacency list from previous question. You run DFS-Explore on this graph. Write down the start, finish, and π array that you are going to get as a result. You must push a alphabetically lower ordered child earlier.
3. Using answer to previous question, draw the dfs tree, the dfs timeline, and perform edge classification

Answer:

1.

$$\begin{array}{llll}
A & B & D & E \\
B & C & D & \\
C & A & D & \\
D & & & \\
E & D & F & \\
F & & & \\
G & H & I & \\
H & F & &
\end{array}$$

2.

vertex	start	finish	π
A	0	11	NULL
B	7	10	A
C	8	9	B
D	4	5	E
E	1	6	A
F	2	3	E
G	12	17	NULL
H	15	16	G
I	13	14	G

Table 1: Returned Values from Explore

3.

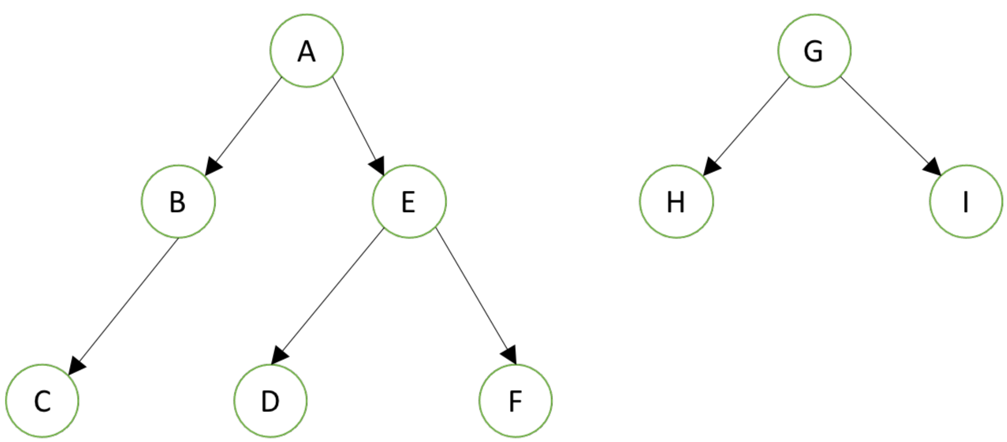

Figure 2: DFS tree

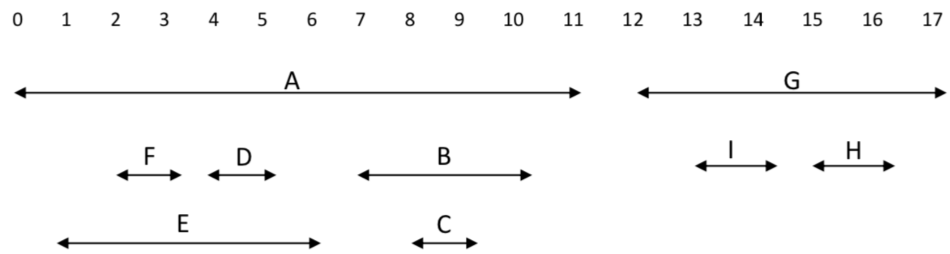

Figure 3: DFS timeline

Edge classification:

- Tree edges: given above
- Forward edges: (A, D)
- Backward edges: (C, A)
- Cross edges: $(C, D), (H, F), (B, D)$