22.3 DEPTH FIRST SEARCH

DEPTH FIRST SEARCH (DFS) IS OFTEN USED AS A SUBROUTINE IN OTHER ALGORITHMS TO DETERMINE THE STRUCTURAL DETAILS OF A GRAPH. IT DOES THIS BY SEARCHING DEEPER INTO A GRAPH WHENEVER POSSIBLE. LIKE BFS, IT REQUIRES VERTICES TO HAVE CERTAIN ATTRIBUTES. FOR EACH $u \in V(G)$:

- color[u] : KEEPS TRACK OF DFS PROGRESS.
- $\pi[u]$: Parent of u, DEFINES PREDECESSOR SUBGRAPH.
- d[u] : DISCOVERY TIME OF u.
- f[u] : FINISH TIME OF u.

THE TIME STAMPS d[u] AND f[u] CAN BE USED BY OTHER ALGORITHMS AND ARE GENERALLY HELPFUL IN REASONING ABOUT THE BEHAVIOR OF DFS.

IN THE FOLLOWING PSEUDO-CODE, time IS A GLOBAL VARIABLE RANGING FROM 1 TO 2n WHERE $n = |V(G)|$. time IS INCREMENTED AT EACH DISCOVERY EVENT (graying) AND AT EACH FINISHING EVENT (blackening). SINCE THERE IS EXACTLY ONE SUCH EVENT FOR EACH VERTEX, THERE ARE NECESSARILY 2n SUCH EVENTS.

DFS(G) CALLS A SUBROUTINE Visit(u) WHICH CALLS ITSELF RECURSIVELY. THE GRAPH $G=(V,E)$ MAY BE DIRECTED OR UNDIRECTED.

DFS(G)

```
1.) for all u ∈ V
2.)     color[u] ← white
3.)     P[u] ← nil
4.) time ← 0
5.) for all u ∈ V
6.)     if color[u] = white
7.)         Visit(u)
```

Visit(u)

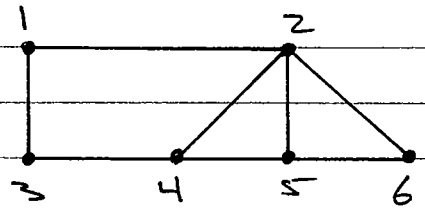
```
1.) color[u] ← gray      (DISCOVERY)
2.) d[u] ← time ← (time + 1)
3.) for all v ∈ Adj[u]
4.)     if color[v] = white
5.)         P[v] ← u
6.)         Visit(v)
7.) color[u] ← black     (FINISH)
8.) f[u] ← time ← (time + 1)
```

THE PREDECESSOR SUBGRAPH (DFS FOREST) G_p IS DEFINED AS $G_p=(V,E_p)$ WHERE

$$E_p=\{(P[u],u)\in E \mid u\in V \text{ AND } P[u]\neq nil\}$$

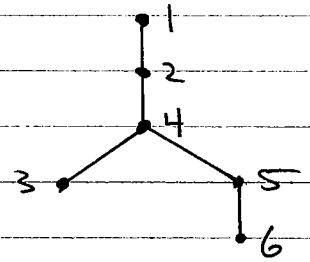
EACH TIME VISIT(u) IS CALLED FROM LINE 7 OF DFS, u BECOMES THE ROOT OF A NEW TREE IN THE DFS FOREST.

EX.



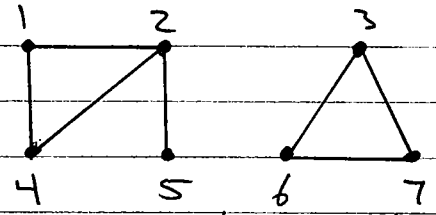
	d	f	p
1 : 2 3	1	12	n
2 : 1 4 5 6	2	11	1
3 : 1 4	4	5	4
4 : 2 3 5	3	10	2
5 : 2 4 6	6	9	4
6 : 2 5	7	8	5

DFS FOREST:



WHEN DFS IS RUN ON AN UNDIRECTED GRAPH G, THE DFS FOREST WILL HAVE AS MANY TREES AS G HAS CONNECTED COMPONENTS. EACH TREE SPANS ONE SUCH COMPONENT.

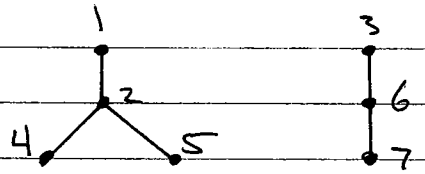
Ex.



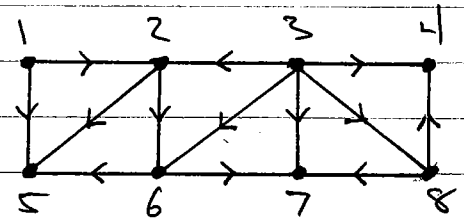
1 : 2 4
2 : 1 4 5
3 : 6 7
4 : 1 2
5 : 2
6 : 3 7
7 : 3 6

	d	f	p
1	1	8	n
2	2	7	1
3	9	14	n
4	3	4	2
5	5	6	2
6	10	13	3
7	11	12	6

DFS FOREST:



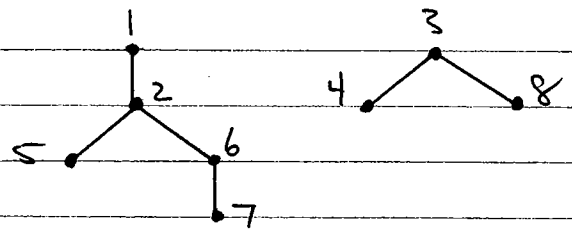
Ex.



1 : 2 5
2 : 5 6
3 : 2 4 6 7 8
4 :
5 :
6 : 5 7
7 :
8 : 4 7

	d	f	p
1	1	10	n
2	2	9	1
3	11	16	n
4	12	13	3
5	3	4	2
6	5	8	2
7	6	7	6
8	14	15	3

DFS FOREST:



Run time

lines 1-3 and 5-6 take time $\Theta(|V|)$ since they process all vertices.

Visit(u) is called exactly once on each vertex, since it is only called on white vertices, and the first thing it does is color a vertex gray. lines 7 in DFS and 6 in Visit are thus executed $|V|$ times.

In Visit(u) loop 3-6 is executed $|Adj[u]|$ times. Recall

$$\sum_u |Adj[u]| = 2|E|$$

By the handshake lemma (in undirected graphs). Thus lines 3-6 take total time $\Theta(|E|)$.

The run time of DFS is therefore $\Theta(|V|+|E|)$.

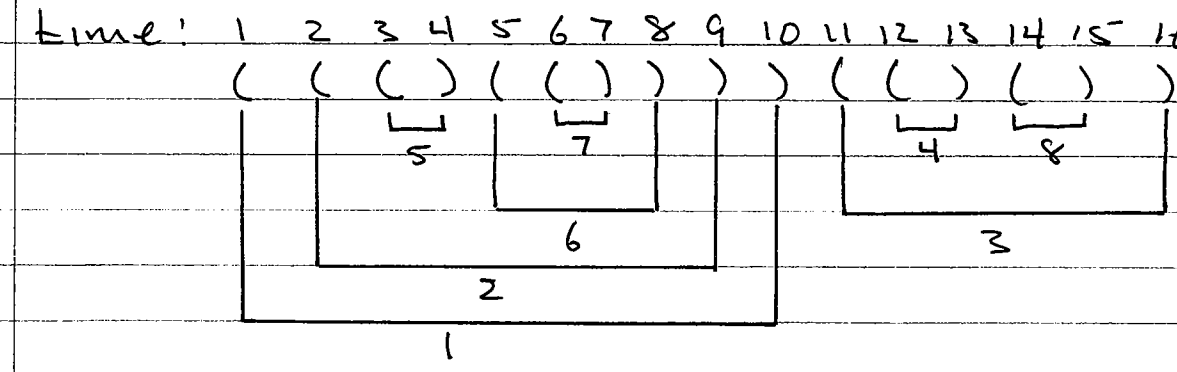
What does DFS(G) tell us about the structure of G? i.e. what can be deduced from the fields d, f, and P after a call to DFS?

First note that the DFS forest G_p exactly mirrors the structure of the recursive calls to Visit().

CREATE A STRING CONSISTING OF THE SYMBOLS '(' AND ')' BY DOING THE FOLLOWING ALGORITHM AFTER A CALL TO DFS.

1.) for $t \leftarrow 1$ TO $2|V|$
2.) if t was A DISCOVERY EVENT
3.) Print '('
4.) else
5.) Print ')'

EX. (FROM PREVIOUS EXAMPLE)



OBSERVE THAT THIS EXPRESSION IS WELL FORMED IN THE SENSE THAT THE PARENTHESES MATCH UP. IN FACT EACH PAIR OF MATCHING PARENTHESES CORRESPONDS TO THE DISCOVER AND FINISH TIMES OF ONE AND THE SAME VERTEX.

THUS THIS STRING CAN BE THOUGHT OF AS THE DFS FOREST G_p IN DIFFERENT NOTATION.