Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CHAI
Humanities-Centered AI

Transformer-based Algorithmics

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

June 3, 2026

Version from June 2, 2026

Topics for Today

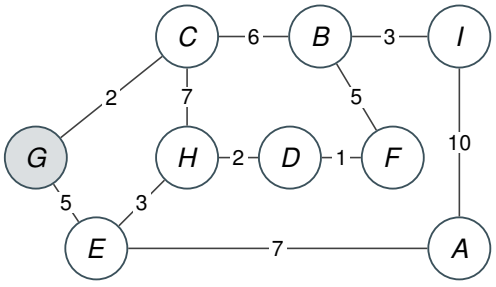
- Dijkstra's algorithm
- A* algorithm

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 – Dijkstra's Algorithm

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



Apply **Dijkstra's algorithm** from vertex G. After each while-loop iteration, state:

- the extracted vertex u (with $u.d$ and $u.parent$),
- the contents of the priority queue q (each v with $v.d$, $v.parent$).

Tie-breaking: on equal distances, keep the original order.

Finally, give the **shortest path** $G \to I$.

Dijkstra's Algorithm – Reminder

Dijkstra in a Nutshell

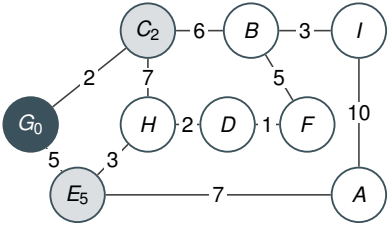
Computes **single-source shortest paths** in a graph with *non-negative* edge weights.

(1) Initialize $s.d = 0$, $s.parent = s$, and $v.d = \infty$ for every other vertex. Insert s into a priority queue q (key = tentative distance).

(2) While q is non-empty: extract the vertex u with smallest $u.d$ and *relax* every edge $u \to v$: if $u.d + w(u,v) < v.d$, set $v.d = u.d + w(u,v)$ and $v.parent = u$ (insert/update v in q).

- Once u is extracted from q, its distance $u.d$ is **final** – it will never decrease.
- The parent pointers form the **shortest-path tree** rooted at s.
- Runtime with a binary heap: $O((|V| + |E|) \log |V|)$.

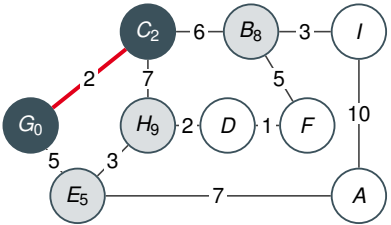
Exercise 1 – Dijkstra Trace from G



Extracted from queue			Current queue q		
u	$u.d$	$u.parent$	v	$v.d$	$v.parent$
G	0	G	C	2	G
			E	5	G

Step 1. Extract source G; discover C, E.

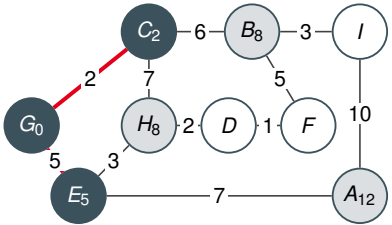
Exercise 1 – Dijkstra Trace from G



Extracted from queue			Current queue q		
u	$u.d$	$u.parent$	v	$v.d$	$v.parent$
G	0	G	C	2	G
			E	5	G
C	2	G	E	5	G
			B	8	C
			H	9	C

Step 2. Extract C; discover B, H.

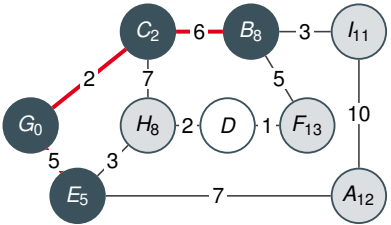
Exercise 1 – Dijkstra Trace from G



Step 3. Extract E; **relax** H: $9 \to 8$ (parent E); discover A.

Extracted from queue			Current queue q		
u	$u.d$	$u.parent$	v	$v.d$	$v.parent$
G	0	G	C	2	G
			E	5	G
C	2	G	E	5	G
			B	8	C
			H	9	C
E	5	G	B	8	C
			H	**8**	**E**
			A	12	E

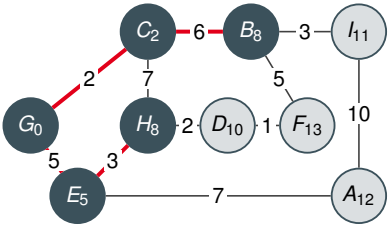
Exercise 1 – Dijkstra Trace from G



Step 4. Extract B; discover I, F.

Extracted from queue			Current queue q		
u	$u.d$	$u.parent$	v	$v.d$	$v.parent$
G	0	G	C	2	G
			E	5	G
C	2	G	E	5	G
			B	8	C
			H	9	C
E	5	G	B	8	C
			H	**8**	**E**
			A	12	E
B	8	C	H	8	E
			I	11	B
			A	12	E
			F	13	B

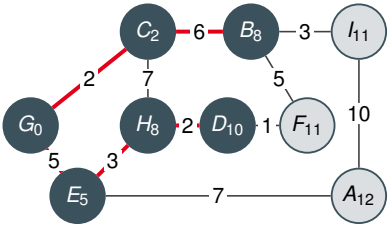
Exercise 1 – Dijkstra Trace from G



Step 5. Extract H; discover D.

Extracted from queue			Current queue q		
u	$u.d$	$u.parent$	v	$v.d$	$v.parent$
G	0	G	C	2	G
			E	5	G
C	2	G	E	5	G
			B	8	C
			H	9	C
E	5	G	B	8	C
			H	**8**	**E**
			A	12	E
B	8	C	H	8	E
			I	11	B
			A	12	E
			F	13	B
H	8	E	D	10	H
			I	11	B
			A	12	E
			F	13	B

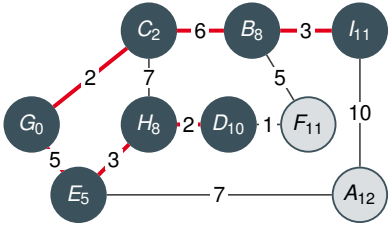
Exercise 1 – Dijkstra Trace from G



Step 6. Extract D; **relax** F: $13 \to 11$ (parent D).

Extracted from queue			Current queue q		
u	$u.d$	$u.parent$	v	$v.d$	$v.parent$
G	0	G	C	2	G
			E	5	G
C	2	G	E	5	G
			B	8	C
			H	9	C
E	5	G	B	8	C
			H	**8**	**E**
			A	12	E
B	8	C	H	8	E
			I	11	B
			A	12	E
			F	13	B
H	8	E	D	10	H
			I	11	B
			A	12	E
			F	13	B
D	10	H	I	11	B
			F	**11**	**D**
			A	12	E

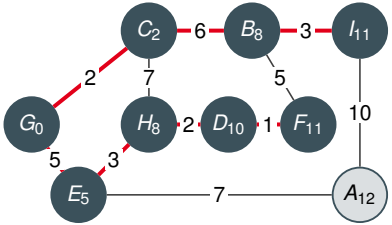
Exercise 1 – Dijkstra Trace from G



Step 7. Extract I; no neighbors improved.

Extracted from queue			Current queue q		
u	$u.d$	$u.parent$	v	$v.d$	$v.parent$
G	0	G	C	2	G
			E	5	G
C	2	G	E	5	G
			B	8	C
			H	9	C
E	5	G	B	8	C
			H	**8**	**E**
			A	12	E
B	8	C	H	8	E
			I	11	B
			A	12	E
			F	13	B
H	8	E	D	10	H
			I	11	B
			A	12	E
			F	13	B
D	10	H	I	11	B
			F	**11**	**D**
			A	12	E
I	11	B	F	11	D
			A	12	E

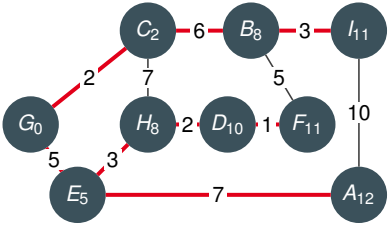
Exercise 1 – Dijkstra Trace from G



Step 8. Extract F; no neighbors improved.

Extracted from queue			Current queue q		
u	$u.d$	$u.parent$	v	$v.d$	$v.parent$
G	0	G	C	2	G
			E	5	G
C	2	G	E	5	G
			B	8	C
			H	9	C
E	5	G	B	8	C
			H	**8**	**E**
			A	12	E
B	8	C	H	8	E
			I	11	B
			A	12	E
			F	13	B
H	8	E	D	10	H
			I	11	B
			A	12	E
			F	13	B
D	10	H	I	11	B
			F	**11**	**D**
			A	12	E
I	11	B	F	11	D
			A	12	E
F	11	D	A	12	E

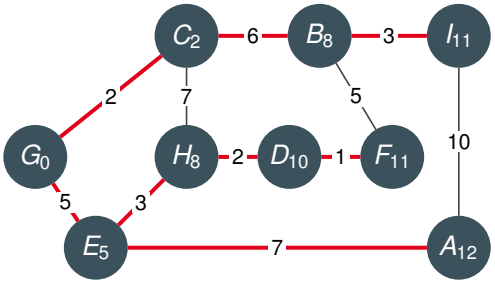
Exercise 1 – Dijkstra Trace from G



Step 9. Extract A. Queue empty – **done**.

Extracted from queue			Current queue q		
u	$u.d$	$u.parent$	v	$v.d$	$v.parent$
G	0	G	C	2	G
			E	5	G
C	2	G	E	5	G
			B	8	C
			H	9	C
E	5	G	B	8	C
			H	**8**	**E**
			A	12	E
B	8	C	H	8	E
			I	11	B
			A	12	E
			F	13	B
H	8	E	D	10	H
			I	11	B
			A	12	E
			F	13	B
D	10	H	I	11	B
			F	**11**	**D**
			A	12	E
I	11	B	F	11	D
			A	12	E
F	11	D	A	12	E
A	12	E	(empty)		

Exercise 1 – Result: Shortest-Path Tree from G



Vertex labels v_d show the final distance from G; **red** edges form the **shortest-path tree**.

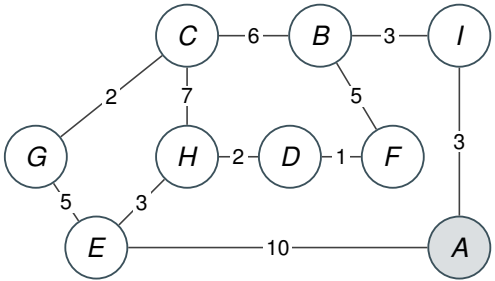
Shortest path $G \to I$:

$$G \xrightarrow{2} C \xrightarrow{6} B \xrightarrow{3} I$$

Total length: 11

Exercise 2 – A* Algorithm

Find the shortest path $A \to G$ in the (undirected) graph below using A* with the heuristic h_1.



Heuristic h_1 (estimate to G):

k	A	B	C	D	E	F	G	H	I
$h_1(k)$	6	6	1	7	2	6	0	1	11

After each iteration, give $k.g$, $k.f$ and $k.parent$ for each vertex.

Tie-breaking: on equal f, pick the lexicographical smallest vertex.

A* Algorithm – Reminder

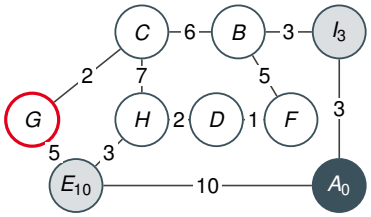
A* in a Nutshell

Best-first search guided by the **evaluation function** $f(n) = g(n) + h(n)$, where $g(n)$ is the cost from start s to n and $h(n)$ estimates the remaining cost from n to goal G. Maintain an open priority queue (key $= f$). Repeat: extract u with smallest $u.f$; if $u = G$, stop. Otherwise relax every edge $u \to v$: if $u.g + w(u, v) < v.g$, set $v.g$, $v.f = v.g + h(v)$, $v.parent = u$.

Heuristic properties

- **Admissible**: $h(n) \leq d^*(n, G)$ for all $n \Rightarrow$ A* finds an optimal path.
- **Optimal**: $h(n) = d^*(n, G)$ – the true remaining distance (perfectly informed).
- $h \equiv 0$: no guidance $\Rightarrow$ A* degenerates to **Dijkstra**.

Exercise 2 – A* Trace from *A* to *G*

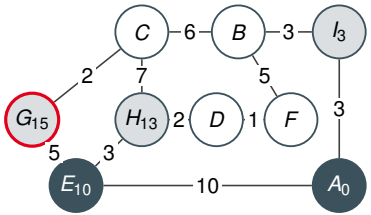


Step 1. Extract *A* (f=6); discover *E*, *I*.

k	$h(k)$	$k.g$	$k.f$	$k.parent$
A	6	0	6	–
B	6			
C	1			
D	7			
E	2	10	12	*A*
F	6			
G	0			
H	1			
I	11	3	14	*A*

current | in open queue | closed

Exercise 2 – A* Trace from *A* to *G*

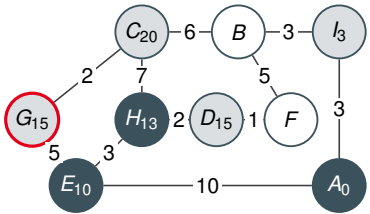


Step 2. Extract E (f=12); discover G, H.

k	$h(k)$	$k.g$	$k.f$	$k.parent$
A	6	0	6	–
B	6			
C	1			
D	7			
E	2	10	12	A
F	6			
G	0	15	15	E
H	1	13	14	E
I	11	3	14	A

current | in open queue | closed

Exercise 2 – A* Trace from *A* to *G*

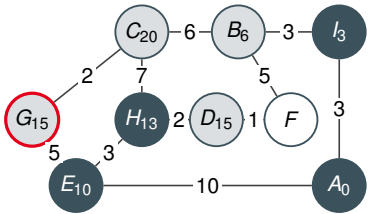


Step 3. Extract H (f=14, lex. before I); discover C, D.

k	$h(k)$	$k.g$	$k.f$	$k.parent$
A	6	0	6	–
B	6			
C	1	20	21	H
D	7	15	22	H
E	2	10	12	A
F	6			
G	0	15	15	E
H	1	13	14	E
I	11	3	14	A

current | in open queue | closed

Exercise 2 – A* Trace from *A* to *G*

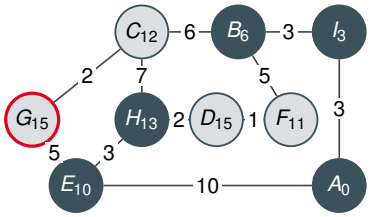


Step 4. Extract *I* (f=14); discover *B*.

k	$h(k)$	$k.g$	$k.f$	$k.parent$
A	6	0	6	–
B	6	6	12	*I*
C	1	20	21	*H*
D	7	15	22	*H*
E	2	10	12	*A*
F	6			
G	0	15	15	*E*
H	1	13	14	*E*
I	11	3	14	*A*

current | in open queue | closed

Exercise 2 – A* Trace from *A* to *G*

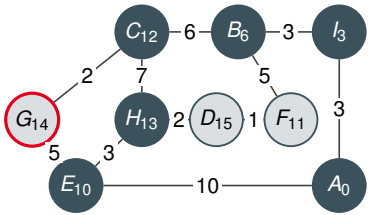


Step 5. Extract B (f=12); **update** C (21 → 13); discover F.

k	$h(k)$	$k.g$	$k.f$	$k.parent$
A	6	0	6	–
B	6	6	12	I
C	1	**12**	**13**	**B**
D	7	15	22	H
E	2	10	12	A
F	6	11	17	B
G	0	15	15	E
H	1	13	14	E
I	11	3	14	A

current | in open queue | closed

Exercise 2 – A* Trace from *A* to *G*

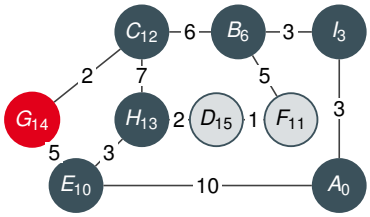


Step 6. Extract C (f=13); **update** G (15 → 14).

k	$h(k)$	$k.g$	$k.f$	$k.parent$
A	6	0	6	–
B	6	6	12	I
C	1	12	13	B
D	7	15	22	H
E	2	10	12	A
F	6	11	17	B
G	0	**14**	**14**	**C**
H	1	13	14	E
I	11	3	14	A

current | in open queue | closed

Exercise 2 – A* Trace from *A* to *G*

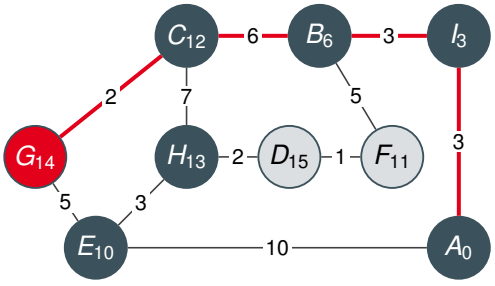


Step 7. Extract G (f=14) – **goal reached!**

k	$h(k)$	$k.g$	$k.f$	$k.parent$
A	6	0	6	–
B	6	6	12	*I*
C	1	12	13	*B*
D	7	15	22	*H*
E	2	10	12	*A*
F	6	11	17	*B*
G	0	14	14	*C*
H	1	13	14	*E*
I	11	3	14	*A*

current | in open queue | closed

Exercise 2 – Result: Shortest Path $A \to G$



Vertex labels v_g show the final g-value.

Shortest path $A \to G$:

$$A \xrightarrow{3} I \xrightarrow{3} B \xrightarrow{6} C \xrightarrow{2} G$$

Total length: $3 + 3 + 6 + 2 = 14$

Nodes D and F were *discovered* but never expanded – A* pruned them because their f-values exceeded the cost of the goal.

Exercise 3 – Evaluating Four Heuristics

Which of the following heuristics are admissible? Which are optimal?

k	A	B	C	D	E	F	G	H	I
$h_1(k)$	6	6	1	7	2	6	0	1	11
$h_2(k)$	12	8	2	10	5	11	0	8	**15**
$h_3(k)$	**12**	8	2	10	5	11	0	8	11
$h_4(k)$	0	0	0	0	0	0	0	0	0
$d^*(k, G)$	14	8	2	10	5	11	0	8	11

Exercise 3 – Evaluating Four Heuristics

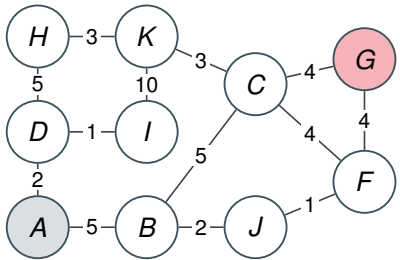
Which of the following heuristics are admissible? Which are optimal?

k	A	B	C	D	E	F	G	H	I
$h_1(k)$	6	6	1	7	2	6	0	1	11
$h_2(k)$	12	8	2	10	5	11	0	8	**15**
$h_3(k)$	**12**	8	2	10	5	11	0	8	11
$h_4(k)$	0	0	0	0	0	0	0	0	0
$d^*(k, G)$	14	8	2	10	5	11	0	8	11

- h_1: admissible (all $\leq d^*$), not optimal.
- h_2: **not admissible** – overestimates I ($15 > 11$); A* would commit to the wrong path ($f(G)=15 < 18 = f(I)$ after A, E expanded).
- h_3: matches d^* except at A ($12 < 14$); admissible. Setting $h_3(A)=14$ makes it optimal.
- $h_4 \equiv 0$: trivially admissible – exactly Dijkstra.

Exercise 4 – Heuristic Design for A*

Consider a different weighted graph (10 vertices, target G).



Given heuristic h (estimate to G):

k	A	B	C	D	F	G	H	I	J	K
$h(k)$	16	5	1	15	4	0	6	11	5	4

(a) Assess h regarding **admissibility** and **optimality**.

(b) Give an admissible *and* an optimal heuristic.

(c) How are A* and Dijkstra related?

Exercise 4 – Solution

(a) The given h is neither admissible nor optimal.

- Not admissible: $h(A)=16 > 12 = d^*(A, G)$ and $h(D)=15 > 14 = d^*(D, G)$ – *overestimates*.
- Not optimal: $h(C)=1 < 4 = d^*(C, G)$ – *underestimates*.

(b) Optimal and admissible heuristics:

k	A	B	C	D	F	G	H	I	J	K
$h_{\text{opt}}=d^*$	12	7	4	14	4	0	10	15	5	7
h_{adm}	12	5	1	14	4	0	6	11	5	4

Any h with $0 \leq h(k) \leq d^*(k, G)$ is admissible.

(c) A* vs. Dijkstra. Dijkstra is exactly A* with $h \equiv 0$ – it expands purely by g. An informed admissible heuristic guides A* towards the goal while preserving optimality.