

Assignment 1

Yulei Sui

University of Technology Sydney, Australia

Assignment 1: Quizzes + A Coding Task

- Two sets of quizzes (20 ponts)
 - Basic C/C++ and secure programming
 - Vulnerability assessment.
- One coding task (10 ponts)
 - Practicing C++ graph traversal algorithm
 - A warm up coding task for later assignments.

You are encouraged to finish the quizzes before starting your coding task.

Assignment 1: C++ Coding Task

Graph Traversal

- You will be using what you have learned to build a C++ program.
- **Goal:** implement a depth first search on a graph and print path from a source node to a sink node on the graph

Assignment 1: C++ Coding Task

Graph Traversal

- You will be using what you have learned to build a C++ program.
- **Goal:** implement a depth first search on a graph and print path from a source node to a sink node on the graph
- **Specification and code template:** <https://github.com/SVF-tools/Teaching-Software-Verification/wiki/Assignment-1>

Assignment 1: C++ Coding Task

Graph Traversal

- You will be using what you have learned to build a C++ program.
- **Goal:** implement a depth first search on a graph and print path from a source node to a sink node on the graph
- **Specification and code template:** <https://github.com/SVF-tools/Teaching-Software-Verification/wiki/Assignment-1>

Depth First Search (DFS)

- An algorithm to traverse or search a graph data structure.
- Exploring as far as possible along each branch before backtracking.

Assignment 1: C++ Coding Task

Graph Traversal

- You will be using what you have learned to build a C++ program.
- **Goal:** implement a depth first search on a graph and print path from a source node to a sink node on the graph
- **Specification and code template:** <https://github.com/SVF-tools/Teaching-Software-Verification/wiki/Assignment-1>

Depth First Search (DFS)

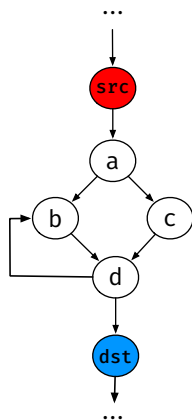
- An algorithm to traverse or search a graph data structure.
- Exploring as far as possible along each branch before backtracking.

Why DFS?

- Efficient, linear time complexity, i.e., $O(V+E)$, where V is nodes and E is edges.
- One of the most commonly used graph algorithms.

Assignment 1: C++ Coding Task

Graph Traversal

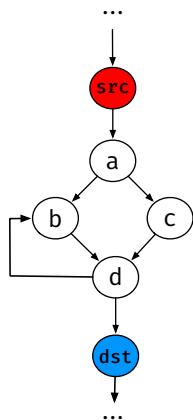


Given a source node `src` and a destination node `dst` on a graph

- (1) can `src` reach `dst`?
- (2) if so, what are the possible paths from `src` to `dst` along the graph?

Assignment 1: C++ Coding Task

Graph Traversal



Given a source node `src` and a destination node `dst` on a graph

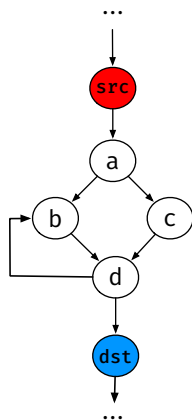
- (1) can `src` reach `dst`?
- (2) if so, what are the possible paths from `src` to `dst` along the graph?

Answer:

- (1) Yes.

Assignment 1: C++ Coding Task

Graph Traversal



Given a source node `src` and a destination node `dst` on a graph

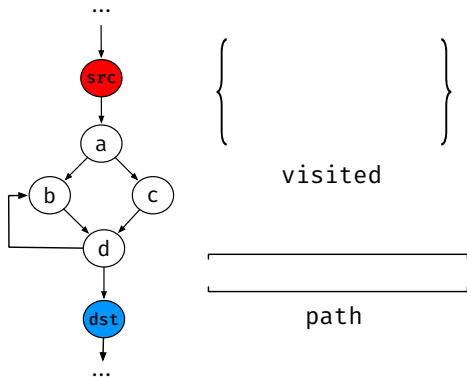
- (1) can `src` reach `dst`?
- (2) if so, what are the possible paths from `src` to `dst` along the graph?

Answer:

- (1) Yes.
- (2) All possible paths:
 - `src` \rightarrow `a` \rightarrow `b` \rightarrow `d` \rightarrow `dst`
 - `src` \rightarrow `a` \rightarrow `c` \rightarrow `d` \rightarrow `dst`
 - `src` \rightarrow `a` \rightarrow `b` \rightarrow `d` \rightarrow `b` \rightarrow `d` \rightarrow `dst`
 - `src` \rightarrow `a` \rightarrow `b` \rightarrow `d` \rightarrow `b` \rightarrow `d` \rightarrow ... `dst`

Assignment 1: C++ Coding Task

DFS algorithm and an example

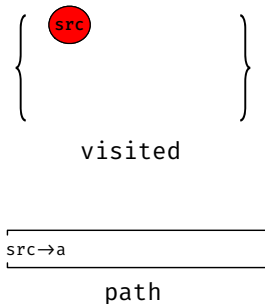
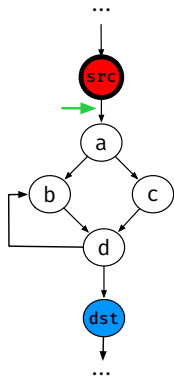


```
//mark the visited node
visited: set<Node*>
//edge seq in the current path during traversal
path: vector<Edge*>

DFS(src_edge, dst)
1  curNode = src_edge.getDst();
2  visited.insert(curNode);
3  path.push_back(src_edge);
4  if src == dst then
5    Print path; //Print node seq of current path
6  foreach edge e ∈ outEdges(src) do
7    nxtNode = e.getDst();
8    if (nxtNode ∉ visited)
9      DFS(visited, path, e, dst);
10 visited.erase(curNode);
11 path.pop_back();
```

Assignment 1: C++ Coding Task

DFS algorithm and an example

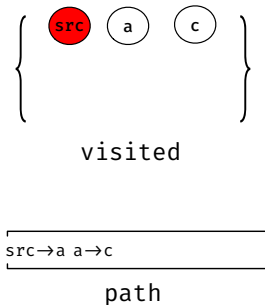
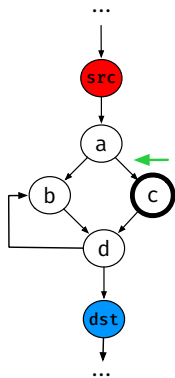


```
//mark the visited node
visited: set<Node*>
//edge seq in the current path during traversal
path: vector<Edge*>

DFS(src_edge, dst)
1  curNode = src_edge.getDst();
2  visited.insert(curNode);
3  path.push_back(src_edge);
4  if src == dst then
5    Print path; //Print node seq of current path
6  foreach edge e ∈ outEdges(src) do
7    nxtNode = e.getDst();
8    if (nxtNode ∉ visited)
9      DFS(visited, path, e, dst);
10 visited.erase(curNode);
11 path.pop_back();
```

Assignment 1: C++ Coding Task

DFS algorithm and an example

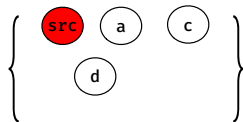
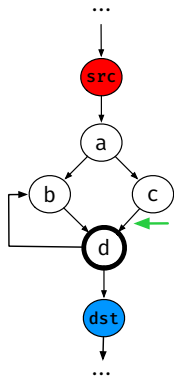


```
//mark the visited node
visited: set<Node*>
//edge seq in the current path during traversal
path: vector<Edge*>

DFS(src_edge, dst)
1  curNode = src_edge.getDst();
2  visited.insert(curNode);
3  path.push_back(src_edge);
4  if src == dst then
5    Print path; //Print node seq of current path
6  foreach edge e ∈ outEdges(src) do
7    nxtNode = e.getDst();
8    if (nxtNode ∉ visited)
9      DFS(visited, path, e, dst);
10 visited.erase(curNode);
11 path.pop_back();
```

Assignment 1: C++ Coding Task

DFS algorithm and an example



visited

src → a a → c c → d

path

```
//mark the visited node
```

```
visited: set<Node*>
```

```
//edge seq in the current path during traversal
```

```
path: vector<Edge*>
```

```
DFS(src_edge, dst)
```

```
1 curNode = src_edge.getDst();
```

```
2 visited.insert(curNode);
```

```
3 path.push_back(src_edge);
```

```
4 if src == dst then
```

```
5     Print path; //Print node seq of current path
```

```
6 foreach edge e ∈ outEdges(src) do
```

```
7     nxtNode = e.getDst();
```

```
8     if (nxtNode ∉ visited)
```

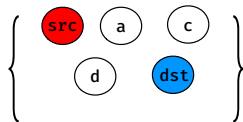
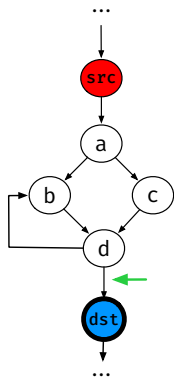
```
● 9         DFS(visited, path, e, dst);
```

```
10 visited.erase(curNode);
```

```
11 path.pop_back();
```

Assignment 1: C++ Coding Task

DFS algorithm and an example



visited

src → a → c → d → dst

path



OUTPUT <src → a → c → d → dst>

```
//mark the visited node
```

```
visited: set<Node*>
```

```
//edge seq in the current path during traversal
```

```
path: vector<Edge*>
```

```
DFS(src_edge, dst)
```

```
1 curNode = src_edge.getDst();
```

```
2 visited.insert(curNode);
```

```
3 path.push_back(src_edge);
```

```
4 if src == dst then
```

```
• 5   Print path; //Print node seq of current path
```

```
6 foreach edge e ∈ outEdges(src) do
```

```
7   nxtNode = e.getDst();
```

```
8   if (nxtNode ∉ visited)
```

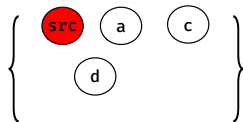
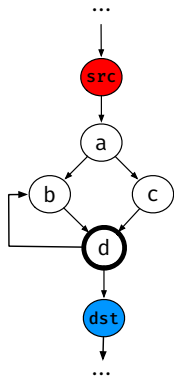
```
9     DFS(visited, path, e, dst);
```

```
10 visited.erase(curNode);
```

```
11 path.pop_back();
```

Assignment 1: C++ Coding Task

DFS algorithm and an example



visited

src → a a → c c → d

path

```
//mark the visited node
```

```
visited: set<Node*>
```

```
//edge seq in the current path during traversal
```

```
path: vector<Edge*>
```

```
DFS(src_edge, dst)
```

```
1 curNode = src_edge.getDst();
```

```
2 visited.insert(curNode);
```

```
3 path.push_back(src_edge);
```

```
4 if src == dst then
```

```
5   Print path; //Print node seq of current path
```

```
6 foreach edge e ∈ outEdges(src) do
```

```
7   nxtNode = e.getDst();
```

```
8   if (nxtNode ∉ visited)
```

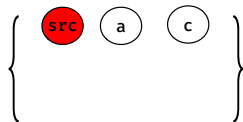
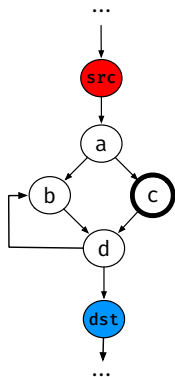
```
9     DFS(visited, path, e, dst);
```

```
10  visited.erase(curNode);
```

```
11  path.pop_back();
```

Assignment 1: C++ Coding Task

DFS algorithm and an example



visited

src → a a → c

path

```
//mark the visited node
```

```
visited: set<Node*>
```

```
//edge seq in the current path during traversal
```

```
path: vector<Edge*>
```

```
DFS(src_edge, dst)
```

```
1 curNode = src_edge.getDst();
```

```
2 visited.insert(curNode);
```

```
3 path.push_back(src_edge);
```

```
4 if src == dst then
```

```
5   Print path; //Print node seq of current path
```

```
6 foreach edge e ∈ outEdges(src) do
```

```
7   nxtNode = e.getDst();
```

```
8   if (nxtNode ∉ visited)
```

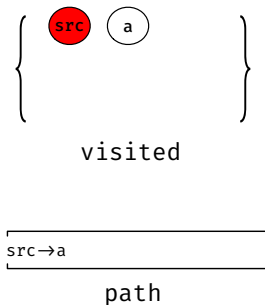
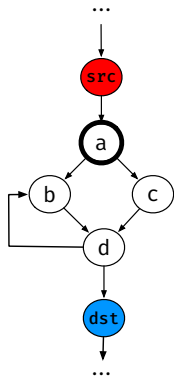
```
9     DFS(visited, path, e, dst);
```

```
10  visited.erase(curNode);
```

```
11  path.pop_back();
```

Assignment 1: C++ Coding Task

DFS algorithm and an example

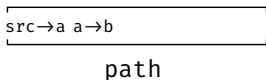
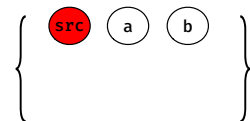
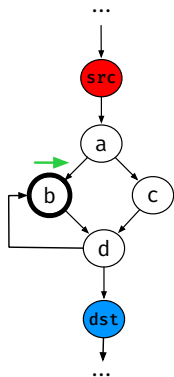


```
//mark the visited node
visited: set<Node*>
//edge seq in the current path during traversal
path: vector<Edge*>

DFS(src_edge, dst)
1  curNode = src_edge.getDst();
2  visited.insert(curNode);
3  path.push_back(src_edge);
4  if src == dst then
5    Print path; //Print node seq of current path
6  foreach edge e ∈ outEdges(src) do
7    nxtNode = e.getDst();
8    if (nxtNode ∉ visited)
9      DFS(visited, path, e, dst);
10 visited.erase(curNode);
11 path.pop_back();
```

Assignment 1: C++ Coding Task

DFS algorithm

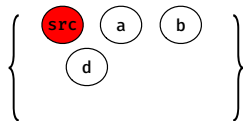
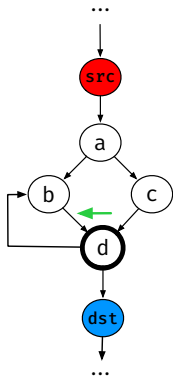


```
//mark the visited node
visited: set<Node*>
//edge seq in the current path during traversal
path: vector<Edge*>

DFS(src_edge, dst)
1  curNode = src_edge.getDst();
2  visited.insert(curNode);
3  path.push_back(src_edge);
4  if src == dst then
5    Print path; //Print node seq of current path
6  foreach edge e ∈ outEdges(src) do
7    nxtNode = e.getDst();
8    if (nxtNode ∉ visited)
9      DFS(visited, path, e, dst);
10 visited.erase(curNode);
11 path.pop_back();
```

Assignment 1: C++ Coding Task

DFS algorithm and an example



visited

src → a a → b b → d

path

```
//mark the visited node
```

```
visited: set<Node*>
```

```
//edge seq in the current path during traversal
```

```
path: vector<Edge*>
```

```
DFS(src_edge, dst)
```

```
1 curNode = src_edge.getDst();
```

```
2 visited.insert(curNode);
```

```
3 path.push_back(src_edge);
```

```
4 if src == dst then
```

```
5   Print path; //Print node seq of current path
```

```
6 foreach edge e ∈ outEdges(src) do
```

```
• 7   nxtNode = e.getDst();
```

```
8   if (nxtNode ∉ visited)
```

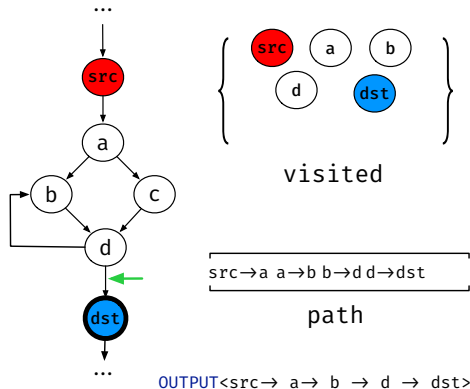
```
9     DFS(visited, path, e, dst);
```

```
10  visited.erase(curNode);
```

```
11  path.pop_back();
```

Assignment 1: C++ Coding Task

DFS algorithm and an example



```
//mark the visited node  
visited: set<Node*>  
//edge seq in the current path during traversal  
path: vector<Edge*>
```

```
DFS(src_edge, dst)  
1 curNode = src_edge.getDst();  
2 visited.insert(curNode);  
3 path.push_back(src_edge);  
4 if src == dst then  
5   Print path; //Print node seq of current path  
6 foreach edge e ∈ outEdges(src) do  
7   nxtNode = e.getDst();  
8   if (nxtNode ∉ visited)  
9     DFS(visited, path, e, dst);  
10 visited.erase(curNode);  
11 path.pop_back();
```