Graph Algorithms: Deapth First Search(DFS)

Nirob Arefin [1] and Protik Dey [2]

[1] Student-ID: 1505050
[2] Student-ID: 1505051

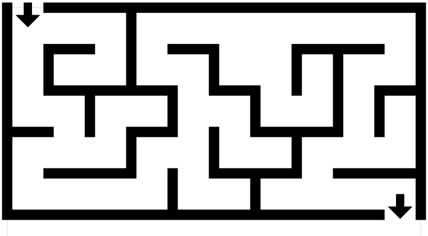
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

July 16, 2018

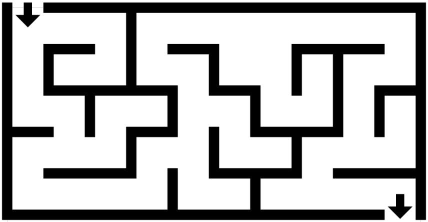
Outline

Outline

Introduction



Introduction



- **It's a maze!!!**

Deapth First Search(DFS)

Introduction

What is DFS?

- Algorithm for traversing graph data structures.
- Starts at the root node.
- Explores as far as possible along each branch before backtracking.

Introduction

What is DFS?

- Algorithm for traversing graph data structures.
- Starts at the root node.
- Explores as far as possible along each branch before backtracking.

Introduction

What is DFS?

- Algorithm for traversing graph data structures.
- Starts at the root node.
- Explores as far as possible along each branch before backtracking.

What is DFS?

- Algorithm for traversing graph data structures.
- Starts at the root node.
- Explores as far as possible along each branch before backtracking.

Outline

Pseudocode

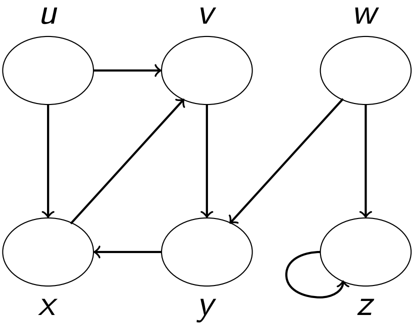
```
DFS(G)
for  each vertex u ∈G.V do
    u.color = WHITE
    u.π = NIL
end
time = 0
for each vertex u ∈G.V do
    if u.color == WHITE
    DFS-VISIT(G, u)
end
```

Pseudocode

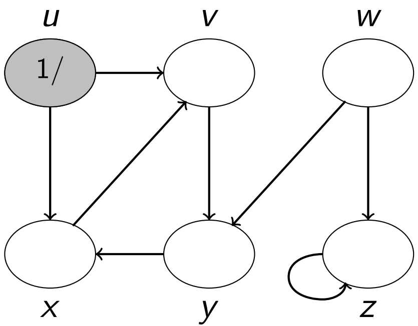
```
DFS-VISIT(G, u)
time = time +1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u] do
    if v.color == WHITE
    v.π = u
    DFS-VISIT(G, v)
end
u.color = BLACK
time = time+1
u.f=time
```

Outline

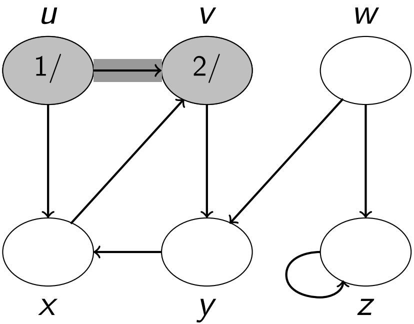
Example



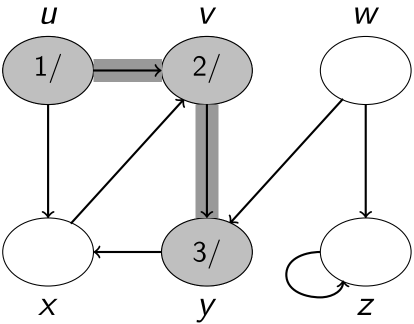
Example



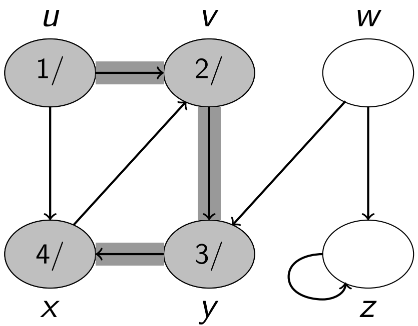
Example



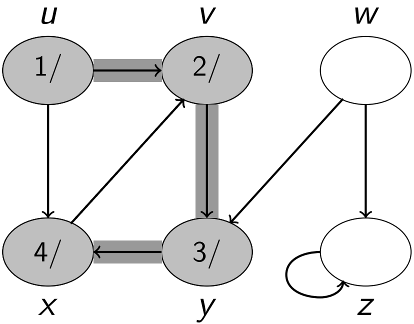
Example



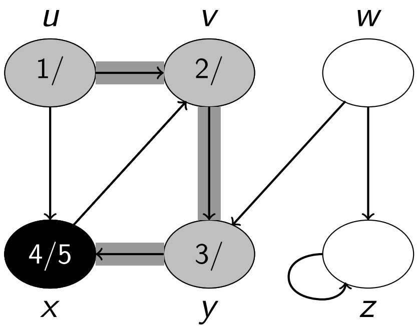
Example



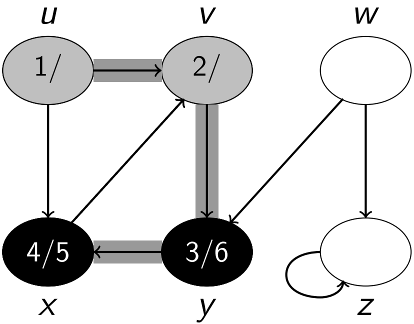
Example



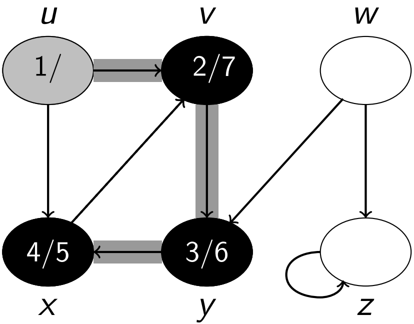
Example



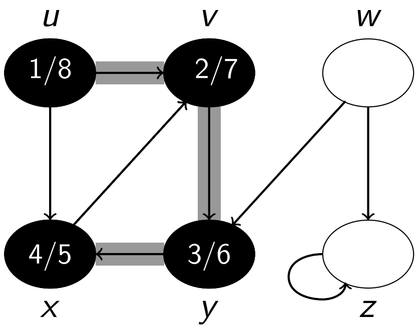
Example



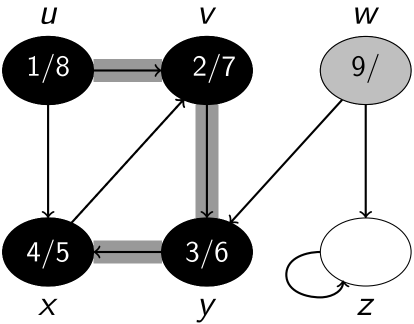
Example



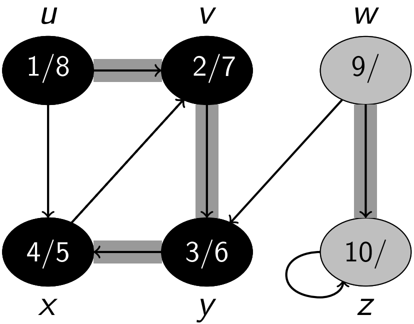
Example



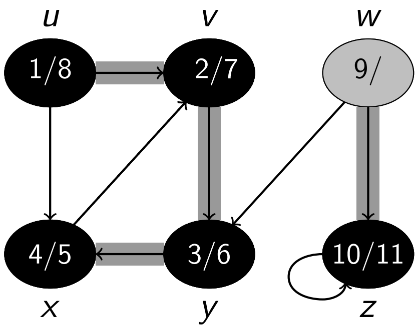
Example



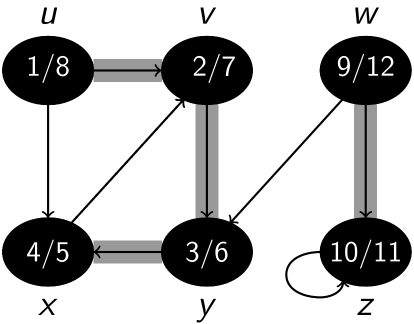
Example



Example



Example



Outline

Complexity

```
DFS(G)
for each vertex u ∈ G.V do
    u.color = WHITE
    u.π = NIL
end
time = 0
for each vertex u ∈ G.V do
    if u.color == WHITE
    DFS-VISIT(G, u)
end
```

```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u] do
    if v.color == WHITE
    v.π = u
    DFS-VISIT(G, v)
end
u.color = BLACK
time = time+1
u.f=time
```

Complexity

```
1  DFS(G)
2  for each vertex u ∈ G.V do
3      u.color = WHITE
4      u.π = NIL
5  end
6  time = 0
7  for each vertex u ∈ G.V do
8      if u.color == WHITE
9      DFS-VISIT(G, u)
10 end
```

- Loop of line 2-5 take $\Theta(V)$
- Loop of line 7-10 take $\Theta(V)$

Complexity

```
1  DFS(G)
2  for each vertex u ∈ G.V
     do
3      u.color = WHITE
4      u.π = NIL
5  end
6  time = 0
7  for each vertex u ∈ G.V
     do
8      if u.color == WHITE
9      DFS-VISIT(G, u)
10 end
```

- Loop of line 2-5 take $\Theta(V)$
- Loop of line 7-10 take $\Theta(V)$

Complexity

```
1  DFS-VISIT(G, u)
2  time = time +1
3  u.d = time
4  u.color = GRAY
5  for each v ∈ G.Adj[u] do
6  |   if v.color == WHITE
7  |   v.π = u
8  |   DFS-VISIT(G, v)
9  end
10 u.color = BLACK
11 time = time+1
12 u.f=time
```

- Loop of line 5-9 executes $|Adj[v]|$ times.
- The preocedure DFS-VISIT is called exactly once for each vertex $v \epsilon V$.
- Total cost of executing lines 5-9 of DFS-VISIT is :
- $$\sum_{v} |Adj[v]| = \Theta(E)$$

Complexity

```
1  DFS-VISIT(G, u)
2  time = time +1
3  u.d = time
4  u.color = GRAY
5  for each v ∈ G.Adj[u] do
6  |   if v.color == WHITE
7  |   v.π = u
8  |   DFS-VISIT(G, v)
9  end
10 u.color = BLACK
11 time = time+1
12 u.f=time
```

- Loop of line 5-9 executes $|Adj[v]|$ times.
- The preocedure DFS-VISIT is called exactly once for each vertex $v \epsilon V$.
- Total cost of executing lines 5-9 of DFS-VISIT is :
- $$\sum_{v} |Adj[v]| = \Theta(E)$$

Complexity

```
1  DFS-VISIT(G, u)
2  time = time +1
3  u.d = time
4  u.color = GRAY
5  for each v ∈ G.Adj[u] do
6  |   if v.color == WHITE
7  |   v.π = u
8  |   DFS-VISIT(G, v)
9  end
10 u.color = BLACK
11 time = time+1
12 u.f=time
```

- Loop of line 5-9 executes $|Adj[v]|$ times.
- The preocedure DFS-VISIT is called exactly once for each vertex $v \epsilon V$.
- Total cost of executing lines 5-9 of DFS-VISIT is :
- $$\sum_{v} |Adj[v]| = \Theta(E)$$

Complexity

```
1  DFS-VISIT(G, u)
2  time = time +1
3  u.d = time
4  u.color = GRAY
5  for each v ∈ G.Adj[u] do
6  |   if v.color == WHITE
7  |   v.π = u
8  |   DFS-VISIT(G, v)
9  end
10 u.color = BLACK
11 time = time+1
12 u.f=time
```

- Loop of line 5-9 executes $|Adj[v]|$ times.
- The preocedure DFS-VISIT is called exactly once for each vertex $v \epsilon V$.
- Total cost of executing lines 5-9 of DFS-VISIT is :
- $$\sum_{v} |Adj[v]| = \Theta(E)$$

Complexity

Finally

The running time of DFS is therefore $\Theta(V + E)$

Outline

Application

- Detecting cycle in a graph
- Path finding
- Topological sorting
- Finding strongly connected components
- Solving mazes

Application

- Detecting cycle in a graph
- Path finding
- Topological sorting
- Finding strongly connected components
- Solving mazes

Application

- Detecting cycle in a graph
- Path finding
- Topological sorting
- Finding strongly connected components
- Solving mazes

Application

- Detecting cycle in a graph
- Path finding
- Topological sorting
- Finding strongly connected components
- Solving mazes

Application

- Detecting cycle in a graph
- Path finding
- Topological sorting
- Finding strongly connected components
- Solving mazes

Topological Sorting

Topological Sorting

Procedure

- call DFS(G) to compute finishing time $v.f$ for each vertex v
- as each vertex is finished,insert it onto the front of a linked list
- **return** the linked list of vertices

Topological Sorting

Procedure

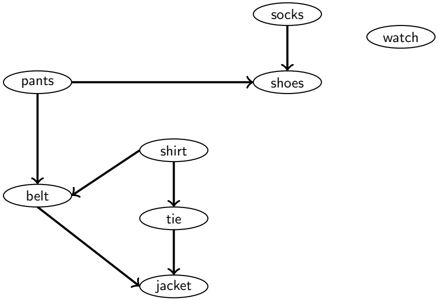
- call DFS(G) to compute finishing time $v.f$ for each vertex v
- as each vertex is finished,insert it onto the front of a linked list
- **return** the linked list of vertices

Topological Sorting

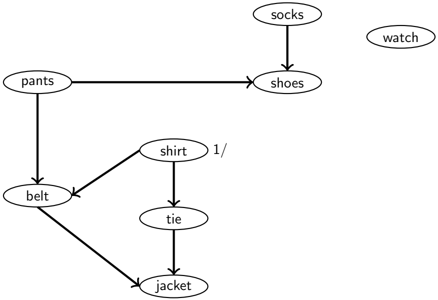
Procedure

- call DFS(G) to compute finishing time $v.f$ for each vertex v
- as each vertex is finished,insert it onto the front of a linked list
- **return** the linked list of vertices

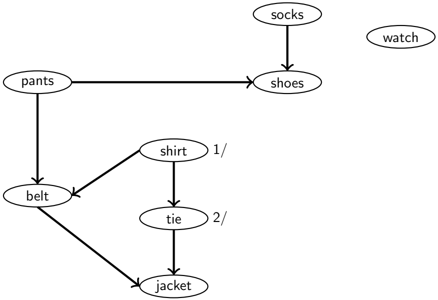
Application



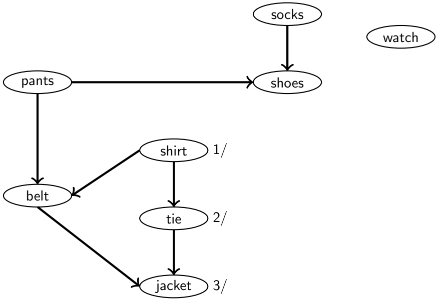
Application



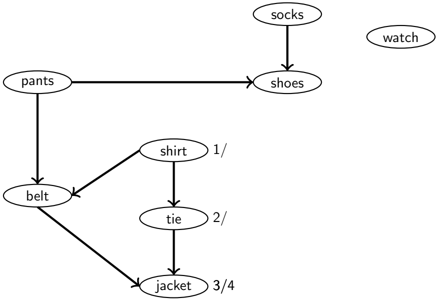
Application



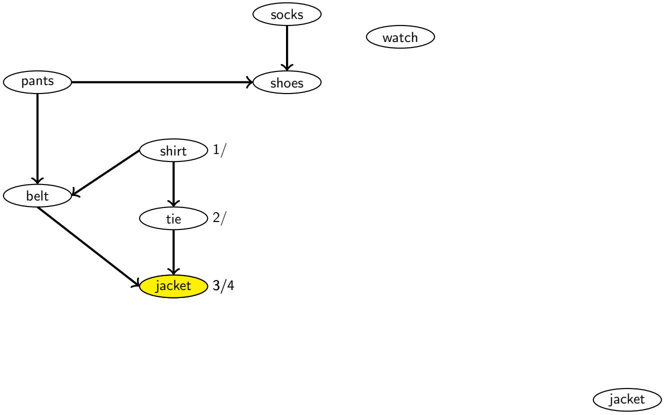
Application



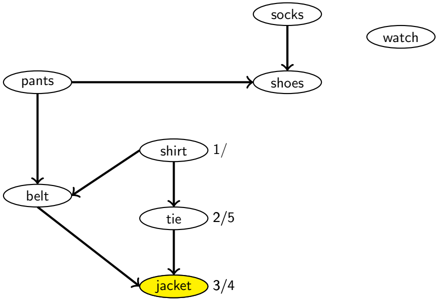
Application



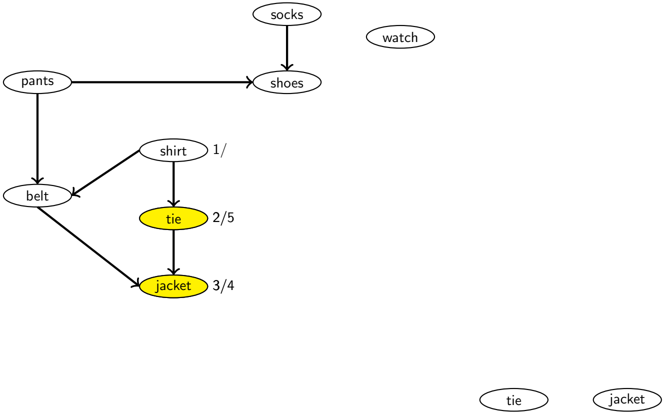
Application



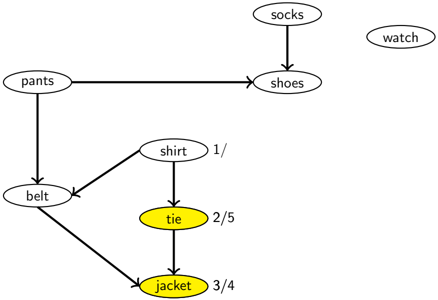
Application

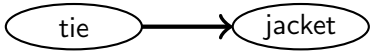


Application

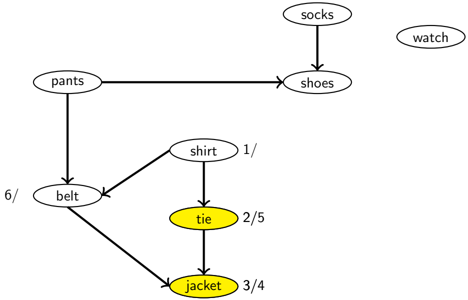


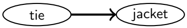
Application



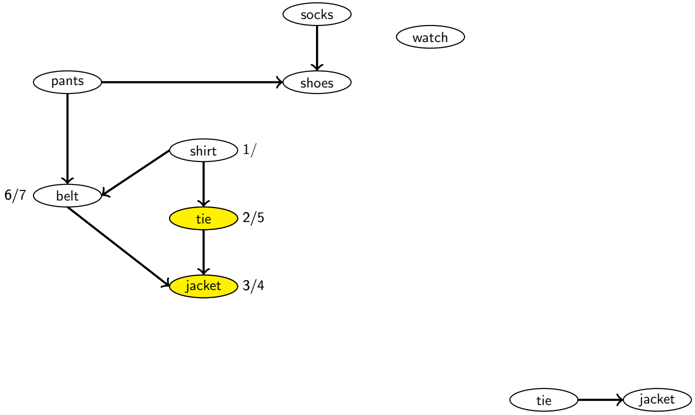


Application

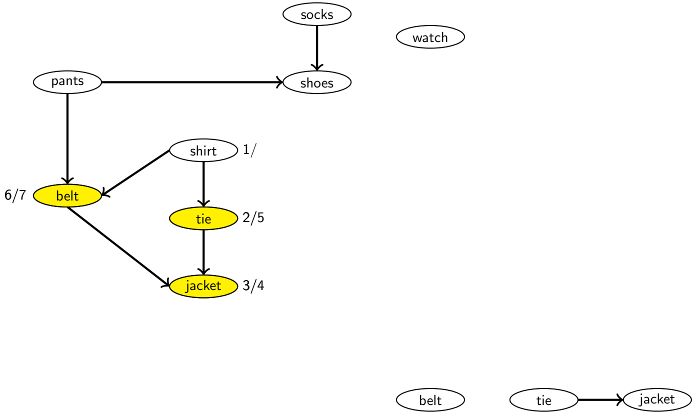




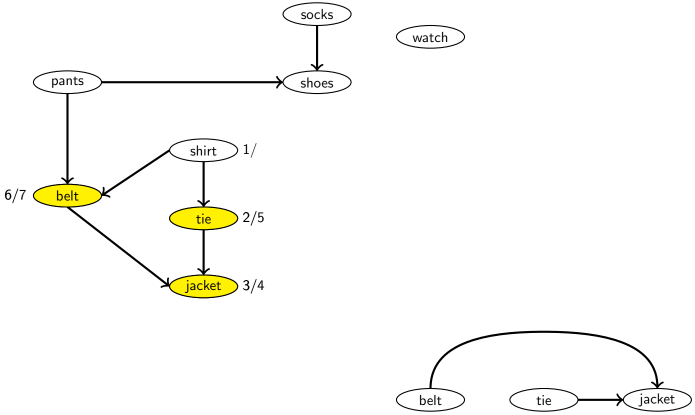
Application



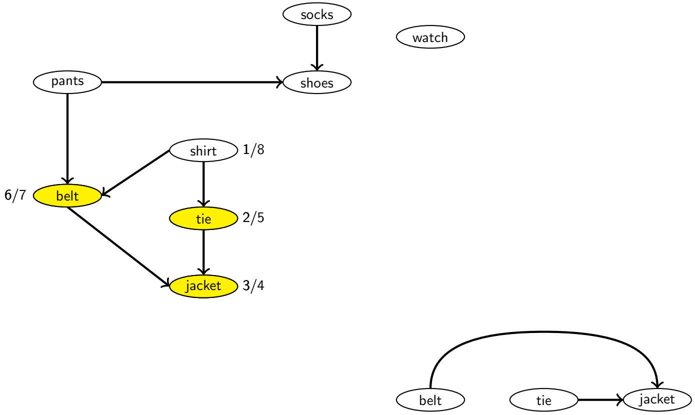
Application



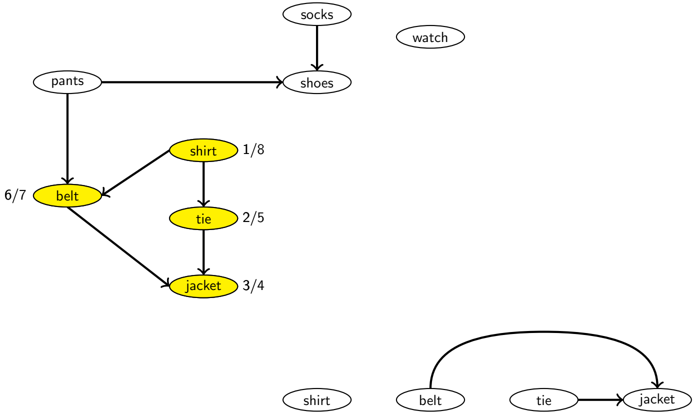
Application



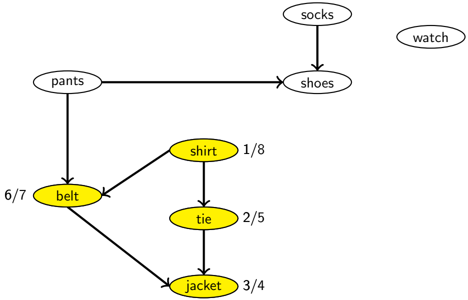
Application

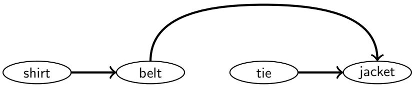


Application

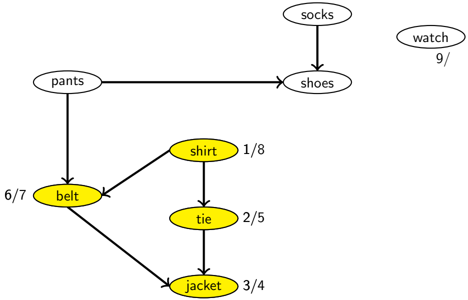


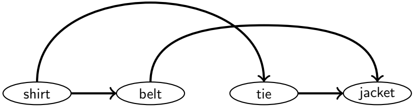
Application



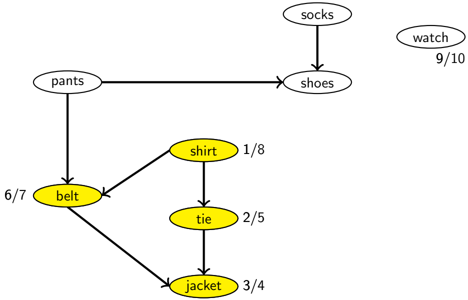


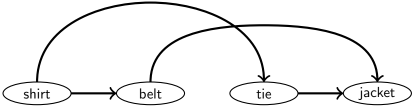
Application



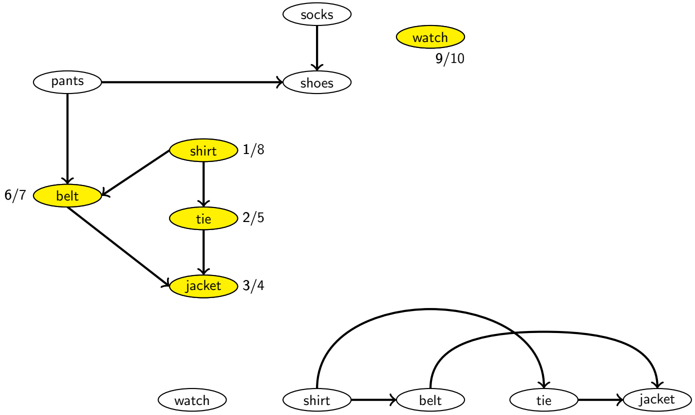


Application

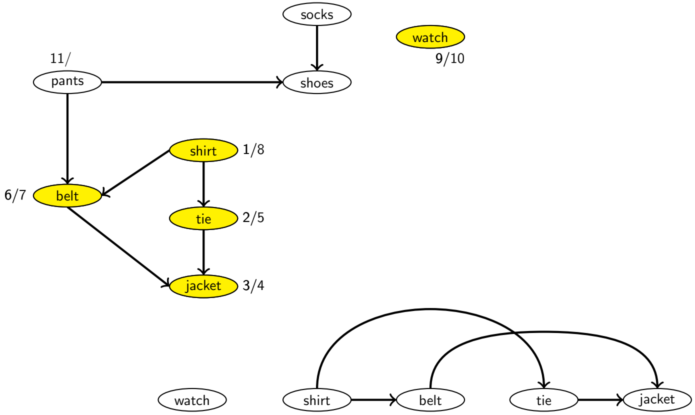




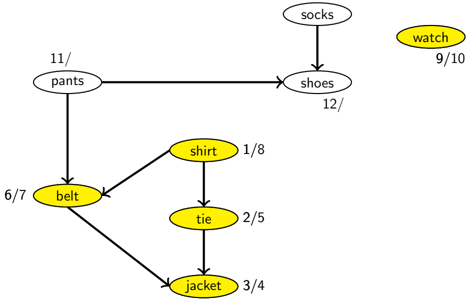
Application

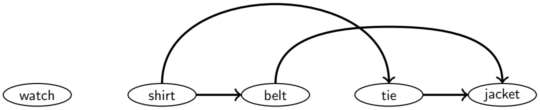


Application

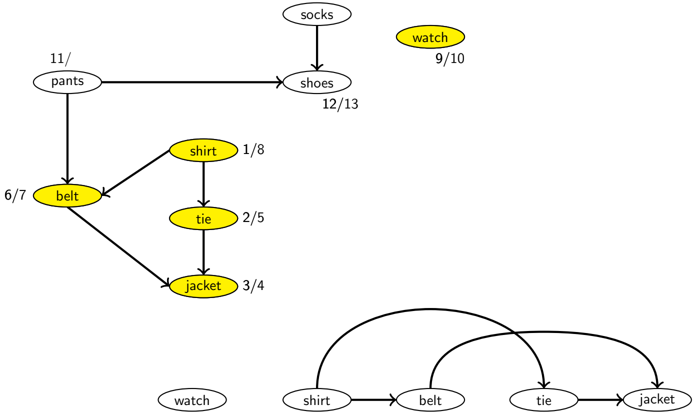


Application

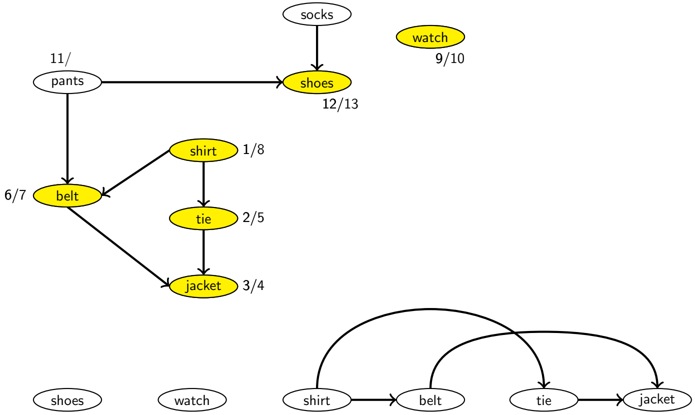




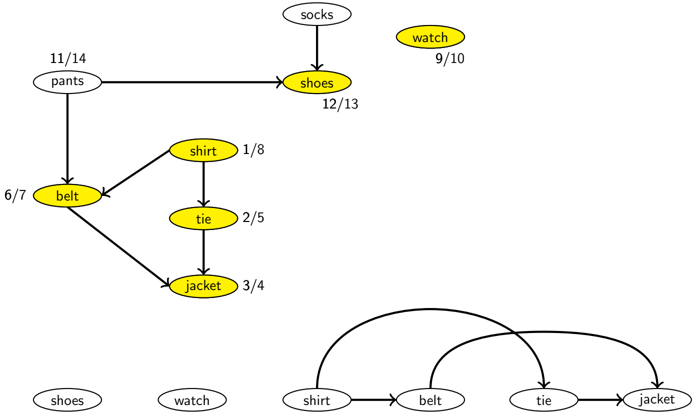
Application



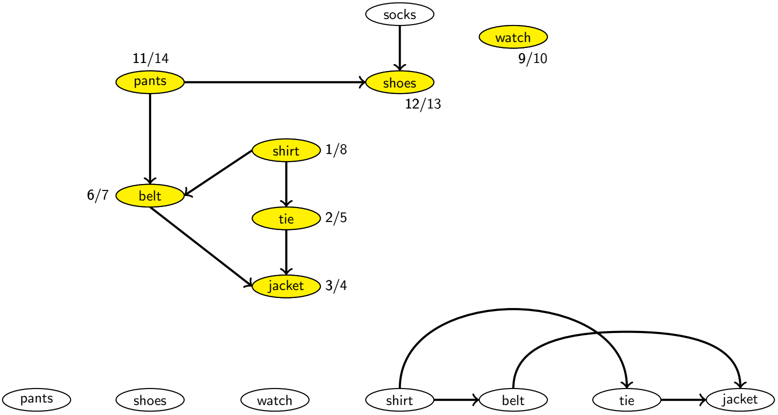
Application



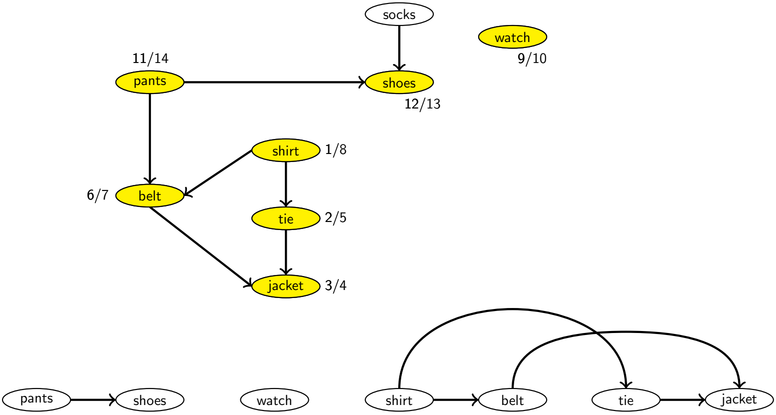
Application



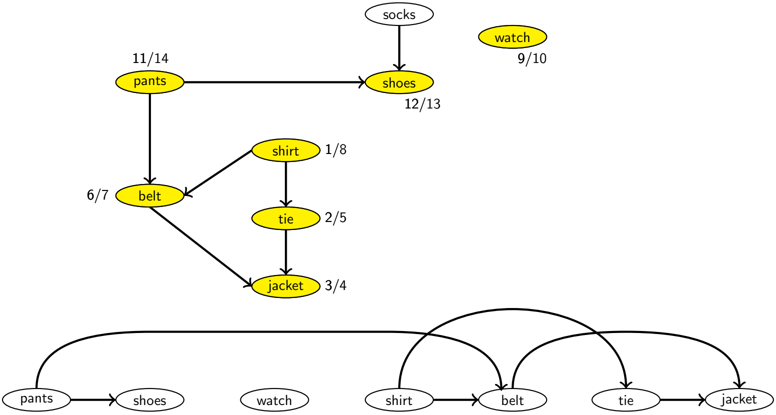
Application



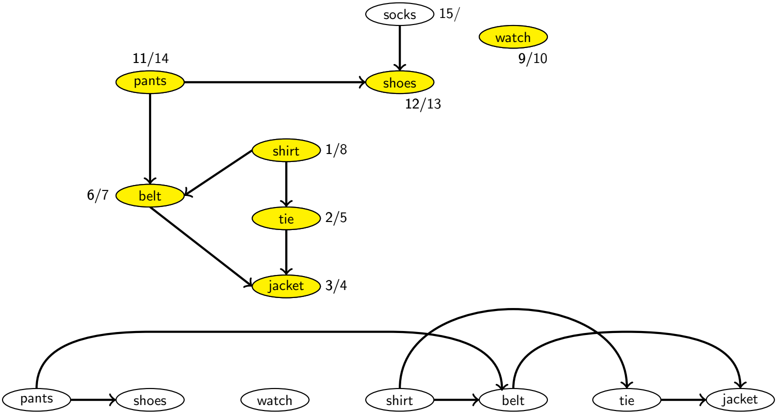
Application



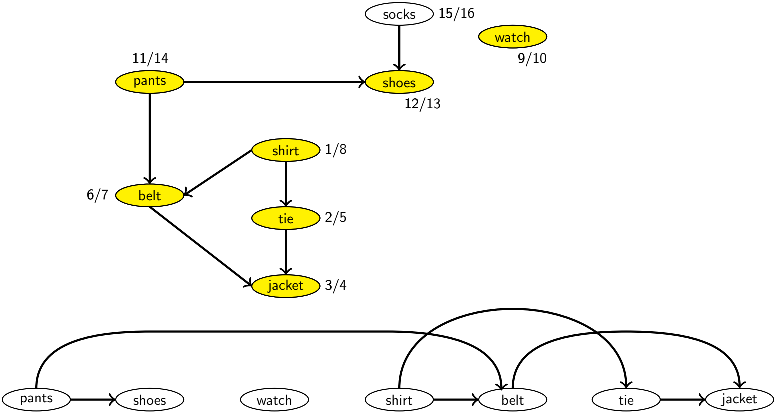
Application



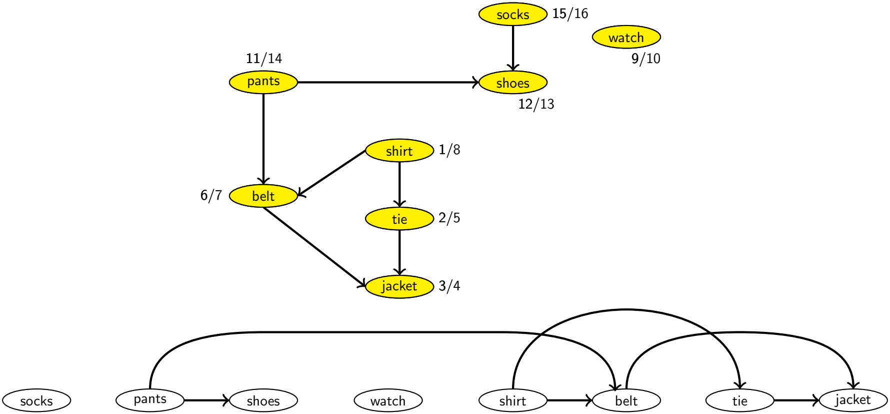
Application



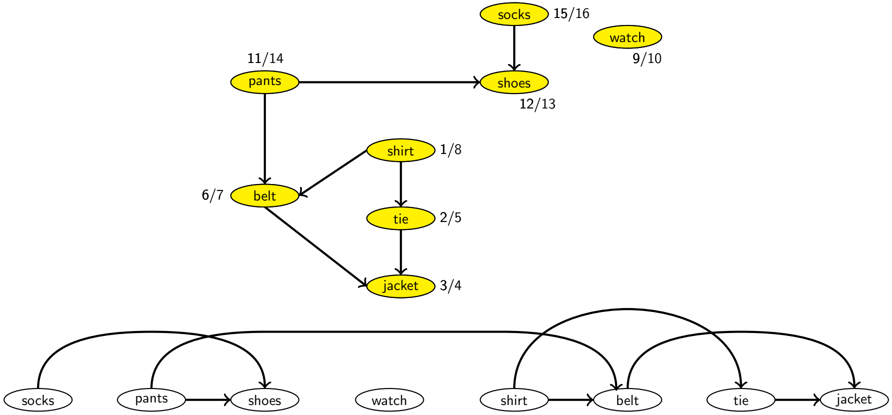
Application



Application



Application



Outline

Acknowledgement

- Introduction to Algorithms by Thomas H.Cormen,Charles E.Leiserson,Ronald L.Rivest and Clifford Stein.

Outline

Conclusion