Informatik II

Übung 10

FS 2019

Program Today

1. Repetition Lectures: Adjacency Lists
2. Breadth-First-Search BFS
3. In-Class-Exercise

Adjacency List

```
class Graph { // G = (V,E) as adjacency list
   private int V; // number of vertices
   private ArrayList<LinkedList<Integer>> adj; // adj. list
   // Constructor
   public Graph(int n) {
      V = n;
      adj = new ArrayList<LinkedList<Integer>>(V);
      for (int i=0; i<V; ++i)
         adj.add(i,new LinkedList<Integer>());
   }
   // Edge adder method
   public void addEdge(int u,int v) {
      adj.get(u).add(v);
   }
}
```

Adjacency List

Properties:

`ArrayList`
: Get element in constant time.

`LinkedList`
: Add element in constant time.
: Iterate over whole list in linear time.

Adjacency List

Properties:

`ArrayList`
: Get element in constant time.

`LinkedList`
: Add element in constant time.
Iterate over whole list in linear time.

- `addEdge(u,v) = adj.get(u).add(v)` runs in constant time $\mathcal{O}(1)$.

Adjacency List

Properties:

`ArrayList`
: Get element in constant time.

`LinkedList`
: Add element in constant time.
Iterate over whole list in linear time.

- `addEdge(u,v) = adj.get(u).add(v)` runs in constant time $\mathcal{O}(1)$.
- `for (int v : adj.get(u))` runs in time $\mathcal{O}(\deg^+(u))$.

Runtimes of simple Operations

Operation	Matrix	List
Find neighbours/successors of $v \in V$		
find $v \in V$ without neighbour/successor		
$(u, v) \in E$?		
Insert edge		
Delete edge		

Runtimes of simple Operations

Operation	Matrix	List
Find neighbours/successors of $v \in V$	$\Theta(n)$	
find $v \in V$ without neighbour/successor		
$(u, v) \in E$?		
Insert edge		
Delete edge		

Runtimes of simple Operations

Operation	Matrix	List
Find neighbours/successors of $v \in V$	$\Theta(n)$	$\Theta(\deg^+ v)$
find $v \in V$ without neighbour/successor		
$(u, v) \in E$?		
Insert edge		
Delete edge		

Runtimes of simple Operations

Operation	Matrix	List
Find neighbours/successors of $v \in V$	$\Theta(n)$	$\Theta(\deg^+ v)$
find $v \in V$ without neighbour/successor	$\Theta(n^2)$	
$(u, v) \in E$?		
Insert edge		
Delete edge		

Runtimes of simple Operations

Operation	Matrix	List
Find neighbours/successors of $v \in V$	$\Theta(n)$	$\Theta(\deg^+ v)$
find $v \in V$ without neighbour/successor	$\Theta(n^2)$	$\Theta(n)$
$(u, v) \in E$?		
Insert edge		
Delete edge		

Runtimes of simple Operations

Operation	Matrix	List
Find neighbours/successors of $v \in V$	$\Theta(n)$	$\Theta(\deg^+ v)$
find $v \in V$ without neighbour/successor	$\Theta(n^2)$	$\Theta(n)$
$(u, v) \in E$?	$\Theta(1)$	
Insert edge		
Delete edge		

Runtimes of simple Operations

Operation	Matrix	List
Find neighbours/successors of $v \in V$	$\Theta(n)$	$\Theta(\deg^+ v)$
find $v \in V$ without neighbour/successor	$\Theta(n^2)$	$\Theta(n)$
$(u, v) \in E$?	$\Theta(1)$	$\Theta(\deg^+ v)$
Insert edge		
Delete edge		

Runtimes of simple Operations

Operation	Matrix	List
Find neighbours/successors of $v \in V$	$\Theta(n)$	$\Theta(\deg^+ v)$
find $v \in V$ without neighbour/successor	$\Theta(n^2)$	$\Theta(n)$
$(u, v) \in E$?	$\Theta(1)$	$\Theta(\deg^+ v)$
Insert edge	$\Theta(1)$	
Delete edge		

Runtimes of simple Operations

Operation	Matrix	List
Find neighbours/successors of $v \in V$	$\Theta(n)$	$\Theta(\deg^+ v)$
find $v \in V$ without neighbour/successor	$\Theta(n^2)$	$\Theta(n)$
$(u, v) \in E$?	$\Theta(1)$	$\Theta(\deg^+ v)$
Insert edge	$\Theta(1)$	$\Theta(1)$
Delete edge		

Runtimes of simple Operations

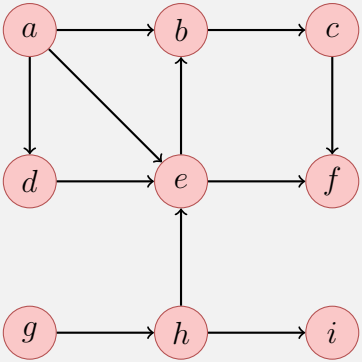
Operation	Matrix	List
Find neighbours/successors of $v \in V$	$\Theta(n)$	$\Theta(\deg^+ v)$
find $v \in V$ without neighbour/successor	$\Theta(n^2)$	$\Theta(n)$
$(u, v) \in E$?	$\Theta(1)$	$\Theta(\deg^+ v)$
Insert edge	$\Theta(1)$	$\Theta(1)$
Delete edge	$\Theta(1)$	

Runtimes of simple Operations

Operation	Matrix	List
Find neighbours/successors of $v \in V$	$\Theta(n)$	$\Theta(\deg^+ v)$
find $v \in V$ without neighbour/successor	$\Theta(n^2)$	$\Theta(n)$
$(u, v) \in E$?	$\Theta(1)$	$\Theta(\deg^+ v)$
Insert edge	$\Theta(1)$	$\Theta(1)$
Delete edge	$\Theta(1)$	$\Theta(\deg^+ v)$

Breadth-First-Search BFS

BFS starting from a:

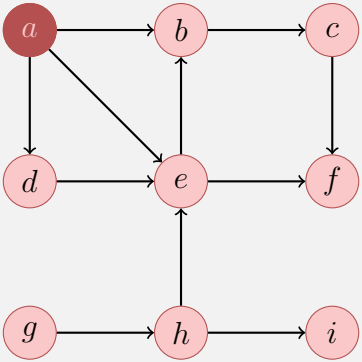


BFS-Tree: Distances and Parents

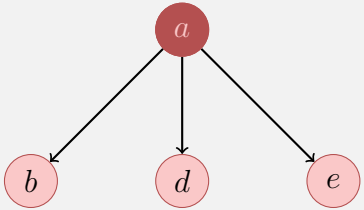
a distance 0

Breadth-First-Search BFS

BFS starting from a:



BFS-Tree: Distances and Parents

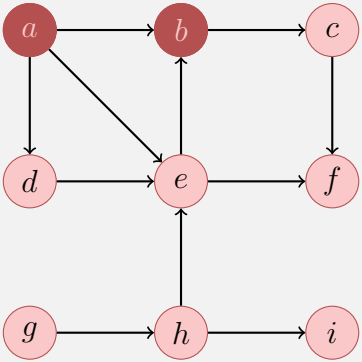


distance 0

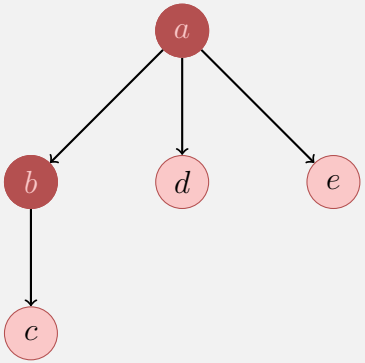
distance 1

Breadth-First-Search BFS

BFS starting from a:



BFS-Tree: Distances and Parents



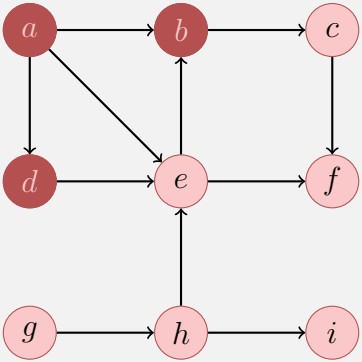
distance 0

distance 1

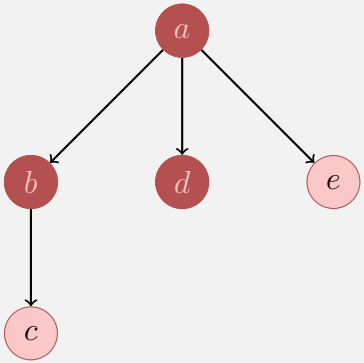
distance 2

Breadth-First-Search BFS

BFS starting from a:



BFS-Tree: Distances and Parents



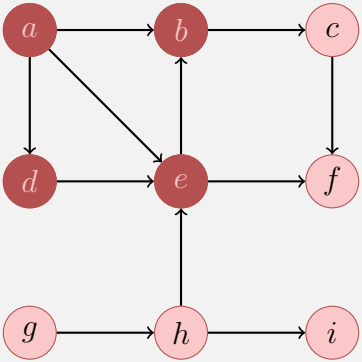
distance 0

distance 1

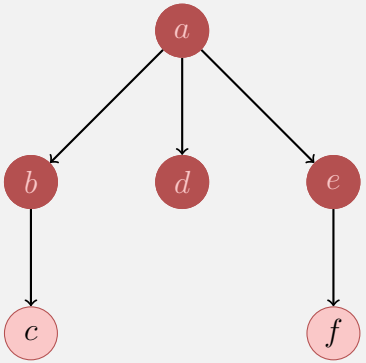
distance 2

Breadth-First-Search BFS

BFS starting from a:



BFS-Tree: Distances and Parents



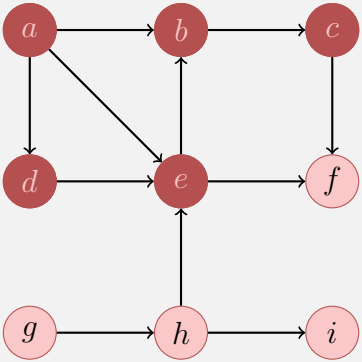
distance 0

distance 1

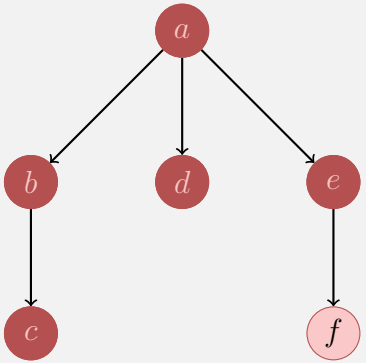
distance 2

Breadth-First-Search BFS

BFS starting from a:



BFS-Tree: Distances and Parents



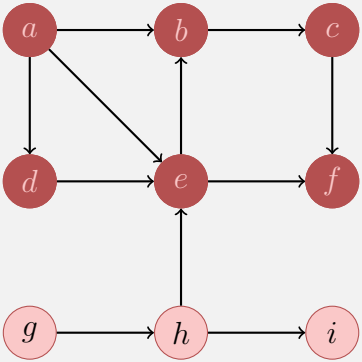
distance 0

distance 1

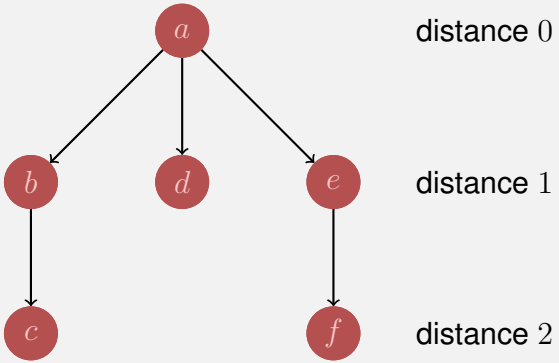
distance 2

Breadth-First-Search BFS

BFS starting from a:

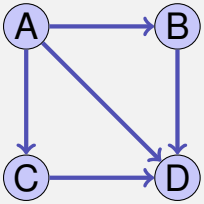


BFS-Tree: Distances and Parents



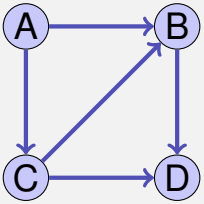
Quiz

In how many ways can the following directed graphs be topologically sorted each?



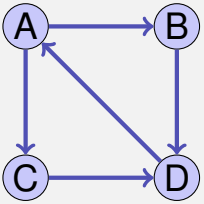
number sortings

?



number sortings

?

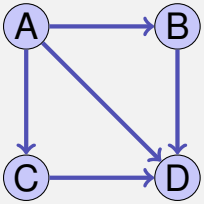


number sortings

?

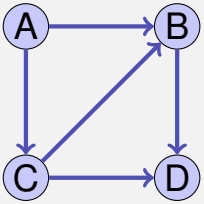
Quiz

In how many ways can the following directed graphs be topologically sorted each?



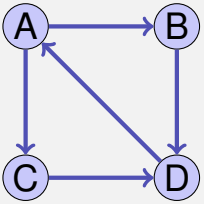
number sortings

2



number sortings

1



number sortings

0

In-Class-Exercises: Route planning

Exercise: You are given

- a directed, unweighted Graph $G = (V, E)$, represented by an adjacency list,
- and a designated node $t \in V$ (e.g., an emergency exit).

Design an algorithm,

- which computes for each node $u \in V$ an outgoing edge in direction of a shortest path to t.
- and has a running time of $\mathcal{O}(|V| + |E|)$.

In-Class-Exercises: Route planning

Solution:

1. Make a copy of the graph with edges having reverse direction: $G^T = (V, E^T)$, where $E^T = \{(v, u) \mid (u, v) \in E\}$.
   Running time: $\mathcal{O}(|V| + |E|)$.
2. Start a breadth-first search of G^T, starting from t,
   and store all edges of the BFS-Tree.
   Running time: $\mathcal{O}(|V| + |E^T|) = \mathcal{O}(|V| + |E|)$.
3. Assign the stored edges (in reverse direction)
   to the discovered nodes. Running time: $\mathcal{O}(|V|)$.

Questions / Suggestions?