Algorithmic Graph Theory

Winter 2014/15

1. Lecture

Graphs: An Introduction

Dr. Joachim Spoerhase and Prof. Dr. Alexander Wolff
Chair for Computer Science I
Daily Schedule

09:00–10:30 Lecture
10:30–12:00 Solving Exercises
12:00–12:30 Discussion of solutions
12:30–13:30 Lunch
13:30–15:00 Lecture
15:00–16:30 Solving Exercises
16:30–17:00 Discussion of solutions
Books

[KN]  [CLRS]  [G]
Prerequisites

- Graph traversal strategies
Prerequisites

Graph traversal strategies

- Breadth-first search
- Depth-first search
Prerequisites

- Graph traversal strategies
  - Breadth-first search (application: connected components)
  - Depth-first search
Prerequisites

Graph traversal strategies

- Breadth-first search (application: connected components)
- Depth-first search (topological sorting)
Prerequisites

- Graph traversal strategies
  - Breadth-first search  (application: connected components)
  - Depth-first search  (topological sorting)

- Computation of shortest paths
Prerequisites

- Graph traversal strategies
  - Breadth-first search (application: connected components)
  - Depth-first search (topological sorting)

- Computation of shortest paths
  - Breadth-first search
  - Dijkstra’s algorithm
Prerequisites

- **Graph traversal strategies**
  - Breadth-first search (application: connected components)
  - Depth-first search (topological sorting)

- **Computation of shortest paths**
  - Breadth-first search
  - Dijkstra’s algorithm

- **Minimum Spanning Trees**
Prerequisites

- **Graph traversal strategies**
  - Breadth-first search (application: connected components)
  - Depth-first search (topological sorting)

- **Computation of shortest paths**
  - Breadth-first search
  - Dijkstra’s algorithm

- **Minimum Spanning Trees**
  - Kruskal’s algorithm
  - Jarník–Prim algorithm
What is this?
What is this?

One (and the same) graph.
What is this?

One (and the same) *graph*.; the three-dimensional hypercube.
Q: What is a graph?
Q: What is a graph?

A₁: An (undirected) graph is a pair \((V, E)\)
Q: What is a graph?

A1: An (undirected) graph is a pair \((V, E)\)
Q: What is a graph?

A₁: An (undirected) graph is a pair \((V, E)\), where

- \(V\) node set and
- \(E \subseteq \binom{V}{2} = \{\{u, v\} \subseteq V \mid u \neq v\}\) edge set.
Q: What is a graph?

A₁: An (undirected) graph is a pair \((V, E)\), where
- \(V\) node set and
- \(E \subseteq \binom{V}{2} = \{\{u, v\} \subseteq V \mid u \neq v\}\) edge set.

\[ V = \{000, 001, \ldots, 111\} \]
\[ \{u, v\} \in E \quad \Leftrightarrow \quad ? \]
Q: What is a graph?

A₁: An (undirected) graph is a pair \((V, E)\), where
- \(V\) node set and
- \(E \subseteq \binom{V}{2} = \{\{u, v\} \subseteq V \mid u \neq v\}\) edge set.

\(V = \{000, 001, \ldots, 111\}\)
\(\{u, v\} \in E \iff \)
Q: What is a graph?

A1: An (undirected) graph is a pair \((V, E)\), where
- \(V\) node set and
- \(E \subseteq \binom{V}{2} = \{\{u, v\} \subseteq V \mid u \neq v\}\) edge set.

\(V = \{000, 001, \ldots, 111\}\)
\(\{u, v\} \in E \iff ?\)

A2: A directed graph is a pair \((V, E)\), where
- \(V\) node set and
- \(E \subseteq V \times V = \{(u, v) \in V^2 \mid u \neq v\}\) edge set.
Q: How can we represent graphs?

undirected graph
Q: How can we represent graphs?

undirected graph
Q: How can we represent graphs?

undirected graph

adjacency list
Q: How can we represent graphs?

We say: Node 3 and 5 are adjacent.

adjacency list
Q: How can we represent graphs?

We say: Node 3 and 5 are adjacent.

adjacency list
Q: How can we represent graphs?

We say: Node 3 and 5 are adjacent.

adjacency list

undirected graph
Q: How can we represent graphs?

We say: Node 3 and 5 are adjacent.

adjacency list

adjacency matrix
Q: How can we represent graphs?

We say: Node 3 and 5 are adjacent.

adjacency list

adjacency matrix
Q: How can we represent graphs?

We say: Node 3 and 5 are adjacent.

adjacency list

adjacency matrix
Q: How can we represent graphs?

We say: Node 3 and 5 are adjacent.

adjacency list

adjacency matrix
Q: How can we represent graphs?

We say: Node 3 and 5 are *adjacent*.

Undirected graph:

```
1 -- 2
  |    |
  |    |
  |    |
  |    |
  1 -- 2
```

Adjacency list:

```
1: 2
2: 3
3: 2
4: 3
5: 4
```

Adjacency matrix:

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```
Q: How can we represent graphs?

We say: Node 3 and 5 are adjacent.

adjacency list

adjacency matrix
Q: How can we represent graphs?

We say: Node 3 and 5 are adjacent.

adjacency list

adjacency matrix
Q: How can we represent graphs?

We say: Node 3 and 5 are adjacent.

**adjacency list**

**adjacency matrix**

undirected graph

directed graph
Q: How can we represent graphs?

We say: Node 3 and 5 are adjacent.

Adjacency list:

\[
\text{Adj}[i] = \{ j \in V \mid (i, j) \in E \}\]

Adjacency matrix:

\[
\begin{bmatrix}
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0
\end{bmatrix}
\]
Q: How can we represent graphs?

We say: Node 3 and 5 are adjacent.

Adjacency list:

\[
\begin{align*}
\text{Adj}[i] &= \{ j \in V \mid (i, j) \in E \} \\
a_{ij} &= 1 \iff (i, j) \in E
\end{align*}
\]
Degree of a node

Def.  

$u$
Degree of a node

**Def.**

\[ \text{deg } u = |\text{Adj}[u]| \]
Degree of a node

Def.

\[ \text{deg } u = |\text{Adj}[u]| \]
Degree of a node

**Def.**

\[ \text{deg } u = |\text{Adj}[u]| \]

\[ \text{outdeg } v = |\text{Adj}[v]| \]
Degree of a node

Def.

\[ \text{deg } u = |\text{Adj}[u]| \]

\[ \text{outdeg } v = |\text{Adj}[v]| \]

\[ \text{indeg } v = |\{ u \in V : (u, v) \in E \}| \]
Degree of a node

**Def.**

\[ \text{deg } u = |\text{Adj}[u]| \]

\[ \text{outdeg } v = |\text{Adj}[v]| \]

\[ \text{indeg } v = |\{u \in V : (u, v) \in E\}| \]

**Obs.** Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is
Degree of a node

Def. \[ \text{deg } u = |\text{Adj}[u]| \]
\[ \text{outdeg } v = |\text{Adj}[v]| \]
\[ \text{indeg } v = |\{u \in V : (u, v) \in E\}| \]

Obs. Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( 2 \cdot |E| \).
**Degree of a node**

**Def.**

\[
\text{deg } u = |\text{Adj}[u]|
\]

\[
\text{outdeg } v = |\text{Adj}[v]|
\]

\[
\text{indeg } v = |\{ u \in V : (u, v) \in E \}|
\]

**Obs.** Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( = 2 \cdot |E| \).

**Proof.** Technique of *double counting:*
Degree of a node

Def. \quad \deg u = |\text{Adj}[u]|

\quad \text{outdeg} \ v = |\text{Adj}[v]|

\quad \text{indeg} \ v = |\{u \in V : (u, v) \in E\}|

Obs. \quad \text{Let } G = (V, E) \text{ be an undirected graph.}
\text{Then, the sum of all node degrees is } = 2 \cdot |E|.

Proof. \quad \text{Technique of double counting:}
\text{count all node-edge incidences}
Degree of a node

**Def.**

\[ \text{deg } u = \lvert \text{Adj}[u] \rvert \]
\[ \text{outdeg } v = \lvert \text{Adj}[v] \rvert \]
\[ \text{indeg } v = \lvert \{u \in V : (u, v) \in E\} \rvert \]

**Obs.** Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( = 2 \cdot \lvert E \rvert \).

**Proof.** Technique of *double counting*:

count all node-edge incidences
Degree of a node

**Def.**

\[
\text{deg } u = |\text{Adj}[u]|
\]

\[
\text{outdeg } v = |\text{Adj}[v]|
\]

\[
\text{indeg } v = |\{u \in V : (u, v) \in E\}|
\]

**Obs.** Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is equal to \( 2 \cdot |E| \).

**Proof.** Technique of *double counting*: count all node-edge incidences.

An edge is *incident* on its endpoints.
Degree of a node

**Def.**

- \( \text{deg } u = |\text{Adj}[u]| \)
- \( \text{outdeg } v = |\text{Adj}[v]| \)
- \( \text{indeg } v = |\{ u \in V : (u, v) \in E \}| \)

**Obs.** Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( = 2 \cdot |E| \).

**Proof.** Technique of *double counting*:
- Count all node-edge incidences

An edge is *incident* on its endpoints.

A node is *incident* on all edges, whose endpoint it is.
Degree of a node

**Def.**

\[
\text{deg } u = |\text{Adj}[u]| \\
\text{outdeg } v = |\text{Adj}[v]| \\
\text{indeg } v = |\{u \in V : (u, v) \in E\}|
\]

**Obs.** Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( 2 \cdot |E| \).

**Proof.** Technique of *double counting*:

- count all node-edge incidences
  - node-wise:
  - edge-wise:
Degree of a node

Def. 
\[ \text{deg } u = |\text{Adj}[u]| \]
\[ \text{outdeg } v = |\text{Adj}[v]| \]
\[ \text{indeg } v = |\{ u \in V : (u, v) \in E \}| \]

Obs. Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( = 2 \cdot |E| \).

Proof. Technique of double counting: count all node-edge incidences

node-wise: \[ \sum_{v \in V} \text{deg } v \]
edge-wise:
Degree of a node

**Def.**

\[
\text{deg } u = |\text{Adj}[u]|
\]

\[
\text{outdeg } v = |\text{Adj}[v]|
\]

\[
\text{indeg } v = |\{u \in V : (u, v) \in E\}|
\]

**Obs.** Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( = 2 \cdot |E| \).

**Proof.** Technique of *double counting*:

- **count all node-edge incidences**
  - **node-wise:** \( \sum_{v \in V} \text{deg } v \)
  - **edge-wise:** \( 2 \cdot |E| \)
Degree of a node

**Def.**
- \( \text{deg } u = |\text{Adj}[u]| \)
- \( \text{outdeg } v = |\text{Adj}[v]| \)
- \( \text{indeg } v = |\{u \in V : (u, v) \in E\}| \)

**Obs.** Let \( G = (V, E) \) be an undirected graph.
Then, the sum of all node degrees is \( \sum_{v \in V} \text{deg } v = 2 \cdot |E| \).

**Proof.** Technique of *double counting*:
count all node-edge incidences
- node-wise: \( \sum_{v \in V} \text{deg } v \)
- edge-wise: \( 2 \cdot |E| \) thus equal!
Degree of a node

**Def.**

\[
\text{deg} \ u = |\text{Adj}[u]| \\
\text{outdeg} \ v = |\text{Adj}[v]| \\
\text{indeg} \ v = |\{u \in V : (u, v) \in E\}|
\]

**Obs.**

Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( = 2 \cdot |E| \).
Degree of a node

**Def.**
\[ \text{deg } u = |\text{Adj}[u]| \]
\[ \text{outdeg } v = |\text{Adj}[v]| \]
\[ \text{indeg } v = |\{ u \in V : (u, v) \in E \}| \]

**Obs.** Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( = 2 \cdot |E| \).

**Thm.** The number of nodes of odd degree is even.
Degree of a node

Def. \[ \text{deg } u = |\text{Adj}[u]| \]
\[ \text{outdeg } \nu = |\text{Adj}[\nu]| \]
\[ \text{indeg } \nu = |\{ u \in V : (u, \nu) \in E \}| \]

Obs. Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( = 2 \cdot |E| \).

Thm. The number of nodes of odd degree is even.

Proof. \[ 2 \cdot |E| = \sum_{\nu \in V} \text{deg } \nu \]
Degree of a node

**Def.**

\[ \text{deg } u = |\text{Adj}[u]| \]

\[ \text{outdeg } v = |\text{Adj}[v]| \]

\[ \text{indeg } v = |\{ u \in V : (u, v) \in E \}| \]

**Obs.** Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( 2 \cdot |E| \).

**Thm.** The number of nodes of odd degree is even.

**Proof.**

\[ 2 \cdot |E| = \sum_{v \in V} \text{deg } v = \sum_{v \in V_{\text{even}}} \text{deg } v + \sum_{v \in V_{\text{odd}}} \text{deg } v \]
Degree of a node

Def.  
\[ \text{deg } u = |\text{Adj}[u]| \]
\[ \text{outdeg } v = |\text{Adj}[v]| \]
\[ \text{indeg } v = |\{ u \in V : (u, v) \in E \}| \]

Obs.  Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( = 2 \cdot |E| \).

Thm.  The number of nodes of odd degree is even.

Proof.  \[ 2 \cdot |E| = \sum_{v \in V} \text{deg } v = \sum_{v \in V_{\text{even}}} \text{deg } v + \sum_{v \in V_{\text{odd}}} \text{deg } v \]

\text{even!}
Degree of a node

**Def.**

\[ \text{deg } u = |\text{Adj}[u]| \]
\[ \text{outdeg } v = |\text{Adj}[v]| \]
\[ \text{indeg } v = |\{ u \in V : (u, v) \in E \}| \]

**Obs.**

Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( 2 \cdot |E| \).

**Thm.**

The number of nodes of odd degree is even.

**Proof.**

\[ 2 \cdot |E| = \sum_{v \in V} \text{deg } v = \sum_{v \in V_{\text{even}}} \text{deg } v + \sum_{v \in V_{\text{odd}}} \text{deg } v \]
Degree of a node

**Def.**

\[
\text{deg } u = |\text{Adj}[u]| \\
\text{outdeg } v = |\text{Adj}[v]| \\
\text{indeg } v = |\{u \in V : (u, v) \in E\}|
\]

**Obs.**

Let \( G = (V, E) \) be an undirected graph.
Then, the sum of all node degrees is \( 2 \cdot |E| \).

**Thm.**

The number of nodes of odd degree is even.

**Proof.**

\[
2 \cdot |E| = \sum_{v \in V} \text{deg } v = \sum_{v \in V_{\text{even}}} \text{deg } v + \sum_{v \in V_{\text{odd}}} \text{deg } v
\]

even! even! even!
Degree of a node

**Def.**

$$\text{deg } u = |\text{Adj}[u]|$$

$$\text{outdeg } v = |\text{Adj}[v]|$$

$$\text{indeg } v = |\{u \in V : (u, v) \in E\}|$$

**Obs.**

Let $G = (V, E)$ be an undirected graph. Then, the sum of all node degrees is $= 2 \cdot |E|$.

**Thm.**

The number of nodes of odd degree is even.

**Proof.**

$$2 \cdot |E| = \sum_{v \in V} \text{deg } v = \sum_{v \in V_{\text{even}}} \text{deg } v + \sum_{v \in V_{\text{odd}}} \text{deg } v$$

$\text{even!} \quad \text{even!} \quad \text{even!}$
Degree of a node

Def. $\text{deg } u = |\text{Adj}[u]|$

$\text{outdeg } v = |\text{Adj}[v]|$

$\text{indeg } v = |\{ u \in V : (u, v) \in E \}|$

Obs. Let $G = (V, E)$ be an undirected graph. Then, the sum of all node degrees is $= 2 \cdot |E|$.

Thm. The number of nodes of odd degree is even.

Proof. $2 \cdot |E| = \sum_{v \in V} \text{deg } v = \sum_{v \in V_{\text{even}}} \text{deg } v + \sum_{v \in V_{\text{odd}}} \text{deg } v$

$\text{even!} \quad \text{even!} \quad \text{even!} \quad \Rightarrow \text{ even!}$
Degree of a node

**Def.**
- \( \text{deg } u = |\text{Adj}[u]| \)
- \( \text{outdeg } v = |\text{Adj}[v]| \)
- \( \text{indeg } v = |\{ u \in V : (u, v) \in E \}| \)

**Obs.** Let \( G = (V, E) \) be an undirected graph. Then, the sum of all node degrees is \( 2 \cdot |E| \).

**Thm.** The number of nodes of odd degree is even.

**Proof.**

\[
2 \cdot |E| = \sum_{v \in V} \text{deg } v = \sum_{v \in V_{\text{even}}} \text{deg } v + \sum_{v \in V_{\text{odd}}} \text{deg } v
\]

The sum of all node degrees is even, as the sum of degrees of even nodes is even and the sum of degrees of odd nodes is even, and the two are added together.

\[
\sum_{v \in V_{\text{odd}}} \text{deg } v \quad \Rightarrow \quad \text{even}
\]
**Degree of a node**

**Def.**

\[
\deg u = |\text{Adj}[u]|
\]

\[
\text{outdeg } v = |\text{Adj}[v]|
\]

\[
\text{indeg } v = |\{u \in V : (u, v) \in E\}|
\]

**Obs.** Let \( G = (V, E) \) be an undirected graph.
Then, the sum of all node degrees is \( = 2 \cdot |E| \).

**Thm.** The number of nodes of odd degree is even.

**Proof.**

\[
2 \cdot |E| = \sum_{v \in V} \deg v = \sum_{v \in V_{even}} \deg v + \sum_{v \in V_{odd}} \deg v
\]

\[
\text{even! even! even} \Rightarrow \text{even!}
\]

\[
\sum_{v \in V_{odd}} \deg v \text{ even } \Rightarrow |V_{odd}| \text{ ist even!}
\]

\[\square\]
Round trip strategies for undirected graphs

1. Traverse a graph on a cycle, so that every edge is traversed exactly once.
Round trip strategies for undirected graphs

1. Traverse a graph on a cycle, so that every edge is traversed exactly once.

*Characterization:* Which graphs admit such tours?
Round trip strategies for undirected graphs

1. Traverse a graph on a cycle, so that every edge is traversed exactly once.

*Characterization:* Which graphs admit such tours?

*Construction:* How (and in which time) can we compute such a cycle, if there is any?
Round trip strategies for undirected graphs

1. Traverse a graph on a cycle, so that every edge is traversed exactly once.

   **Characterization:** Which graphs admit such tours?
   **Construction:** How (and in which time) can we compute such a cycle, if there is any?

2. Traverse a graph on a cycle, so that every *node* is visited exactly once.
Round trip strategies for undirected graphs

1. Traverse a graph on a cycle, so that every edge is traversed exactly once.
   
   **Characterization:** Which graphs admit such tours?

   **Construction:** How (and in which time) can we compute such a cycle, if there is any?

2. Traverse a graph on a cycle, so that every node is visited exactly once.

   **Characterization:** Which graphs admit such cycles?

   **Construction:** How (and in which time) can we compute such a cycle, if there is any?
Round trip strategies for undirected graphs

1. Traverse a graph on a cycle, so that every edge is traversed exactly once.

   **Characterization:** Which graphs admit such tours?
   **Construction:** How (and in which time) can we compute such a cycle, if there is any?

2. Traverse a graph on a cycle, so that every node is visited exactly once.

   **Characterization:** Which graphs admit such cycles?
   **Construction:** How (and in which time) can we compute such a cycle, if there is any?
Round trip strategies for undirected graphs

1. Traverse a graph on a cycle, so that every edge is traversed exactly once.

Characterization: Which graphs admit such tours?
Construction: How (and in which time) can we compute such a cycle, if there is any?

2. Traverse a graph on a cycle, so that every node is visited exactly once.

Characterization: Which graphs admit such cycles?
Construction: How (and in which time) can we compute such a cycle, if there is any?