Assignment 1

Yulei Sui

University of Technology Sydney, Australia

Assignment 1: A C++ Programming Practice

Graph Traversal

- You will be using what you have learned to conduct a C++ programming practice.
- **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: A C++ Programming Practice
Graph Traversal

- You will be using what you have learned to conduct a C++ programming practice.
- **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/SVF-Teaching/wiki/Assignment-1

Assignment 1: A C++ Programming Practice

Graph Traversal

- You will be using what you have learned to conduct a C++ programming practice.
- **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/SVF-Teaching/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: A C++ Programming Practice

Graph Traversal

- You will be using what you have learned to conduct a C++ programming practice.
- **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/SVF-Teaching/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: A C++ Programming Practice
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
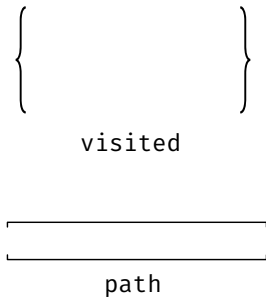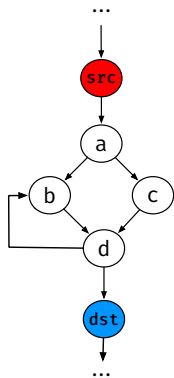 `src` to `dst` along the graph?

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: A C++ Programming Practice
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` → `a` → `b` → `d` → `dst`
  - `src` → `a` → `c` → `d` → `dst`
  - `src` → `a` → `b` → `d` → `b` → `d` → `dst`
  - `src` → `a` → `b` → `d` → `b` → `d` → . . . `dst`

Assignment 1: A C++ Programming Practice
DFS algorithm and an example


```
//mark the visited node
visited: set<NodeID>
//node seq in the current path during traversal
path: vector<NodeID>

DFS(visited, path, src, dst)
1   visited ← visited ∪ {src};
2   path.push_back(src);
3   if src = dst then
4     Print path; //Print node seq of current path
5   foreach edge e ∈ outEdges(src) do
6     if (e.dst ∉ visited)
7         DFS(visited, path, e.dst, dst);
8   visited.erase(src);
9   path.pop_back();
```
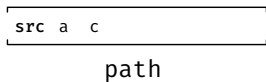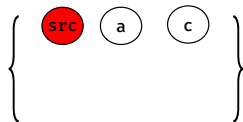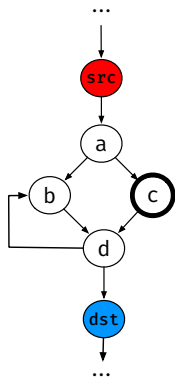
Assignment 1: A C++ Programming Practice

DFS algorithm and an example


```
//mark the visited node
visited: set<NodeID>
//node seq in the current path during traversal
path: vector<NodeID>

DFS(visited, path, src, dst)
1   visited ← visited ∪ {src};
2   path.push_back(src);
3   if src = dst then
4       Print path; //Print node seq of current path
5   foreach edge e ∈ outEdges(src) do
6       if (e.dst ∉ visited)
7           DFS(visited, path, e.dst, dst);
8   visited.erase(src);
9   path.pop_back();
```
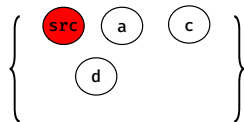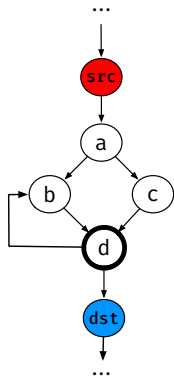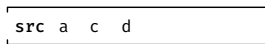
Assignment 1: A C++ Programming Practice

DFS algorithm and an example

visited

src a c

path

```
//mark the visited node
visited: set<NodeID>
//node seq in the current path during traversal
path: vector<NodeID>

DFS(visited, path, src, dst)
1  visited ← visited ∪ {src};
2  path.push_back(src);
3  if src = dst then
4    Print path; //Print node seq of current path
5  foreach edge e ∈ outEdges(src) do
6    if (e.dst ∉ visited)
7      DFS(visited, path, e.dst, dst);
8  visited.erase(src);
9  path.pop_back();
```
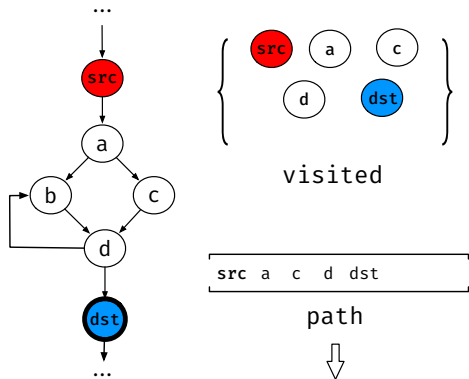
Assignment 1: A C++ Programming Practice
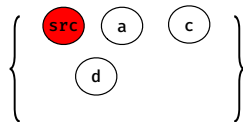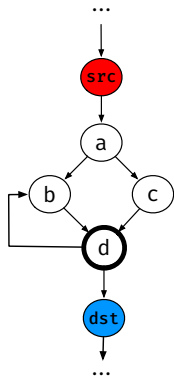## DFS algorithm and an example


```
//mark the visited node
visited: set<NodeID>
//node seq in the current path during traversal
path: vector<NodeID>

DFS(visited, path, src, dst)
1   visited ← visited ∪ {src};
2   path.push_back(src);
3   if src = dst then
4     Print path; //Print node seq of current path
5   foreach edge e ∈ outEdges(src) do
6     if (e.dst ∉ visited)
7       DFS(visited, path, e.dst, dst);
8   visited.erase(src);
9   path.pop_back();
```

DFS algorithm and an example


```
//mark the visited node
visited: set<NodeID>
//node seq in the current path during traversal
path: vector<NodeID>

DFS(visited, path, src, dst)
1  visited ← visited ∪ {src};
2  path.push_back(src);
3  if src = dst then
4    Print path; //Print node seq of current path
5  foreach edge e ∈ outEdges(src) do
6    if (e.dst ∉ visited)
7       DFS(visited, path, e.dst, dst);
8  visited.erase(src);
9  path.pop_back();
```
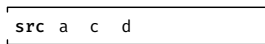
visited

path

src a c d dst

⇩

OUTPUT<src→ a→ c → d → dst>

Assignment 1: A C++ Programming Practice

DFS algorithm and an example

visited

path

```
//mark the visited node
visited: set<NodeID>
//node seq in the current path during traversal
path: vector<NodeID>

DFS(visited, path, src, dst)
1   visited ← visited ∪ {src};
2   path.push_back(src);
3   if src = dst then
4     Print path; //Print node seq of current path
5   foreach edge e ∈ outEdges(src) do
6     if (e.dst ∉ visited)
7         DFS(visited, path, e.dst, dst);
8   visited.erase(src);
9   path.pop_back();
```
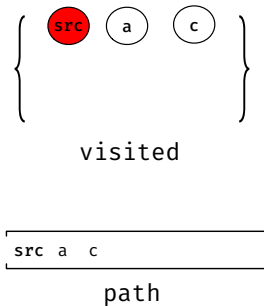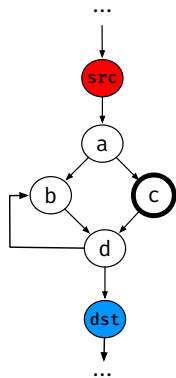
Assignment 1: A C++ Programming Practice
DFS algorithm and an example
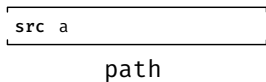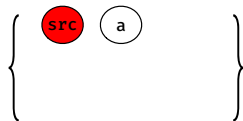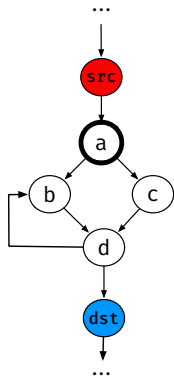


```
//mark the visited node
visited: set<NodeID>
//node seq in the current path during traversal
path: vector<NodeID>

DFS(visited, path, src, dst)
1  visited ← visited ∪ {src};
2  path.push_back(src);
3  if src = dst then
4    Print path; //Print node seq of current path
5  foreach edge e ∈ outEdges(src) do
6    if (e.dst ∉ visited)
7      DFS(visited, path, e.dst, dst);
8  visited.erase(src);
9  path.pop_back();
```

Assignment 1: A C++ Programming Practice
DFS algorithm and an example


```
//mark the visited node
visited: set<NodeID>
//node seq in the current path during traversal
path: vector<NodeID>

DFS(visited, path, src, dst)
1   visited ← visited ∪ {src};
2   path.push_back(src);
3   if src = dst then
4     Print path; //Print node seq of current path
5   foreach edge e ∈ outEdges(src) do
6     if (e.dst ∉ visited)
7         DFS(visited, path, e.dst, dst);
8   visited.erase(src);
9   path.pop_back();
```
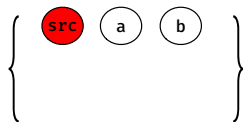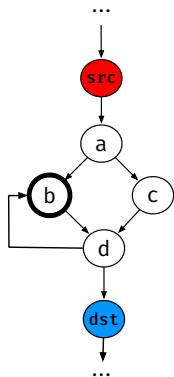
Assignment 1: A C++ Programming Practice
DFS algorithm


```
//mark the visited node
visited: set<NodeID>
//node seq in the current path during traversal
path: vector<NodeID>

DFS(visited, path, src, dst)
1  visited ← visited ∪ {src};
2  path.push_back(src);
3  if src = dst then
4    Print path; //Print node seq of current path
5  foreach edge e ∈ outEdges(src) do
6      if (e.dst ∉ visited)
7          DFS(visited, path, e.dst, dst);
8  visited.erase(src);
9  path.pop_back();
```
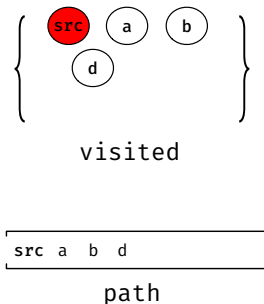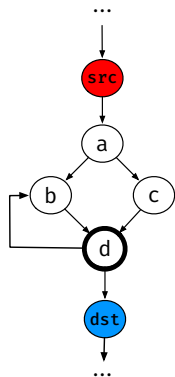
Assignment 1: A C++ Programming Practice
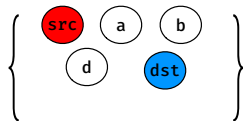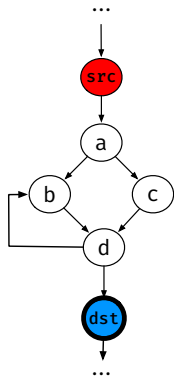## DFS algorithm and an example


```
//mark the visited node
visited: set<NodeID>
//node seq in the current path during traversal
path: vector<NodeID>

DFS(visited, path, src, dst)
1   visited ← visited ∪ {src};
2   path.push_back(src);
3   if src = dst then
4     Print path; //Print node seq of current path
5   foreach edge e ∈ outEdges(src) do
6     if (e.dst ∉ visited)
7       DFS(visited, path, e.dst, dst);
8   visited.erase(src);
9   path.pop_back();
```

Assignment 1: A C++ Programming Practice
DFS algorithm and an example

visited

src a b d dst

path

```
//mark the visited node
visited: set<NodeID>
//node seq in the current path during traversal
path: vector<NodeID>

DFS(visited, path, src, dst)
1   visited ← visited ∪ {src};
2   path.push_back(src);
3   if src = dst then
4     Print path; //Print node seq of current path
5   foreach edge e ∈ outEdges(src) do
6     if (e.dst ∉ visited)
7         DFS(visited, path, e.dst, dst);
8   visited.erase(src);
9   path.pop_back();
```

14