Design and Analysis of Algorithms

instructor: Sharma Thankachan

Lecture 12:
Elementary Graph Algorithms II
About this lecture

• Depth First Search
  • DFS Tree and DFS Forest

• Properties of DFS
  • Parenthesis theorem (very important)
  • White-path theorem (very useful)
Depth First Search (DFS)

• An alternative algorithm to find all vertices reachable from a particular source vertex $s$

• Idea:
  Explore a branch as far as possible before exploring another branch

• Easily done by recursion or stack
The DFS Algorithm

DFS(u)
{
  Mark u as discovered;
  while (u has unvisited neighbor v)
    DFS(v);
  Mark u as finished;
}

The while-loop explores a branch as far as possible before the next branch.
Example \((s = \text{source})\)

- \(r\)
- \(s\)
- \(t\)
- \(u\)
- \(v\)
- \(w\)
- \(x\)
- \(y\)

![Graph](image)

finished

discovered

direction of edge when new node is discovered
Example ($s = \text{source}$)

![Diagram showing the direction of edge when a new node is discovered and finished and discovered nodes.](image-url)
Example \((s = \text{source})\)

- **Finished**: Red nodes
- **Discovered**: Light red nodes
- **Direction of edge when new node is discovered**: Arrows

Steps:
1. Initial state
2. Node \(s\) discovered
3. Node \(t\) discovered
4. Node \(u\) discovered
5. Node \(r\) finished
6. Node \(v\) finished
7. Node \(w\) finished
8. Node \(x\) finished
9. Node \(y\) finished
Example \((s = \text{source})\)

- Finished
- Discovered
- Direction of edge when new node is discovered
Example \((s = \text{source})\)

![Graph Diagram]

- Finished
- Discovered
- Direction of edge when new node is discovered
Example \((s = \text{source})\)

The directed edges form a tree that contains all nodes \textbf{reachable} from \(s\)

\textbf{Called DFS tree of} \(s\)

Done when \(s\) is discovered
Generalization

• Just like BFS, DFS may not visit all the vertices of the input graph $G$, because:
  • $G$ may be disconnected
  • $G$ may be directed, and there is no directed path from $s$ to some vertex

• In most application of DFS (as a subroutine), once DFS tree of $s$ is obtained, we will continue to apply DFS algorithm on any unvisited vertices ...
Suppose the input graph is directed
1. After applying DFS on $s$
Generalization (Example)

2. Then, after applying DFS on $t$
3. Then, after applying DFS on $y$
4. Then, after applying DFS on r
5. Then, after applying DFS on v
Result: a collection of rooted trees called **DFS forest**

```
  r  s
  v   w
```

```
  t   u
```

```
  x
  y
```
Performance

• Since no vertex is discovered twice, and each edge is visited at most twice (why?)
  ➔ Total time: \( O(|V|+|E|) \)

• As mentioned, apart from recursion, we can also perform DFS using a LIFO stack (Do you know how?)
Who will be in the same tree?

- Because we can only explore branches in an unvisited node

=> DFS(u) may not contain all nodes reachable by u in its DFS tree

E.g., in the previous run, v can reach r, s, w, x, but v’s tree does not contain any of them

Can we determine who will be in the same tree?
Who will be in the same tree?

- Yes, we will soon show that by white-path theorem, we can determine who will be in the same tree as \( v \) at the time when DFS is performed on \( v \).

- Before that, we will define the discovery time and finishing time for each node, and show interesting properties of them.
Discovery and Finishing Times

• When the DFS algorithm is run, let us consider a **global time** such that the time increases one unit:
  • when a node is **discovered**, or
  • when a node is **finished**

    (i.e., finished exploring all unvisited neighbors)

• Each node \( u \) records:
  \[ d(u) = \text{the time when } u \text{ is discovered, and} \]
  \[ f(u) = \text{the time when } u \text{ is finished} \]
Discovery and Finishing Times

In our first example (undirected graph)
Discovery and Finishing Times

In our second example (directed graph)
Nice Properties

Lemma: For any node \( u \), \( d(u) < f(u) \)

Lemma: For nodes \( u \) and \( v \),
\[ d(u), d(v), f(u), f(v) \] are all distinct

Theorem (Parenthesis Theorem):
Let \( u \) and \( v \) be two nodes with \( d(u) < d(v) \).
Then, either
1. \( d(u) < d(v) < f(v) < f(u) \) [contain], or
2. \( d(u) < f(u) < d(v) < f(v) \) [disjoint]
Proof of Parenthesis Theorem

• Consider the time when $v$ is discovered
• Since $u$ is discovered before $v$, there are two cases concerning the status of $u$:
  • Case 1: ($u$ is not finished)
    This implies $v$ is a descendant of $u$
    $\Rightarrow f(v) < f(u)$  (why?)
  • Case 2: ($u$ is finished)
    $\Rightarrow f(u) < d(v)$
Corollary:

\[ v \text{ is a (proper) descendant of } u \]

if and only if

\[ d(u) < d(v) < f(v) < f(u) \]

Proof:

\[ v \text{ is a (proper) descendant of } u \]

\[ \iff d(u) < d(v) \text{ and } f(v) < f(u) \]

\[ \iff d(u) < d(v) < f(v) < f(u) \]
White-Path Theorem

Theorem: By the time when DFS is performed on $u$, for any way DFS is done, the descendants of $u$ are the same, and they are exactly those nodes reachable by $u$ with unvisited (white) nodes only.

E.g.,

If we perform DFS($w$) now, will the descendant of $w$ always be the same set of nodes?
Proof (Part 1)

• Suppose that $v$ is a descendant of $u$
  Let $P = (u, w_1, w_2, ..., w_k, v)$ be the directed path from $u$ to $v$ in DFS tree of $u$

Then, apart from $u$, each node on $P$ must be discovered after $u$

⇒ They are all unvisited by the time we perform DFS on $u$

⇒ Thus, at this time, there exists a path from $u$ to $v$ with unvisited nodes only
Proof (Part 2)

• So, every descendant of $u$ is reachable from $u$ with unvisited nodes only.

• To complete the proof, it remains to show the converse:

Any node reachable from $u$ with unvisited nodes only becomes $u$'s descendant is also true

(We shall prove this by contradiction)
Proof (Part 2)

- Suppose on contrary the converse is false
- Then, there exists some $v$, reachable from $u$ with unvisited nodes only, does not become $u$'s descendant
- If more than one choice of $v$, let $v$ be one such vertex closest to $u$

$\Rightarrow \quad d(u) < f(u) < d(v) < f(v) \quad \ldots \text{EQ.1}$
Proof (Part 2)

- Let $P = (u, w_1, w_2, ..., w_k, v)$ be any path from $u$ to $v$ using unvisited nodes only.
- By our choice of $v$ (closest one), all $w_1, w_2, ..., w_k$ become $u$'s descendants.
- This implies:
  \[
  d(u) \leq d(w_k) < f(w_k) \leq f(u)
  \]
- Combining with EQ.1, we have
  \[
  d(w_k) < f(w_k) < d(v) < f(v)
  \]
Proof (Part 2)

• However, since there is an edge (no matter undirected or directed) from \( w_k \) to \( v \), if \( d(w_k) < d(v) \), then we must have \( d(v) < f(w_k) \) ... (why??)

• Consequently, it contradicts with:
  \[
  d(w_k) < f(w_k) < d(v) < f(v)
  \]

\( \rightarrow \) Proof completes