Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CHAI
Humanities-Centered AI

Transformer-based Algorithmics

Malte Luttermann – Institute for Humanities-Centered Artificial Intelligence (CHAI)

May 27, 2026

Version from May 27, 2026

Topics for Today

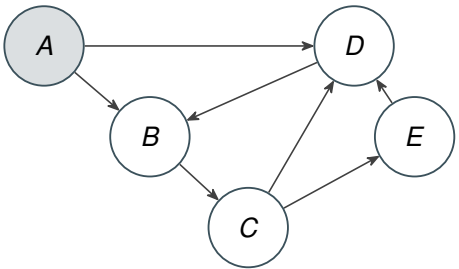
- Breadth-First Search (BFS)
- Depth-First Search (DFS)

Acknowledgement

- Some of the upcoming content is taken from the following lectures and corresponding exercises and has been translated and partially modified:
  - Prof. Dr. Ralf Möller: »Algorithmen und Datenstrukturen – Vorlesung«
  - Dr. Magnus Bender: »Algorithmen und Datenstrukturen – Übung«

Exercise 1 – Graph Algorithms

Consider the directed graph $G = (V, E)$ below.



(a) Perform a **breadth-first search** on G starting from A. List the tree edges and the distance of each vertex. *Tie-breaking:* enqueue successors in lexicographic order ($A < B < \ldots < E$).

(b) Perform a **depth-first search** on G starting from A. State *dfsNum* and *finishTime* for each vertex and classify every edge as *tree*, *forward*, *back*, or *cross* edge. *Tie-breaking:* visit the lexicographically smallest successor first.

Breadth-First Search – Reminder

BFS in a Nutshell

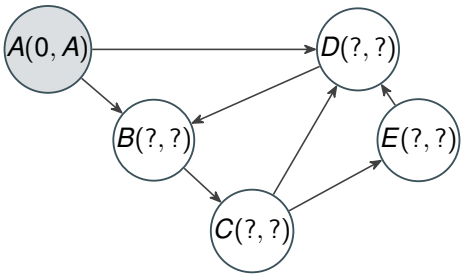
Maintains a **FIFO queue** of *discovered* vertices.

(1) Mark the source s with $s.d = 0$, set $s.parent = s$, and enqueue s.
(2) While the queue is non-empty: dequeue u and, for every outgoing neighbor v of u that is still unvisited, enqueue v and set $v.d = u.d + 1$, $v.parent = u$.

- Edges that lead to the discovery of a new vertex form the **BFS tree**; we will draw them in **red**.
- BFS computes **shortest-path distances** (in number of edges) from s to every reachable vertex.
- Runtime: $O(|V| + |E|)$.

Exercise 1 (a) – BFS from A

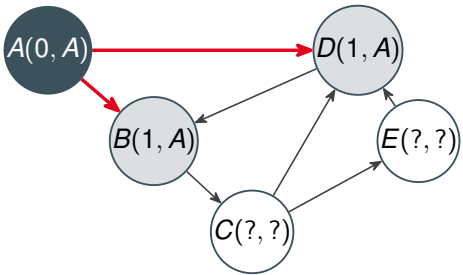
Vertex label: $v(v.d,\ v.parent)$. **Red** = tree edge. Gray = in queue, dark = processed.



Queue: $[A]$ Enqueue source A with $d = 0$.

Exercise 1 (a) – BFS from A

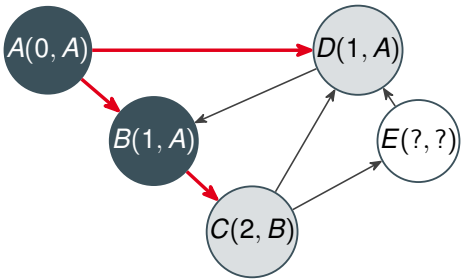
Vertex label: $v(v.d,\ v.parent)$. **Red** = tree edge. Gray = in queue, dark = processed.



Queue: $[B,\ D]$ Process A: discover B and D (lex. order).

Exercise 1 (a) – BFS from A

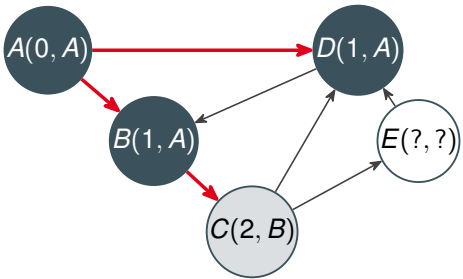
Vertex label: $v(v.d,\ v.parent)$. **Red** = tree edge. Gray = in queue, dark = processed.



Queue: $[D,\ C]$ Process B: discover C.

Exercise 1 (a) – BFS from A

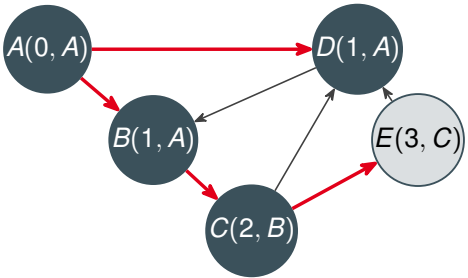
Vertex label: $v(v.d,\ v.parent)$. **Red** = tree edge. Gray = in queue, dark = processed.



Queue: $[C]$ Process D: $D \rightarrow B$, but B already discovered.

Exercise 1 (a) – BFS from A

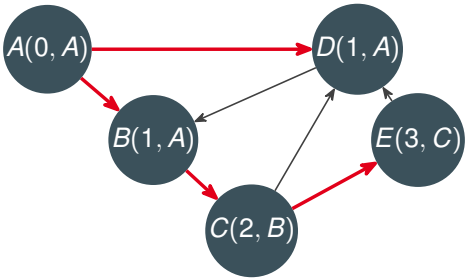
Vertex label: $v(v.d,\ v.parent)$. **Red** = tree edge. Gray = in queue, dark = processed.



Queue: $[E]$ Process C: discover E (skip D, already visited).

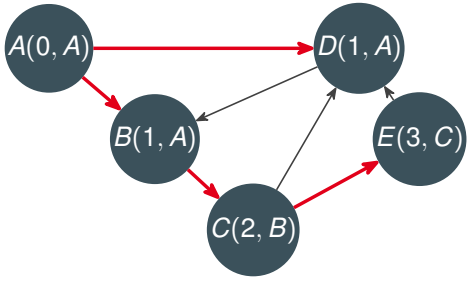
Exercise 1 (a) – BFS from *A*

Vertex label: $v(v.d,\ v.parent)$. **Red** = tree edge. Gray = in queue, dark = processed.



Queue: [ ] Process *E*: *D* already visited. **BFS done.**

Exercise 1 (a) – BFS Result



Tree edges: $A \to B$, $A \to D$, $B \to C$, $C \to E$.

BFS distances from A

v	A	B	C	D	E
$v.d$	0	1	2	1	3

Observation

Although the edge $C \to D$ exists, D is reached *directly* from A via $A \to D$ – BFS always finds a **shortest path** (in number of edges).

Depth-First Search – Reminder

DFS in a Nutshell

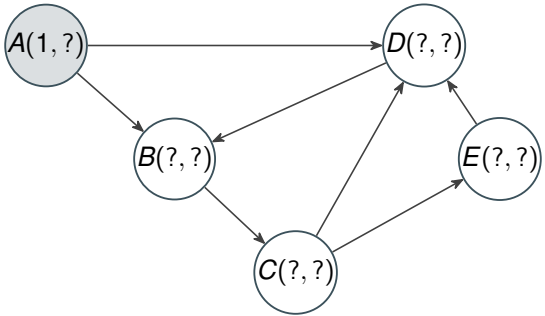
Recursive traversal with two counters: *dfsNum* (discovery order) and *finishTime* (completion order). For each outgoing neighbor v of u: recurse if v is unvisited; otherwise classify the edge $u \to v$ (see below).

Edge classification for $u \to v$:

- **Tree**: v was unvisited when u explored it.
- **Back**: v is still on the DFS stack (ancestor of u).
- **Forward**: v already finished, descendant of u ($u.dfsNum < v.dfsNum$).
- **Cross**: v already finished, $v.dfsNum < u.dfsNum$ (different subtree).

Exercise 1 (b) – DFS from A

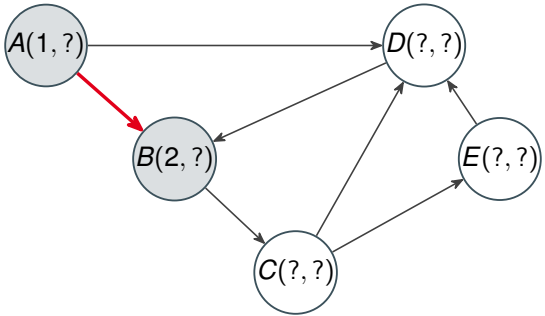
Vertex label: $v(\mathit{dfsNum},\ \mathit{finishTime})$. tree back forward cross



Visit A ($\mathit{dfsNum} = 1$). Stack: $[A]$.

Exercise 1 (b) – DFS from A

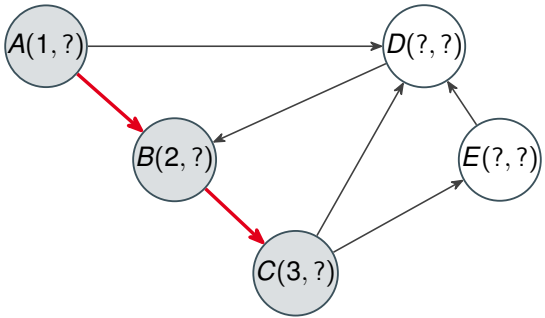
Vertex label: $v(\mathit{dfsNum},\ \mathit{finishTime})$. tree back forward cross



Visit B via tree edge $A \to B$ ($\mathit{dfsNum} = 2$). Stack: $[A, B]$.

Exercise 1 (b) – DFS from *A*

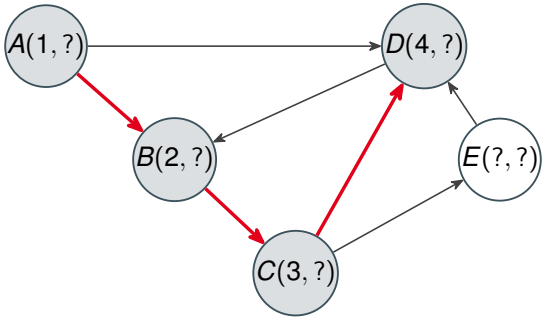
Vertex label: $v(dfsNum,\ finishTime)$. tree back forward cross



Visit C via tree edge $B \to C$ $(dfsNum = 3)$. Stack: $[A, B, C]$.

Exercise 1 (b) – DFS from A

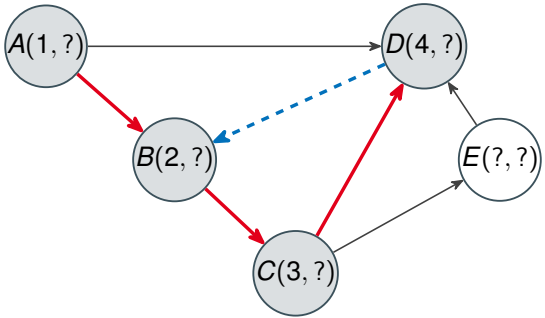
Vertex label: $v(\mathit{dfsNum},\ \mathit{finishTime})$. tree back forward cross



Visit D via tree edge $C \to D$ ($\mathit{dfsNum} = 4$). Stack: $[A, B, C, D]$.

Exercise 1 (b) – DFS from *A*

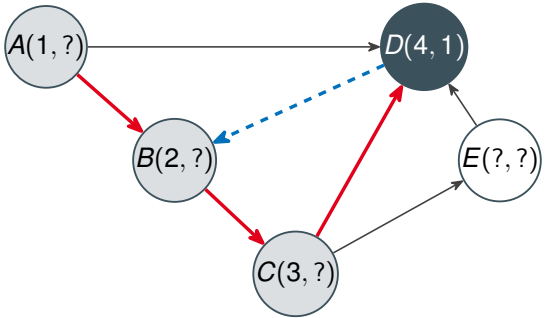
Vertex label: $v(dfsNum,\ finishTime)$. tree back forward cross



Edge $D \to B$: B is on the stack $\Rightarrow$ **back edge**.

Exercise 1 (b) – DFS from A

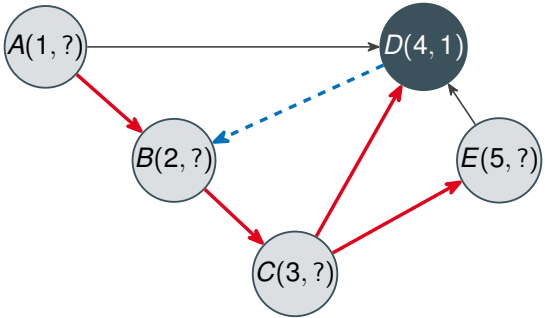
Vertex label: $v(\mathit{dfsNum},\ \mathit{finishTime})$. tree back forward cross



Finish D $(\mathit{finishTime} = 1)$. Stack: $[A, B, C]$.

Exercise 1 (b) – DFS from *A*

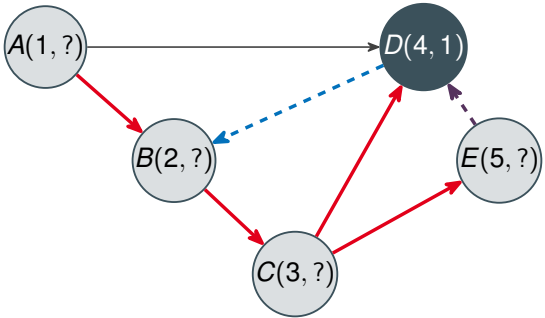
Vertex label: $v(dfsNum,\ finishTime)$. tree back forward cross



Back at C: **visit** E via tree edge $C \to E$ $(dfsNum = 5)$. Stack: $[A, B, C, E]$.

Exercise 1 (b) – DFS from *A*

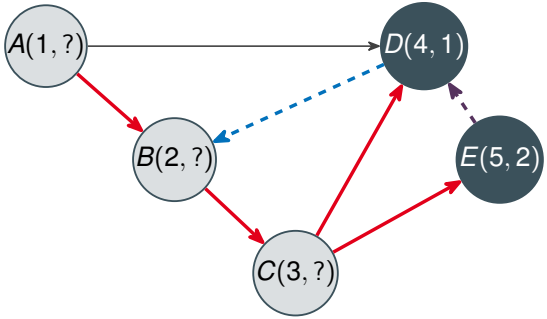
Vertex label: $v(dfsNum,\ finishTime)$. tree back forward cross



Edge $E \to D$: D finished and $D.dfsNum < E.dfsNum \Rightarrow$ **cross edge**.

Exercise 1 (b) – DFS from *A*

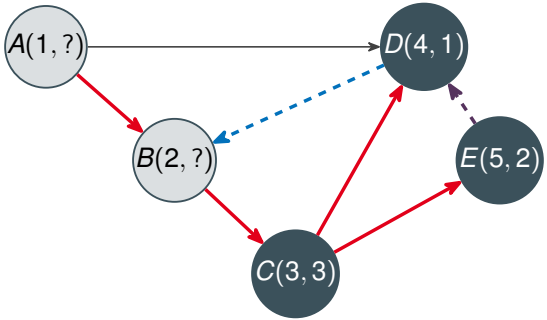
Vertex label: $v(dfsNum,\ finishTime)$. tree back forward cross



Finish E $(finishTime = 2)$.

Exercise 1 (b) – DFS from *A*

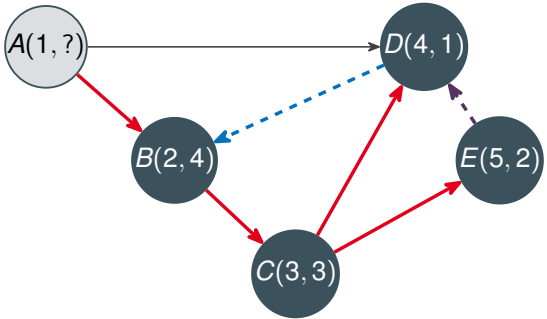
Vertex label: $v(\mathit{dfsNum}, \mathit{finishTime})$. tree back forward cross



Finish C $(\mathit{finishTime} = 3)$.

Exercise 1 (b) – DFS from *A*

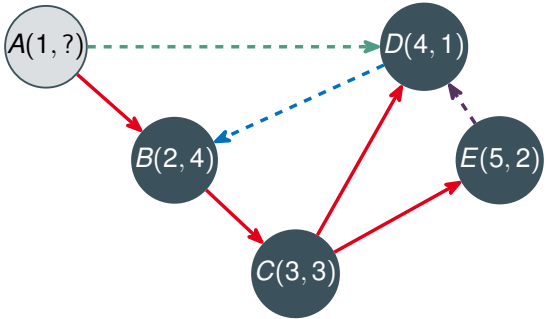
Vertex label: $v(dfsNum,\ finishTime)$. tree back forward cross



Finish B $(finishTime = 4)$.

Exercise 1 (b) – DFS from *A*

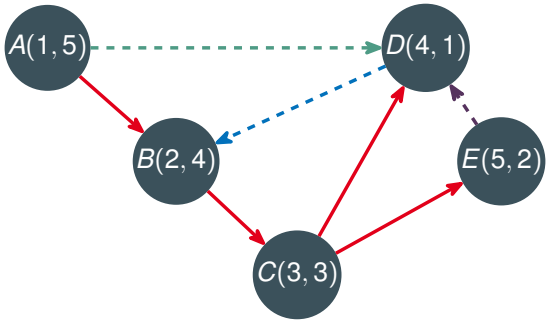
Vertex label: $v(dfsNum,\ finishTime)$. tree back forward cross



Back at *A*: edge $A \to D$, *D* finished and $A.dfsNum < D.dfsNum \Rightarrow$ **forward edge**.

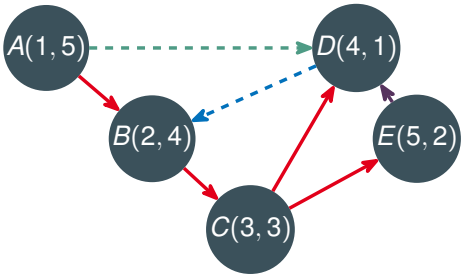
Exercise 1 (b) – DFS from *A*

Vertex label: $v(dfsNum,\ finishTime)$. tree back forward cross



Finish A $(finishTime = 5)$. **DFS done.**

Exercise 1 (b) – DFS Result



v	A	B	C	D	E
dfsNum	1	2	3	4	5
finish	5	4	3	1	2

Edge classification

- **Tree**: $A \to B$, $B \to C$, $C \to D$, $C \to E$
- **Back**: $D \to B$
- **Forward**: $A \to D$
- **Cross**: $E \to D$

Observation

The *back edge* $D \to B$ proves that G contains a **cycle**: $B \to C \to D \to B$.