CS 369: Introduction to Robotics

Prof. Thao Nguyen

Spring 2026

HAVERFORD COLLEGE

Admin

- Lab 3 due tonight

- Lab 4 posted:
    - group work
    - due Tuesday (Feb 24)

HIP

Haverford **Innovations** Program

SUMMER INNOVATION Incubator 2026!

Your Idea

8Weeks

Work Stipend

(BiCo)Solos & Teams

Seed Funds

Agile Coaches

Industry Mentors

Time+Resources

Community

Support

Apply Now!

APPLICATIONS CLOSE:

Feb 24 2026 11:59 PM

All ideas welcome!

? snickel@hc

Outline for today

- Motion planning algorithms
  - Graph search
  - Sampling-based

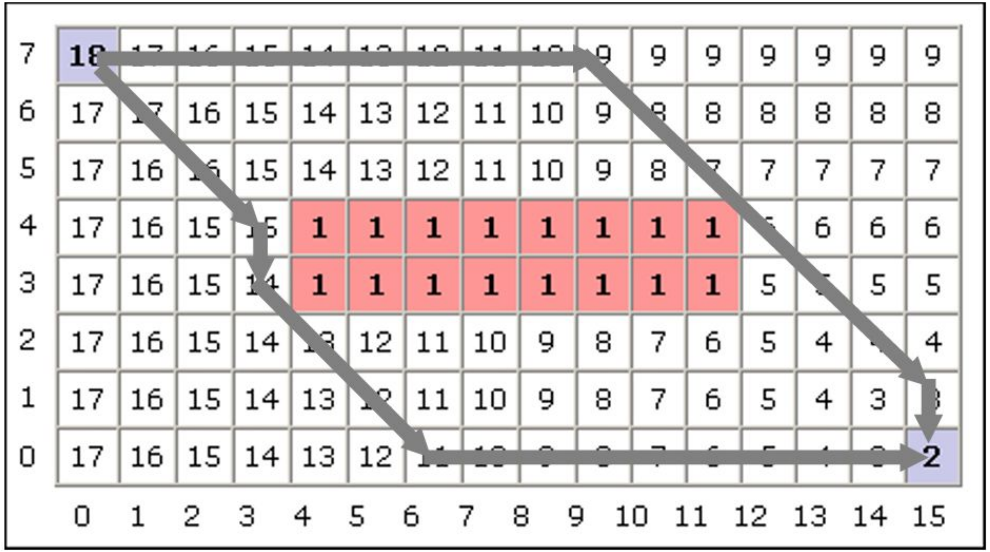
Outline for today

- Motion planning algorithms
  - Graph search
  - Sampling-based

Wavefront planner

- Common algorithm used to determine the shortest path between two points
- To find the shortest path, always move toward a cell with a lower number

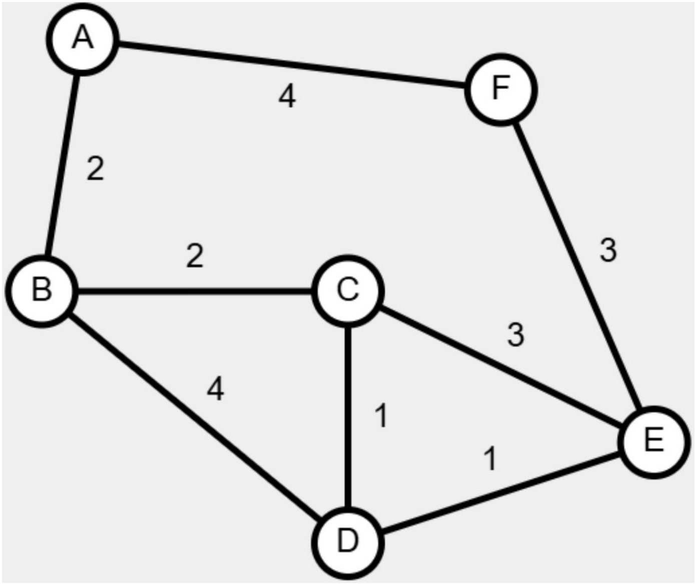
Two possible shortest paths shown



Dijkstra's algorithm

Algorithm for finding the shortest paths between nodes in a weighted graph

- Assumes non-negative edge weights



Example

```
function Dijkstra(Graph, source):

    for each vertex v in Graph.Vertices:
        dist[v] ← INFINITY          // Unknown distance from source to v
        prev[v] ← UNDEFINED        // Predecessor of v
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with minimum dist[u]
        Q.remove(u)

        for each edge (u, v) in Graph:
            alt ← dist[u] + Graph.Distance(u,v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

Wikipedia

Dijkstra's algorithm

To find the shortest path, perform reverse iteration:

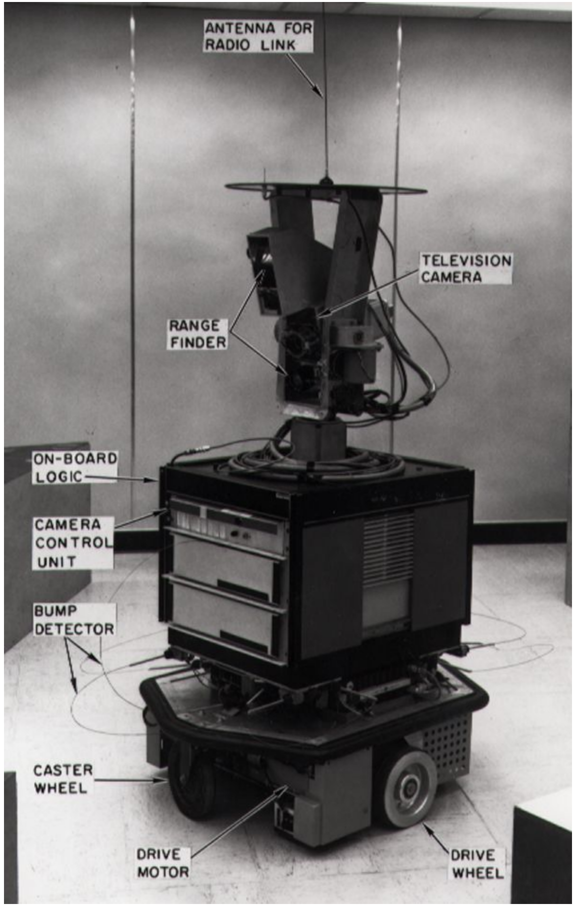
```
S ← empty sequence
u ← target
if prev[u] is defined or u = source:      // Proceed if the vertex is reachable
    while u is defined:                   // Construct shortest path with stack S
        S.push(u)                         // Push the vertex onto the stack
        u ← prev[u]                       // Traverse from target to source
return S
```

Search algorithms

- Uninformed search
  - Has no information about goal beyond identifying when reached
  - Blind search: BFS, DFS, etc.
- Informed search
  - Use problem-specific knowledge
  - More efficient
  - Heuristic search: A*, etc.

A* search

- Created as part of the Shakey project
- Finds the shortest path from source to goal
- Selects path that minimizes: $f(n) = g(n) + h(n)$
- *h(n)*: heuristic function that estimates the cost of the cheapest path from *n* to the goal

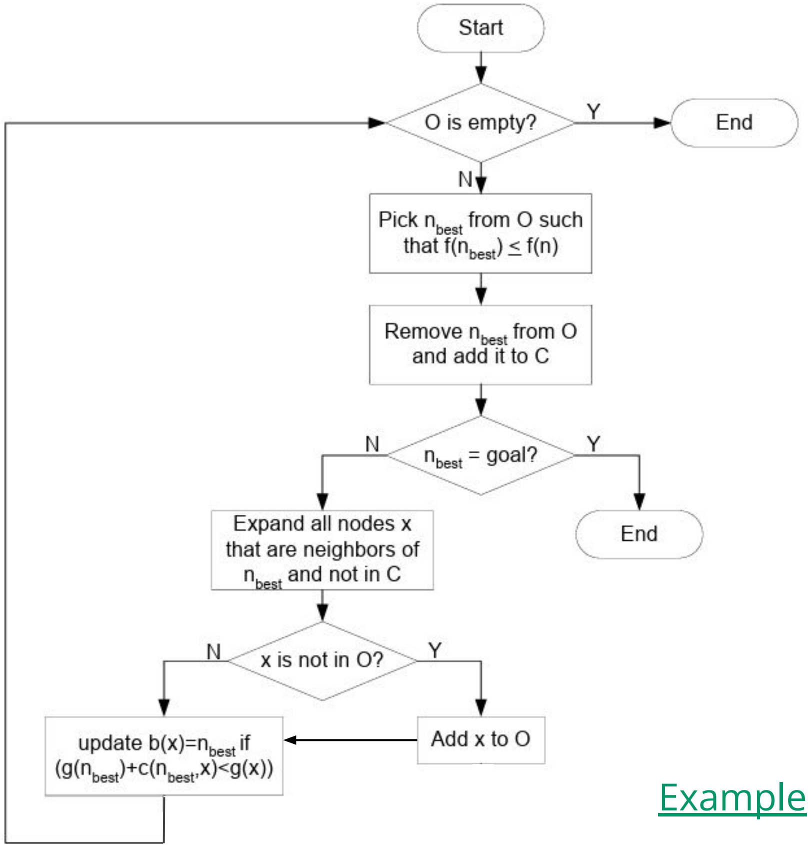


Wikipedia

Heuristic functions

- Approximate true values
- Problem-specific
- Admissible: never overestimates the cost of reaching the goal
  - With an admissible heuristic, A* is guaranteed to return an optimal solution
- Consistent: $h(n) \leq c(n,v) + h(v)$; $h(goal) = 0$
  - With a consistent heuristic, A* is guaranteed to find an optimal path without processing any node more than once

A* search with consistent heuristic



Example

The search requires 2 lists to store information about nodes

1) **Open list (O)** stores nodes for expansions
2) **Closed list (C)** stores nodes which we have explored

$c(n_1,n_2)$: length of edge between n_1 and n_2

$b(n_1)=n_2$: backpointer of n_1 to its predecessor n_2

Outline for today

- Motion planning algorithms
  - Graph search
  - Sampling-based

Motion planning algorithms

	Graph search	Sampling-based
Completeness	Complete	Probabilistically complete
Path optimality	Optimal	Depends
Memory usage	High (store entire map)	Low (store sampled nodes)
Best for	Low-dimensional, 2D grids	High-dimensional, complex spaces