Informed Route Planning Algorithms

Jonas Blocher

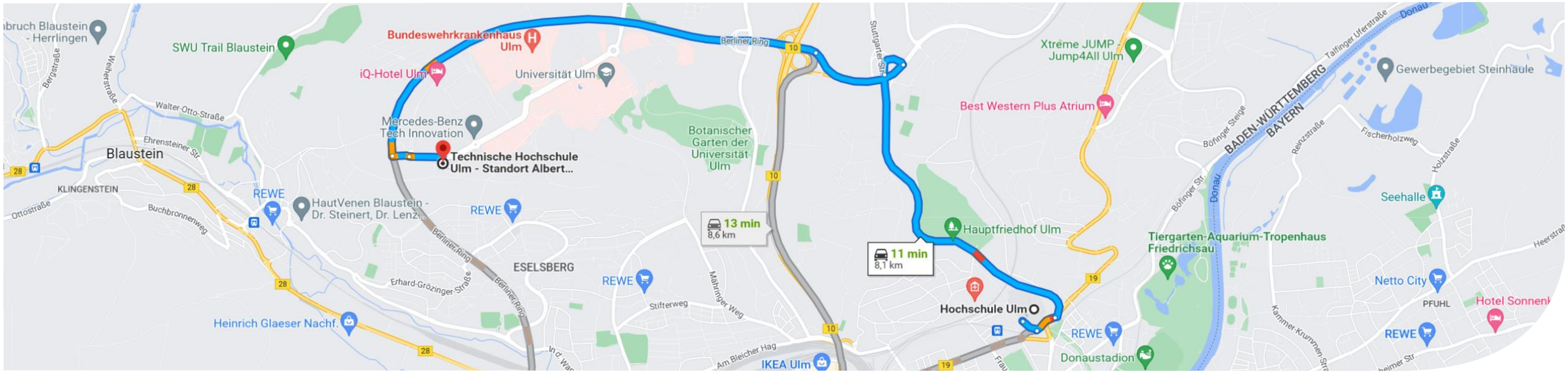


Fig. 1 https://goo.gl/maps/he7dryepb1nauwYZ6

Agenda

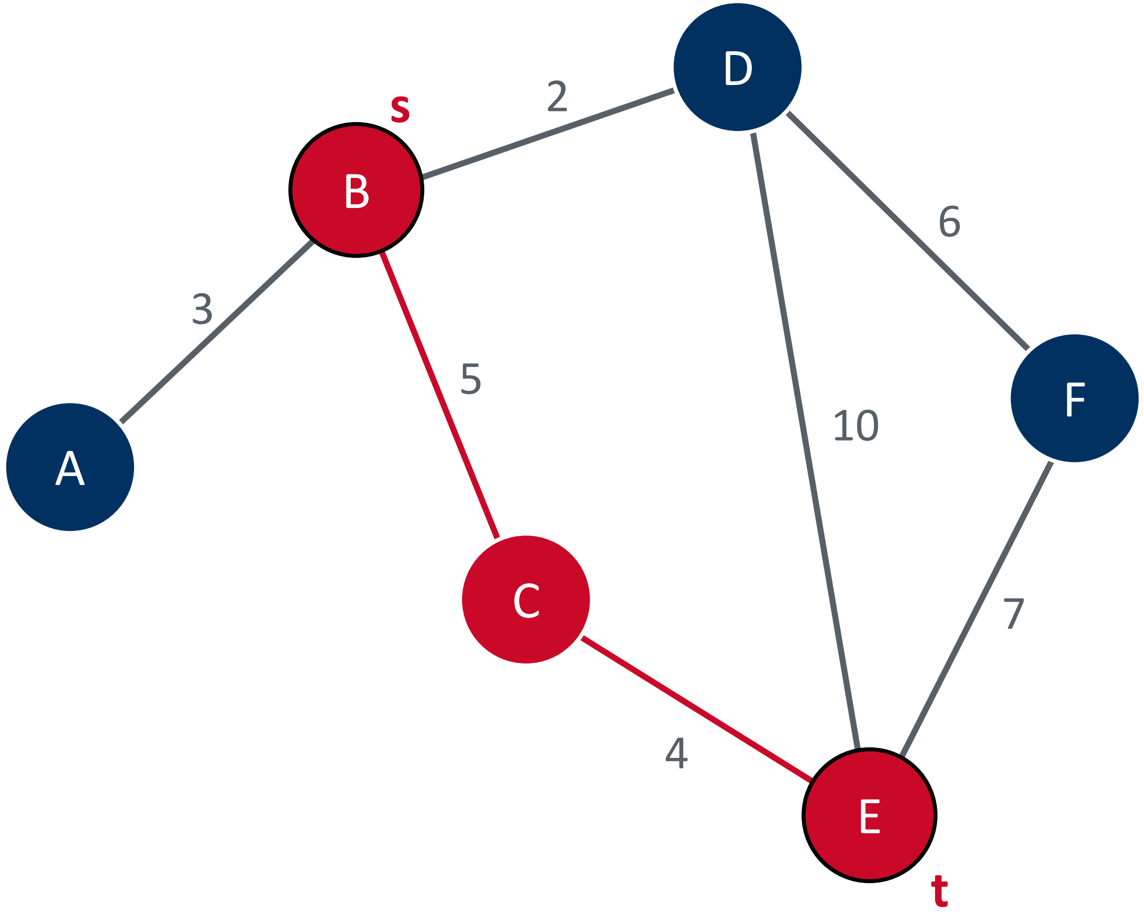
1. Shortest Path Problem
2. Best-first Search
3. Dijkstra‘s Algorithm
4. A* Algorithm
5. Speedup Techniques
6. Efficiency Comparison
7. Conclusion

Fig. 2 https://uncrate.com/exploride-heads-up-car-display/

Shortest Path Problem

Definition

- Graph $G = (N, E)$
- Nodes $N = \{A, B, C, D, E, F\}$
- Edges $E = \{\{A, B\}, \{B, C\}, \{B, D\}, \dots\}$
- Each edge $\{u, v\}$ has weight value $w(u, v)$
- For each path $(s, \dots, t)$ total cost $c(s, t)$ can be calculated
- "Shortest Path" is path with **minimal cost**



Shortest Path Problem

Practical Applications

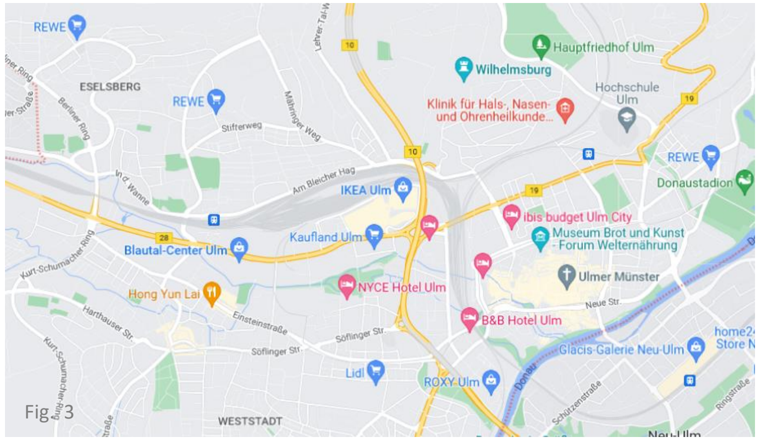


Fig. 3


Fig. 4

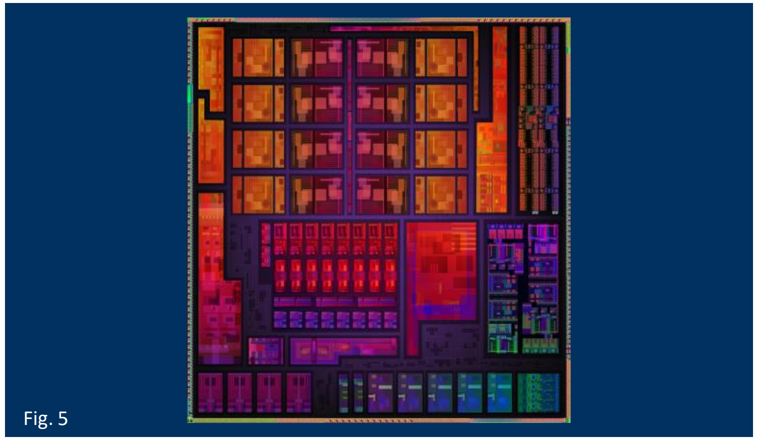


Fig. 5


Fig. 6

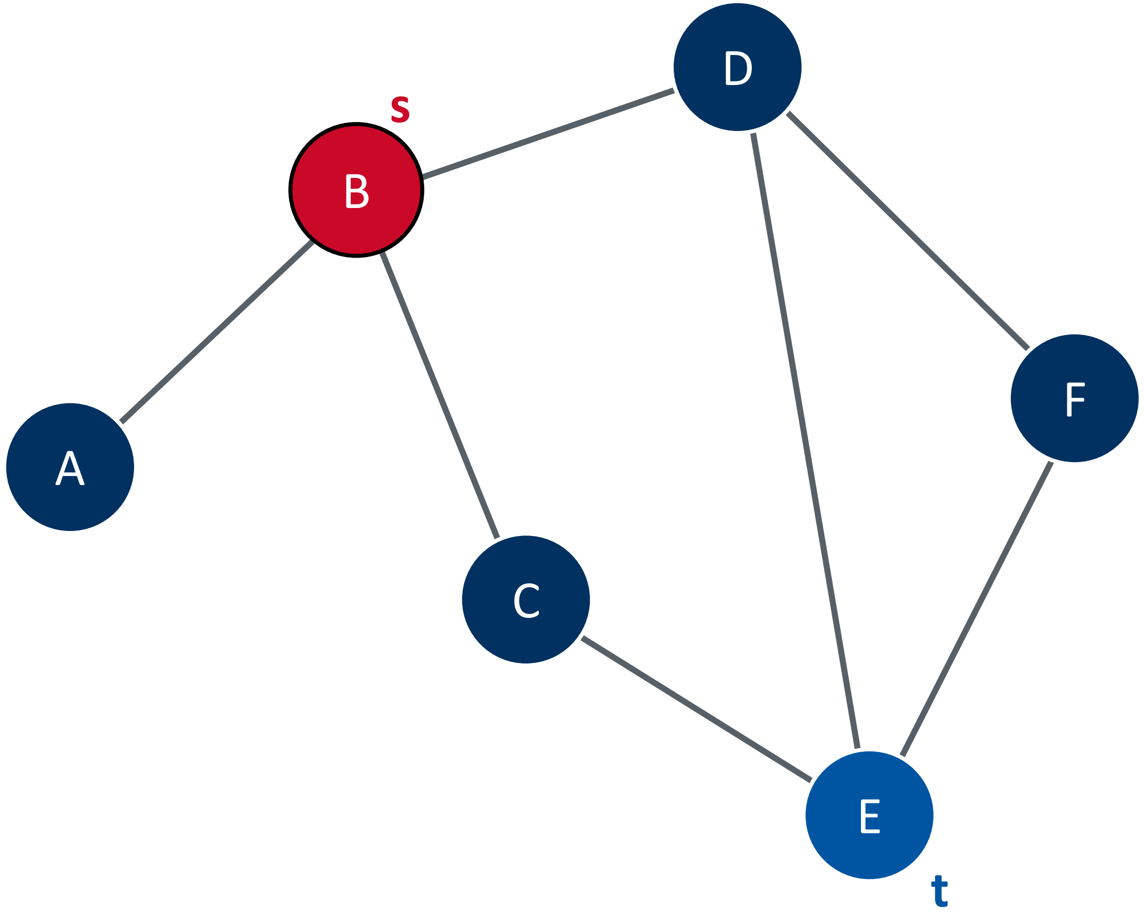

Fig. 7


Fig. 8

Fig. 3 https://goo.gl/maps/X3dcQ5f7YqhbkP17A
Fig. 4 https://www.viro-group.com/de/specials/cable-routing/
Fig. 5 https://library.amd.com/media/?mediaId=DE133045-FEF1-4B57-A4741EC209817C03
Fig. 6 https://3dswiss.net/luftaufnahmen/
Fig. 7 https://www.supplychainbrain.com/blogs/1-think-tank/post/28906-the-robotic-future-of-manufacturing
Fig. 8 https://de.wikipedia.org/wiki/Zauberw%C3%BCrfel#/media/Datei:Rubik's_cube_v3.svg

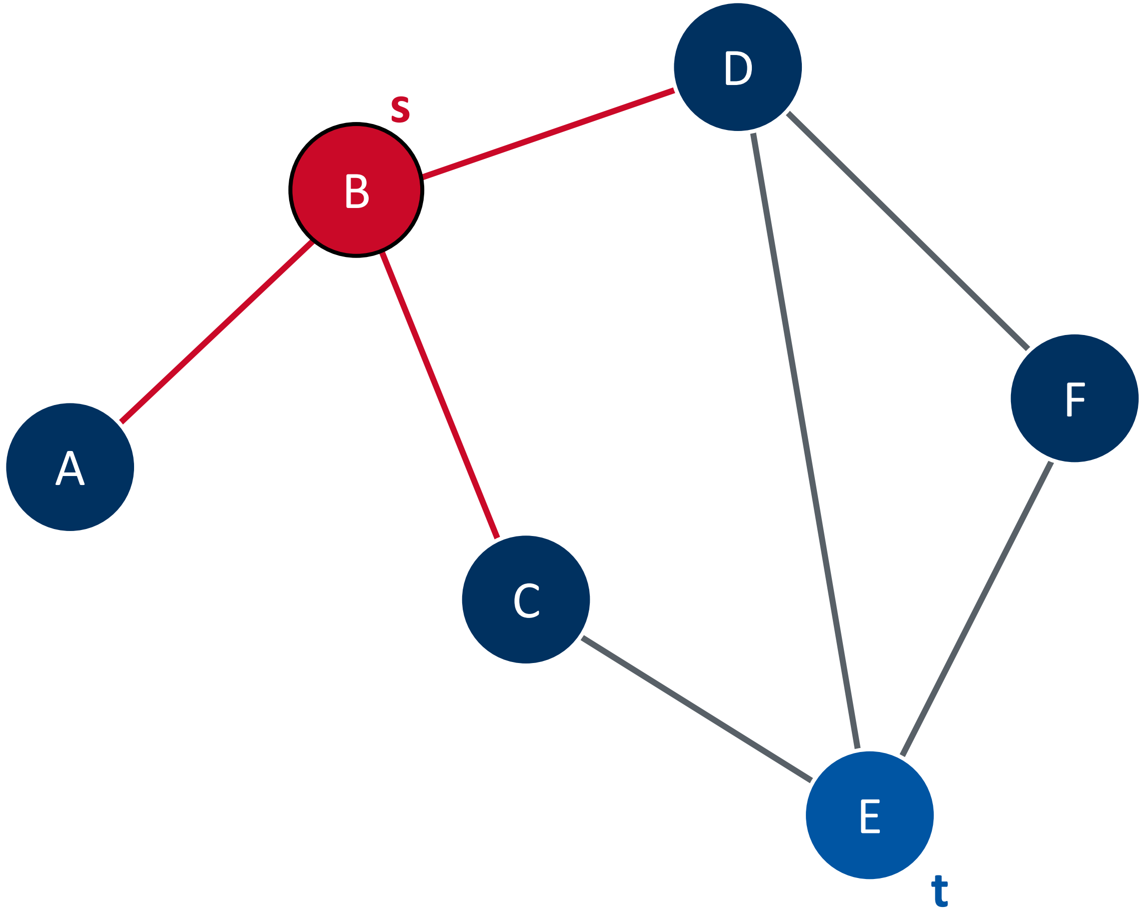
Best-first Search

- Superimpose search tree over graph
- Start node is root
- Leaf nodes are expanded
  - New neighbors added to the tree as child nodes



Best-first Search

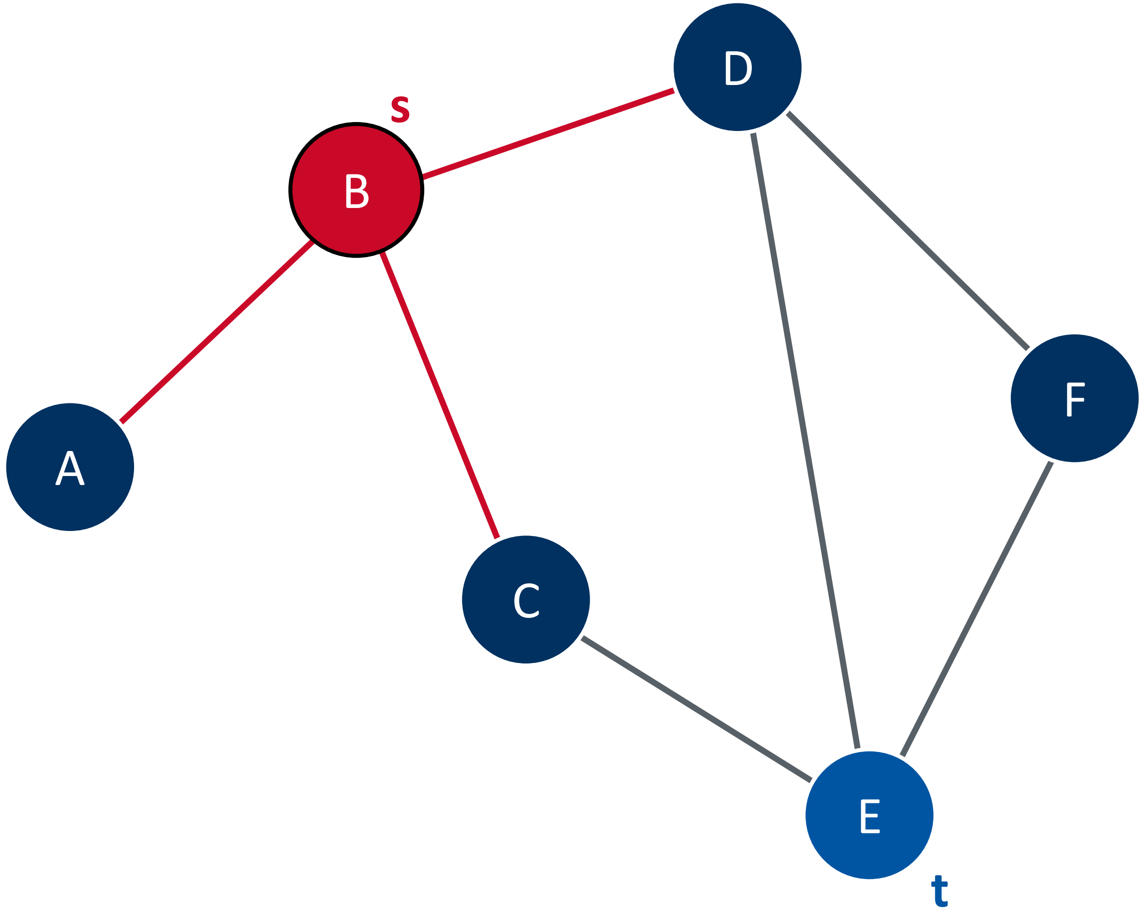
- Superimpose search tree over graph
- Start node is root
- Leaf nodes are expanded
  - New neighbors added to the graph as child nodes
- **Interior** region with already expanded nodes $\{B\}$
- **Exterior** region with unreached nodes $\{E, F\}$
- **Frontier** with unexpanded leave nodes $\{A, C, D\}$



Which node to expand first?

Best-first Search

- Superimpose search tree over graph
- Start node is root
- Leaf nodes are expanded
  - New neighbors added to the graph as child nodes
- **Interior** region with already expanded nodes $\{B\}$
- **Exterior** region with unreached nodes $\{E, F\}$
- **Frontier** with unexpanded leave nodes $\{A, C, D\}$



Which node to expand first?

- Evaluation function $f(n)$
- Expand "Best" Node with minimal $f(n)$ first

Dijkstra‘s Algorithm

Uniform cost search

- Path cost from root s to node n is evaluation function
  - $f(n) = c(s, n)$

Edsger W. Dijkstra, 2002

Fig. 9 https://de.wikipedia.org/wiki/Edsger_W._Dijkstra#/media/Datei:Edsger_Wybe_Dijkstra.jpg

Dijkstra‘s Algorithm

Example

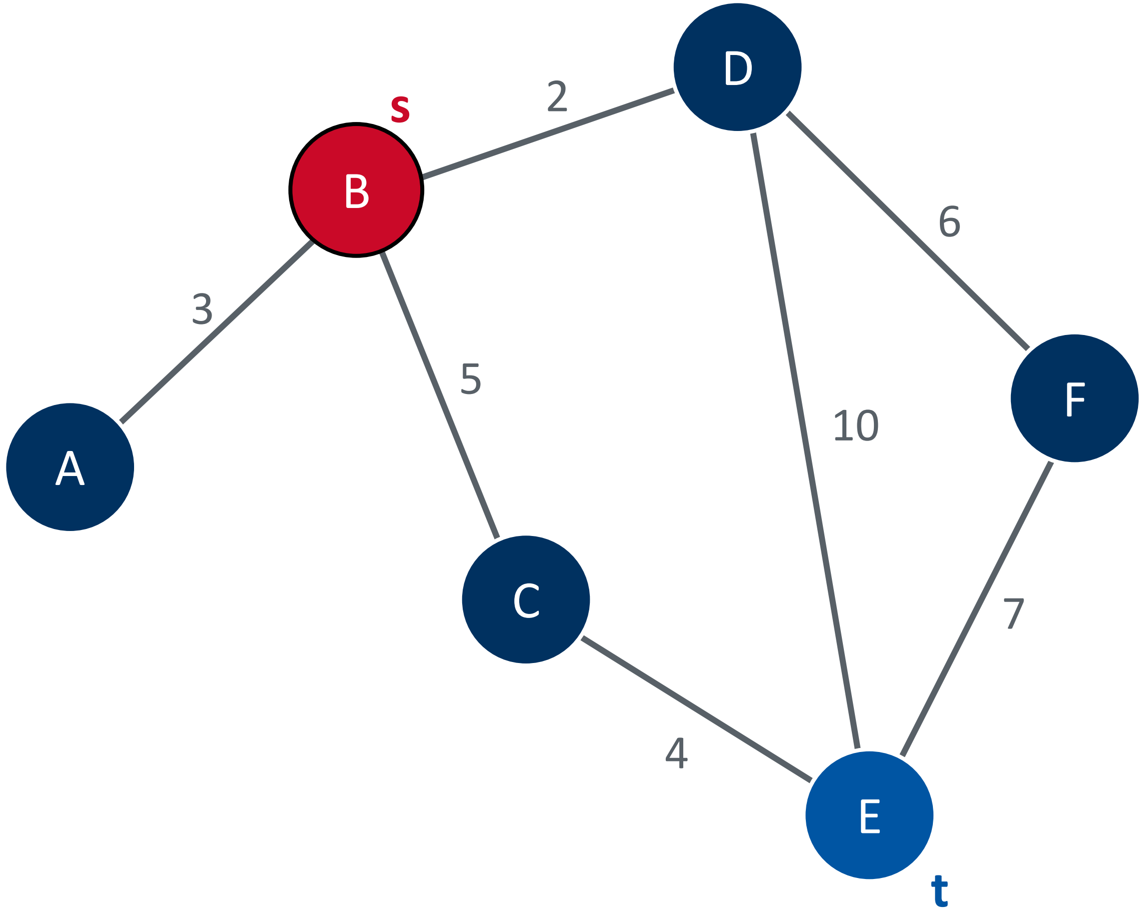
After expanding B:

Frontier $\{A, C, D\}$

$$f(A) = c(B, A) = 3$$
$$f(C) = c(B, C) = 5$$
$$f(D) = c(B, D) = 2$$

$$f(D) < f(A) < f(C)$$

- Expand D



Dijkstra‘s Algorithm

Example

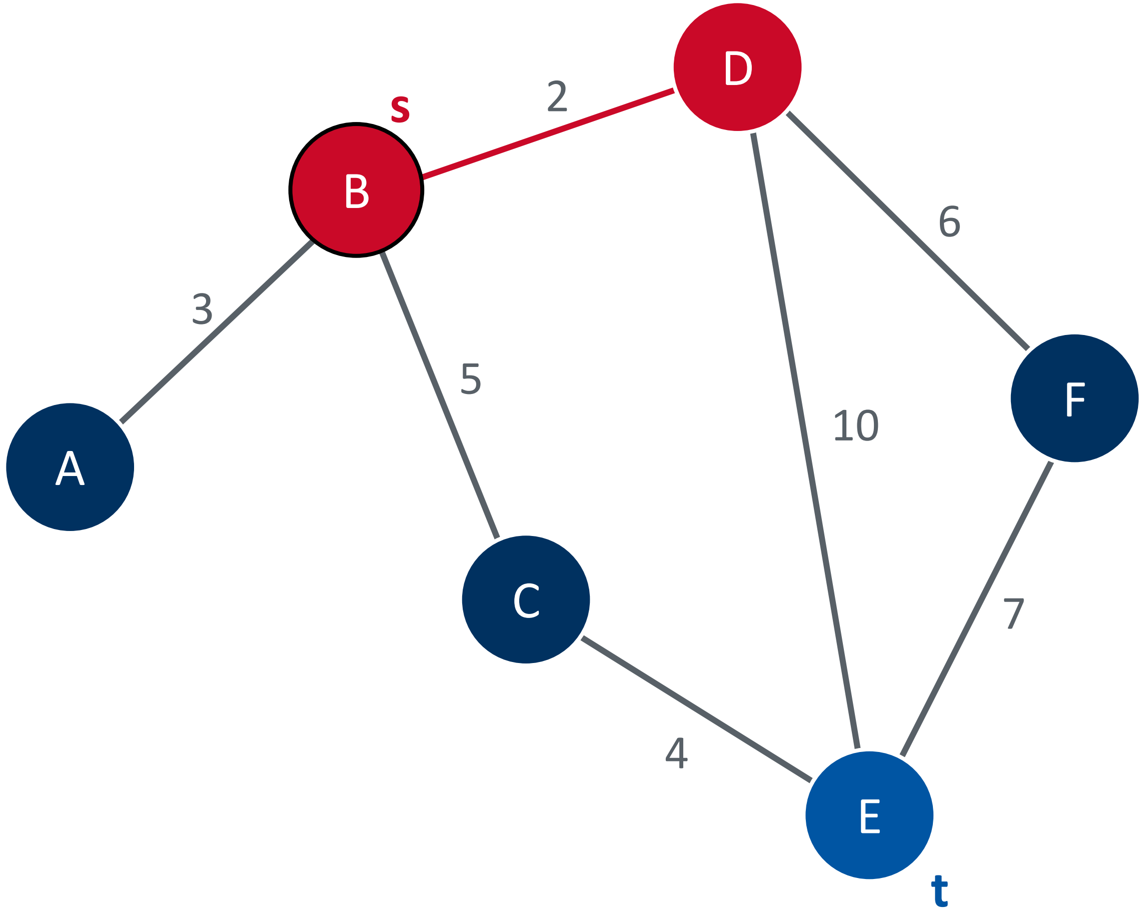
After expanding D:

Frontier $\{A, C, E, F\}$

$$f(A) = c(B, A) = 3$$
$$f(C) = c(B, C) = 5$$
$$f(E) = c(B, E) = 2 + 10 = 12$$
$$f(F) = c(B, F) = 2 + 6 = 8$$

$$f(A) < f(C) < f(F) < f(E)$$

- Expand A



Dijkstra‘s Algorithm

Example

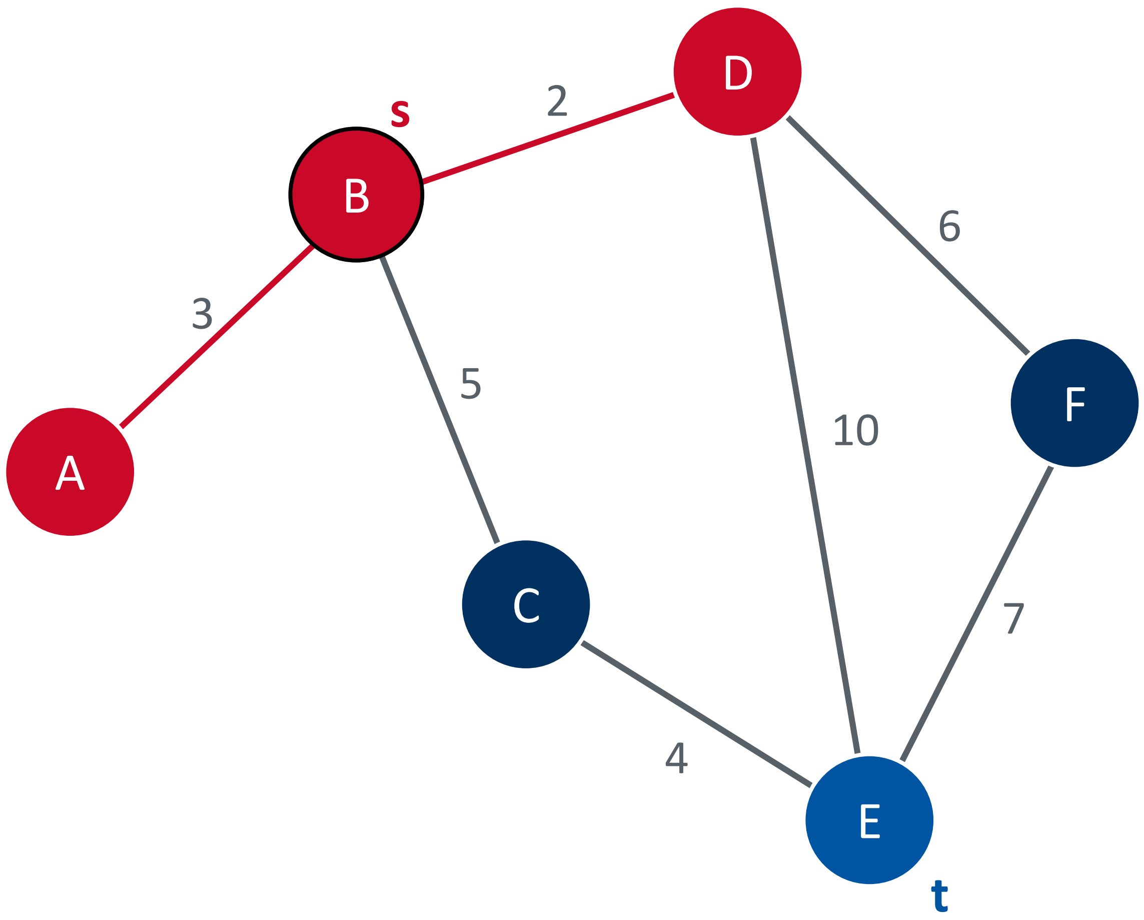
After expanding A:

Frontier $\{C, E, F\}$

$$f(C) = c(B,C) = 5$$
$$f(E) = c(B,E) = 2 + 10 = 12$$
$$f(F) = c(B,F) = 2 + 6 = 8$$

$$f(C) < f(F) < f(E)$$

- Expand C



Dijkstra‘s Algorithm

Example

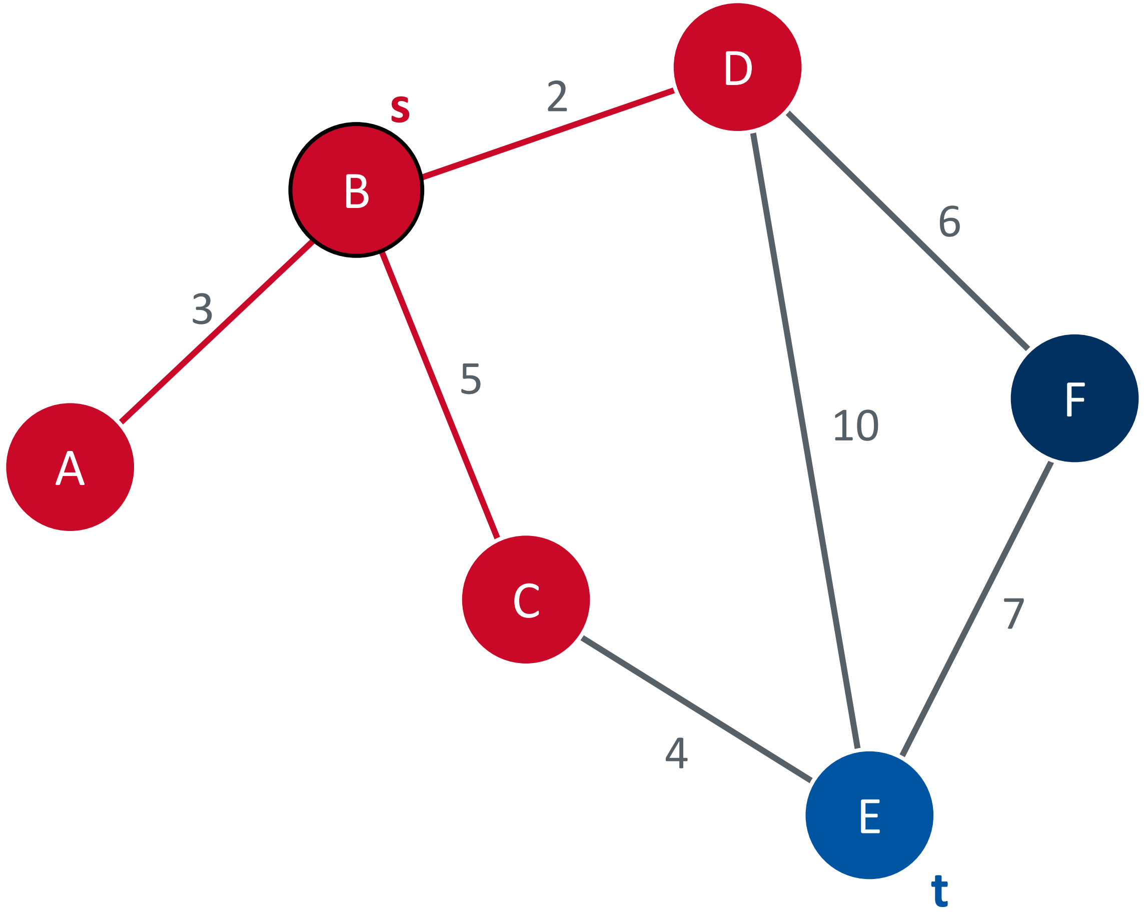
After expanding C:

Frontier $\{E, F\}$

$$f(E) = c(B, E) = 5 + 4 = 9$$
$$f(F) = c(B, F) = 2 + 6 = 8$$

$$f(F) < f(E)$$

- Expand F



Dijkstra‘s Algorithm

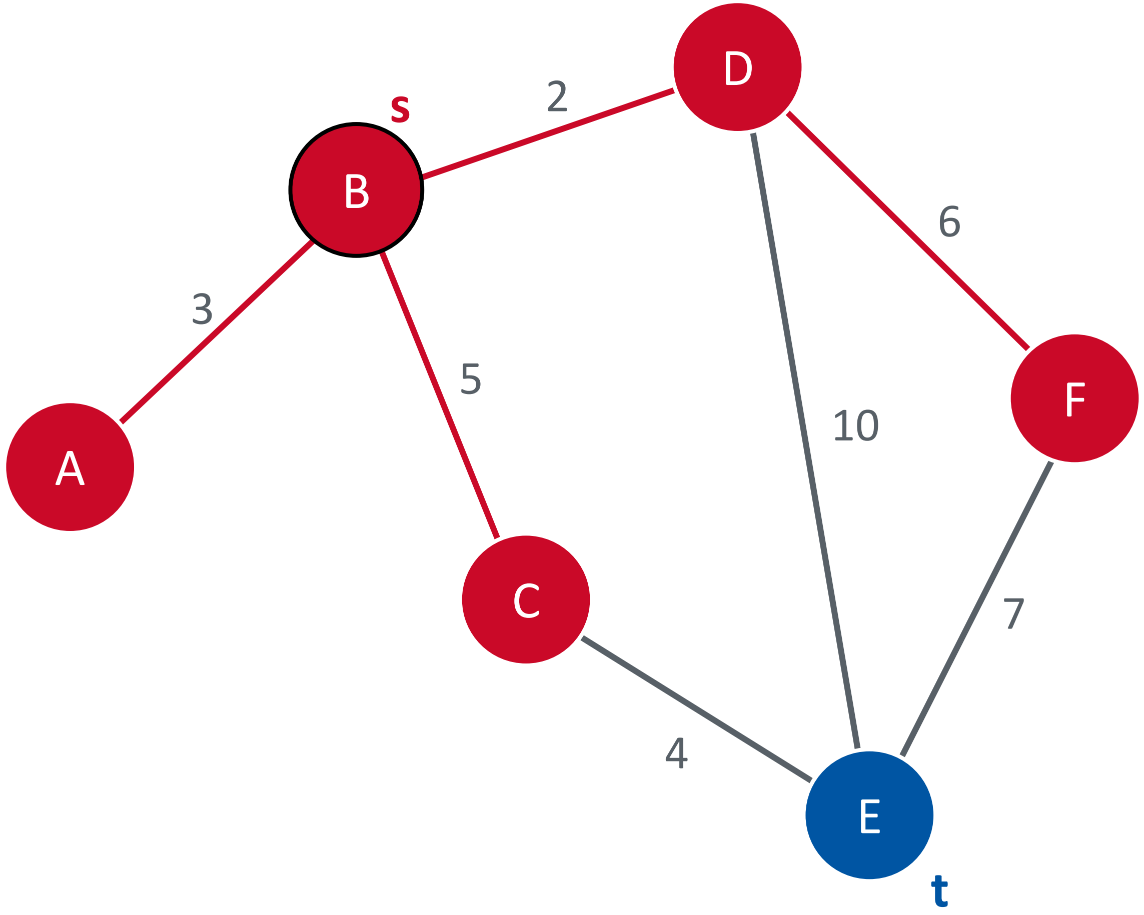
Example

After expanding F:

Frontier $\{E\}$

$$f(E) = c(B, E) = 5 + 4 = 9$$

- Expand E



Dijkstra's Algorithm

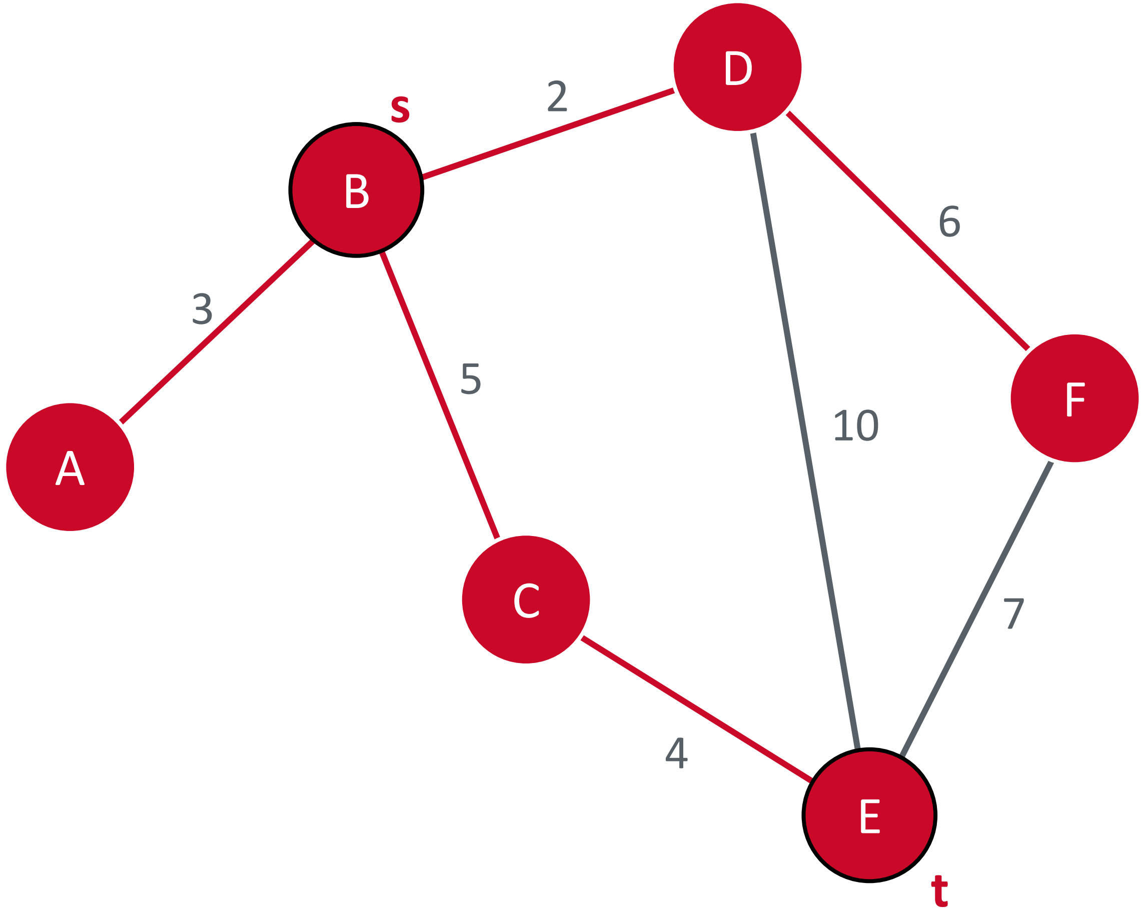
Example

After expanding F:

Frontier $\{E\}$

$$f(E) = c(B, E) = 5 + 4 = 9$$

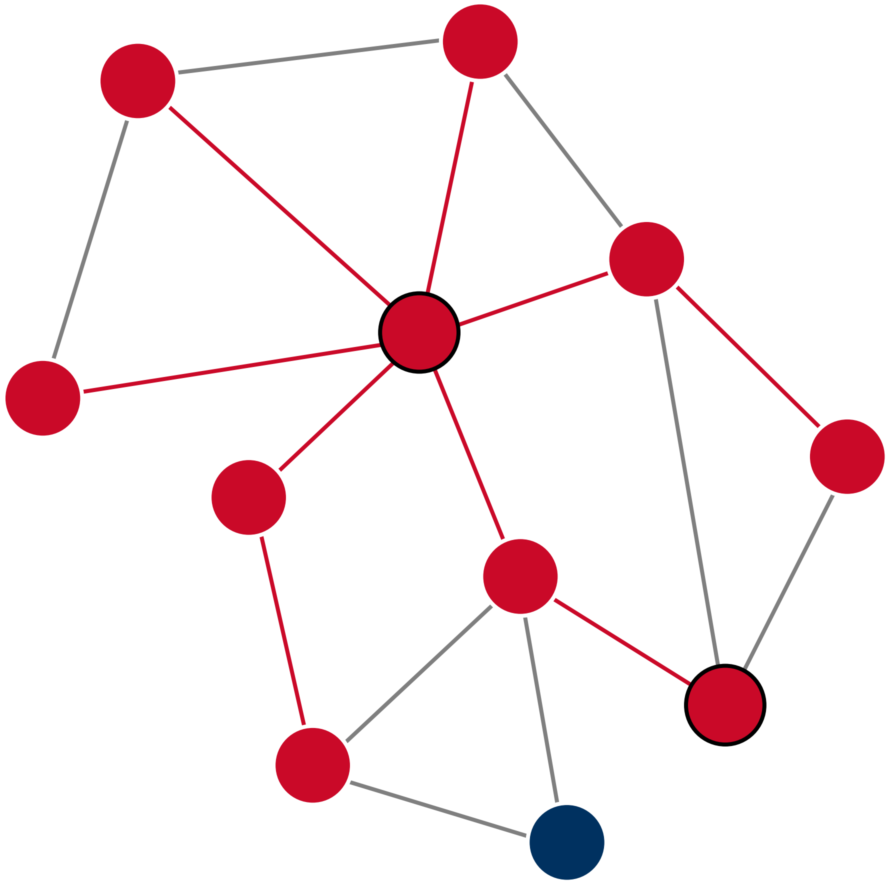
- Expand E
  - Goal reached
  - Shorthest path is (B, C, E)



Dijkstra‘s Algorithm

Evaluation

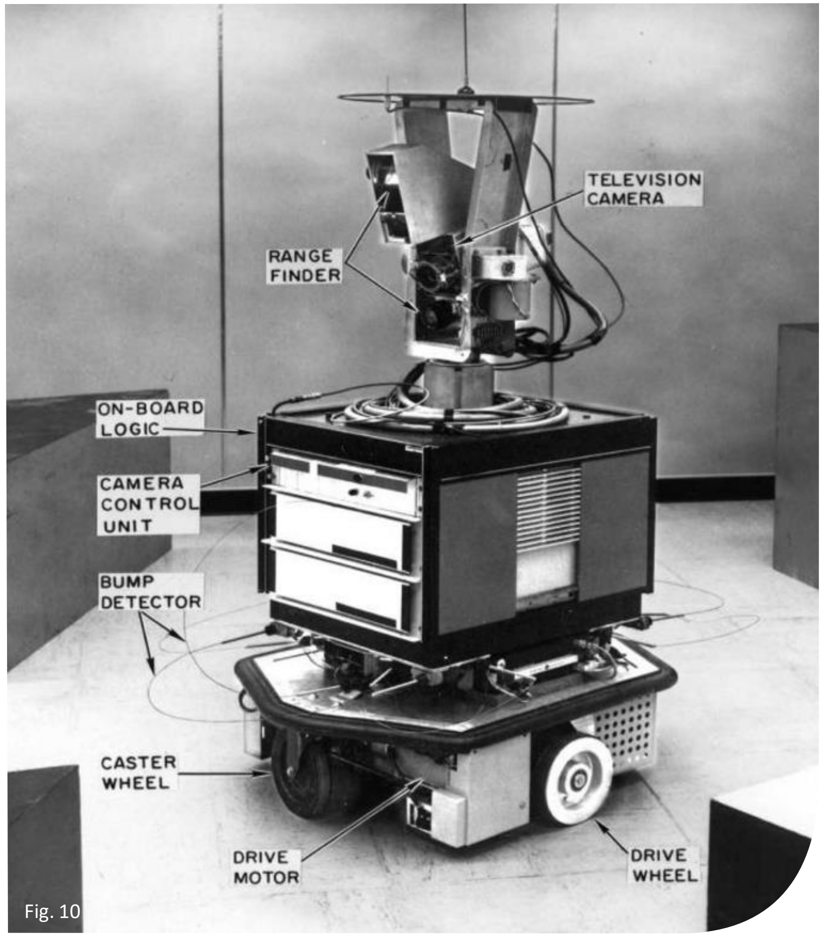
- Is **complete**
  - Guaranteed to find solution if there is one & guaranteed to report failure
- Is **cost-optimal**
  - Solution is guaranteed to be shortest path
- High **time complexity**
  - Nodes are expanded with uniform cost
  - Circular expansion outwards from root
  - Not directed towards target



A* Algorithm

Goal directed search

- Shortest Path leads towards the target
- Use heuristic for guidance
  - $f(n) = g(n) + h(n)$



Shakey-the-robot

Fig. 10 https://www.researchgate.net/figure/Shakey-the-robot-Probably-the-first-mobile-robot-It-was-developed-at-the-Artificial_fig2_259503287

A* Algorithm

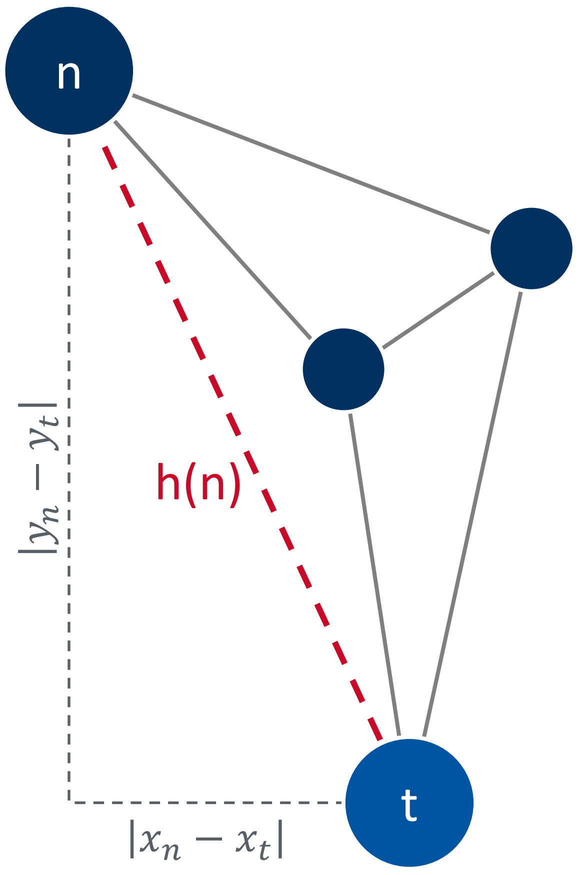
Heuristics

- Estimated cost of shortest path from node to target
- **Admissible heuristics** never overestimate cost to reach the goal
- **Inadmissible heuristics** can return bigger values than the actual cost
  - Only admissible heuristics are cost-optimal

- Common in routing: **Euclidean distance**
  - $h(n) = \sqrt{(x_n - x_t)^2 + (y_n - y_t)^2}$

- $h(n) = 0 \rightarrow f(n) = g(n)$ is Dijkstra‘s Algorithm

- Adding **weight** to the heuristic (inadmissible but faster):
  - $f(n) = g(n) + W \times h(n)$ $\quad W > 0$



A* Algorithm

Example

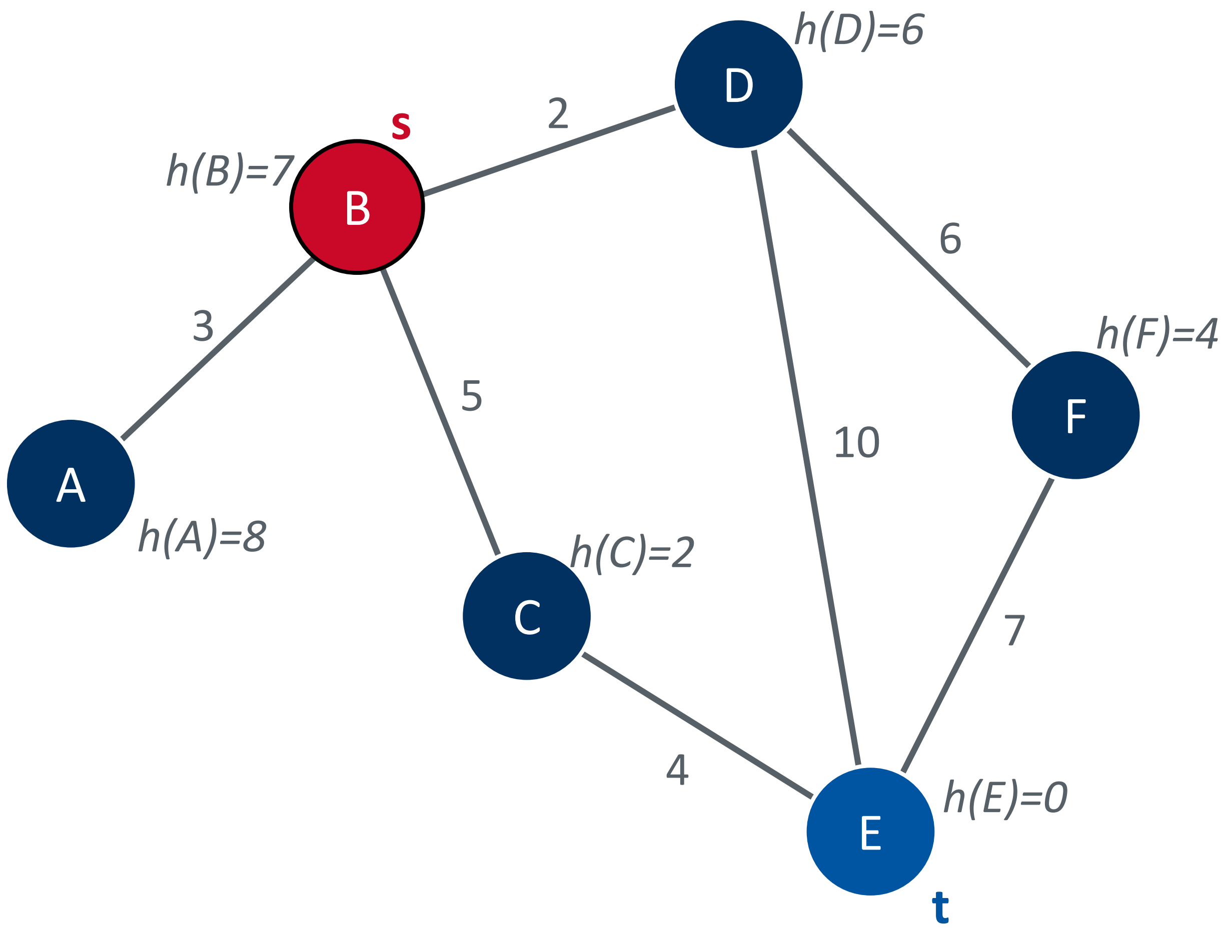
After expanding B:

Frontier $\{A, C, D\}$

$$f(A) = c(B, A) + h(A) = 3 + 8 = 11$$
$$f(C) = c(B, C) + h(C) = 5 + 2 = 7$$
$$f(D) = c(B, D) + h(D) = 2 + 6 = 8$$

$$f(C) < f(D) < f(A)$$

- Expand C



A* Algorithm

Example

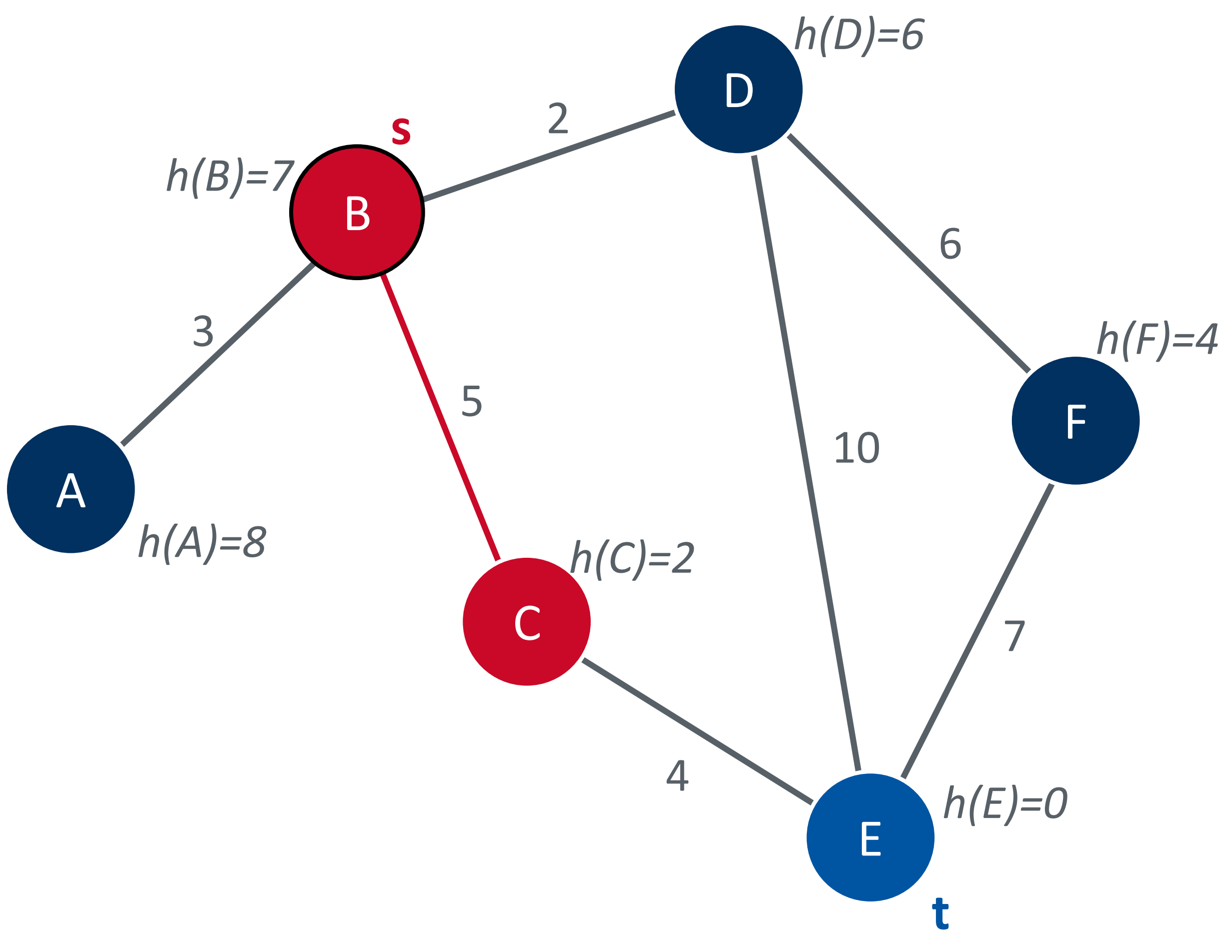
After expanding C:

Frontier $\{A, D, E\}$

$$f(A) = c(B,A) + h(A) = 3 + 8 = 11$$
$$f(D) = c(B,D) + h(D) = 2 + 6 = 8$$
$$f(E) = c(B,E) + h(E) = 9 + 0 = 9$$

$$f(D) < f(E) < f(A)$$

- Expand D



A* Algorithm

Example

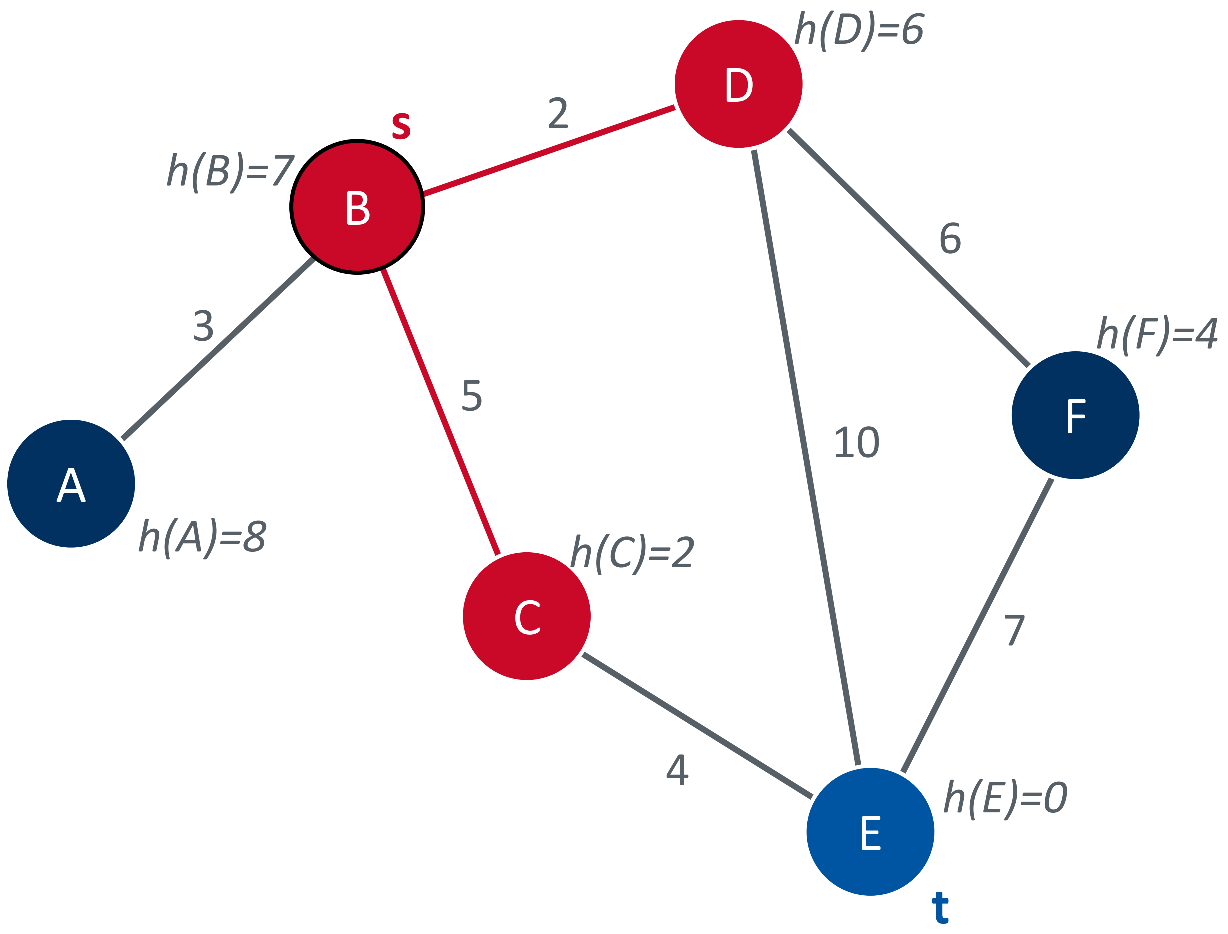
After expanding D:

Frontier $\{A, E, F\}$

$$f(A) = c(B, A) + h(A) = 3 + 8 = 11$$
$$f(E) = c(B, E) + h(E) = 9 + 0 = 9$$
$$f(F) = c(B, F) + h(F) = 8 + 4 = 12$$

$$f(E) < f(A) < f(F)$$

- Expand E



A* Algorithm

Example

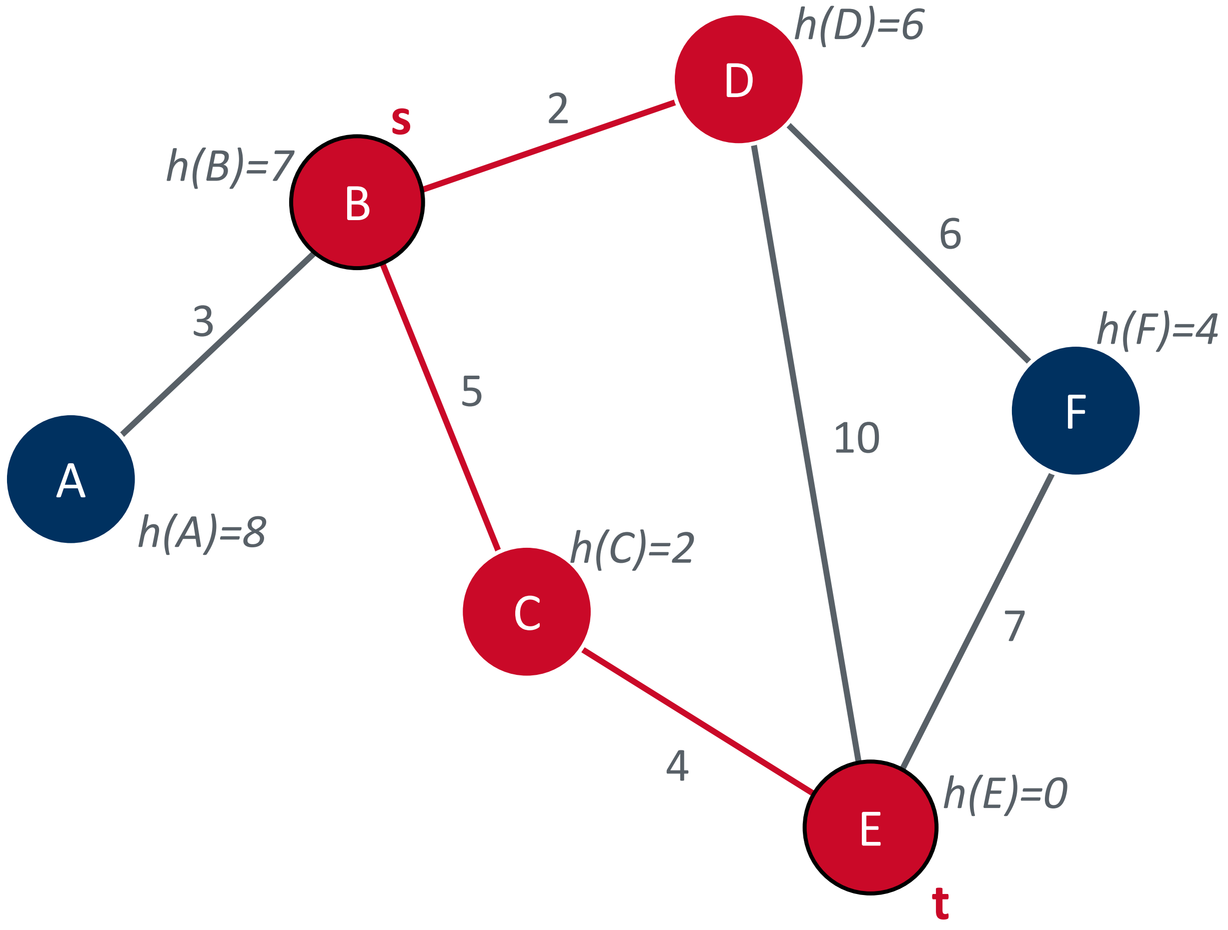
After expanding D:

Frontier $\{A, E, F\}$

$$f(A) = c(B, A) + h(A) = 3 + 8 = 11$$
$$f(E) = c(B, E) + h(E) = 9 + 0 = 9$$
$$f(F) = c(B, F) + h(F) = 8 + 4 = 12$$

$$f(E) < f(A) < f(F)$$

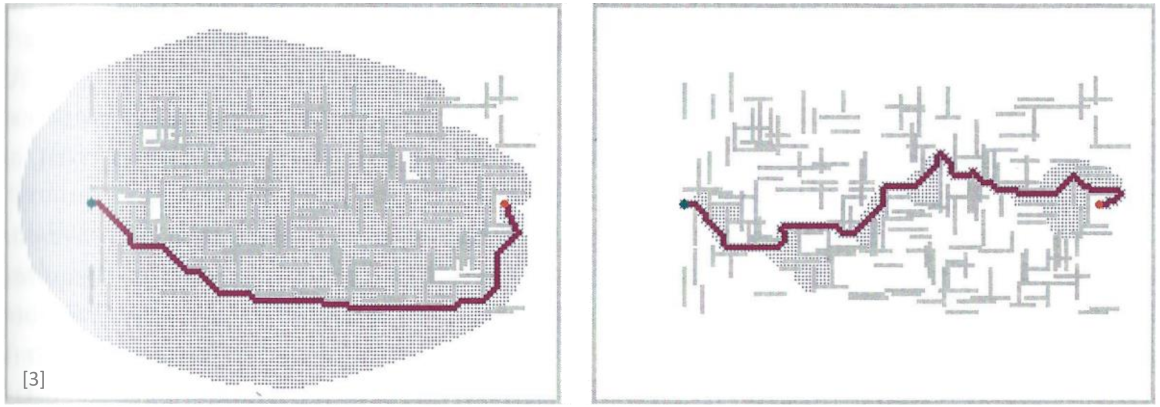
- Expand E
  - Goal reached
  - Shortest path is (B, C, E)



A* Algorithm

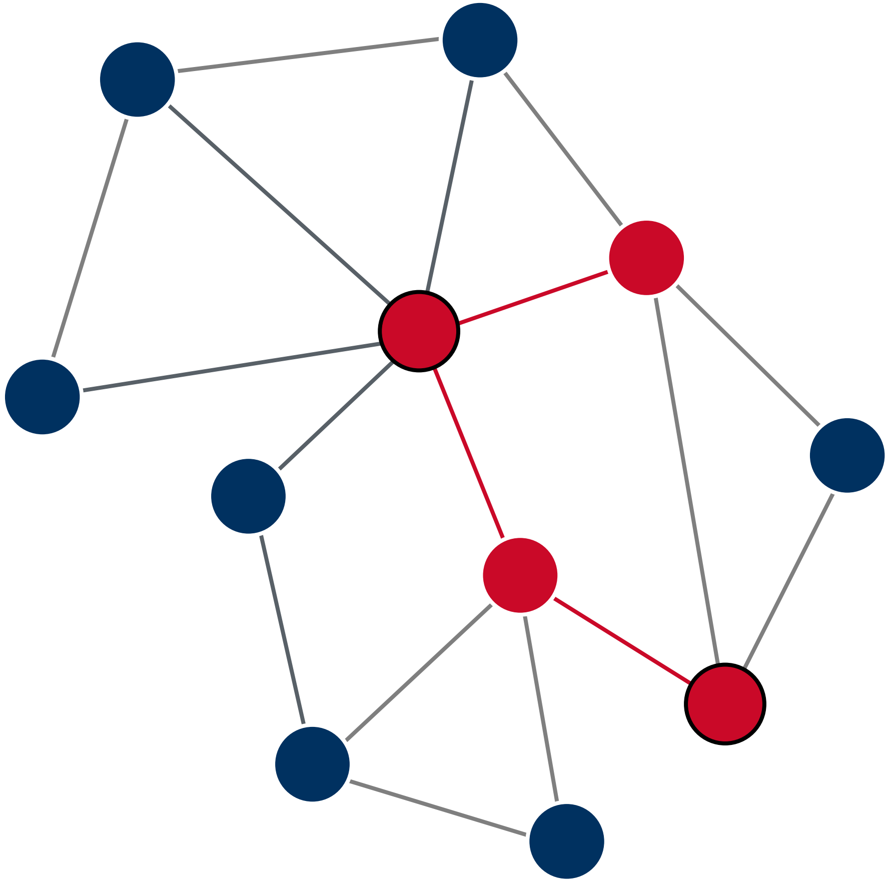
Evaluation

- Is **complete**
- Is **cost-optimal** if an **admissible heuristic** was used
- Lower **time complexity** than Dijkstra's alg.
  - Nodes are expanded towards target
  - Less Nodes to expand
  - Dependent on accuracy of heuristic
  - Weighted A* is even faster, while not cost-optimal



[3]

Comparison of A* and weighted A* with W=2



Speedup Techniques

Landmarks

- A* still million times slower than modern navigation systems
- Landmarks spread around perimeter of network
- Precomputation of shortest paths to landmarks
- Heuristic based on cost from node to landmark $C^*(n, L)$
  - $h_{DH}(n) = \max_{L \in Landmarks} |C^*(n, L) - C^*(t, L)|$

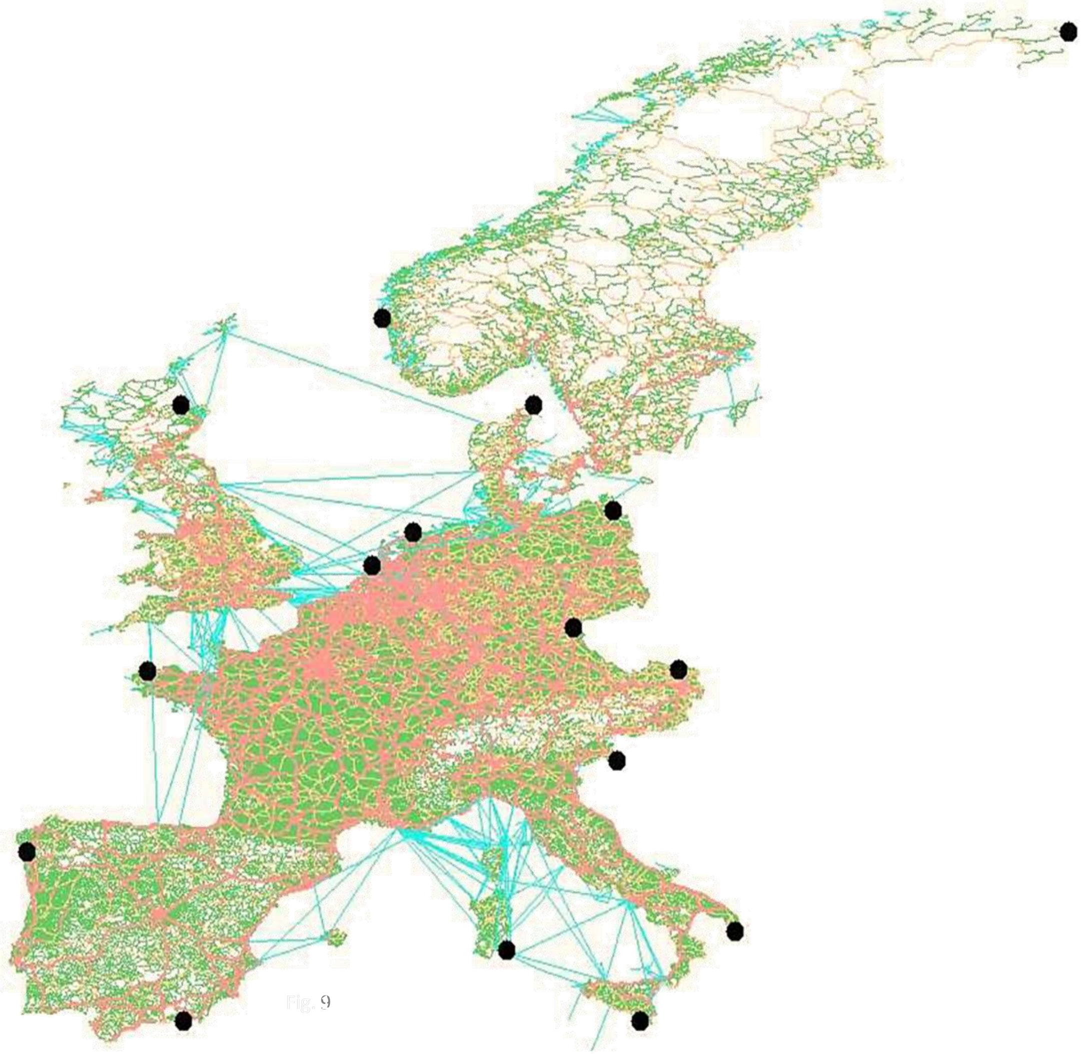


Fig. 11 http://algo2.iti.kit.edu/documents/routeplanning/hhStarSubmit.pdf

Speedup Techniques

Highway Hierarchies

- Utilizing hierarchical structure of road networks
- Defining neighbourhoods & highway network interconnecting the neighbourhoods
- Outside of neighbourhood, only important nodes are considered, bypassable nodes are contracted
- Multiple levels of highway networks in hierarchical order
- Greatly reduces number of nodes
- **Distance tables** can hold precomputed cost for all nodes in highway network

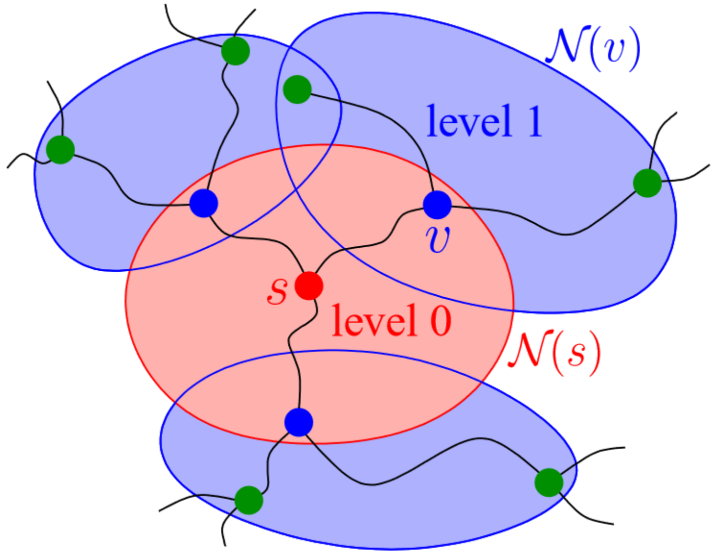


Fig. 12 http://algo2.iti.kit.edu/documents/routeplanning/hhStarSubmit.pdf

Efficiency Comparison

method	pub. date mm/yy	space [B/n]	preproc. [min]	speedup
Dijkstra’s Algorithm	08/59	21	0	1
A* + Landmarks	07/04	89	13	28
Highway Hierarchies	04/05	49	161	2 645
A* + Highway Hierarchies + Distance Tables	08/06	92	14	12 902
…				
Transit Nodes + Edge Flags	01/08	341	229	3 327 327

Conclusion

- Information about the network can be used to create optimized algorithms
- A* is able to reduce time-complexity compared to Dijkstra’s Algorithm
- If an admissible heuristic is used, A* is cost-optimal and complete
- Modern techniques such as landmarks or highway hierarchies and distance tables can further speed up search queries up to 12 000 times

QUESTIONS TIME

Let’s start the discussion!