

# Search Problem Formulation and Uninformed Search

Alice Gao

Lecture 3

Readings: RN 3.2, 3.3, 3.4.1, 3.4.3.

# Outline

Learning Goals

Applications of Search

Formulating a Search Problem

Uninformed Search Algorithms

- Depth-first search

- Breadth-first search

- Iterative-deepening search

- Lowest-cost-first search

Revisiting the Learning Goals

# Learning goals

By the end of the lecture, you should be able to

- ▶ Formulate a real world problem as a search problem.
- ▶ Trace the execution of and implement uninformed search algorithms (Breadth-first search, Depth-first search, Iterative-deepening search, and Lowest-cost-first search)
- ▶ Given an uninformed search algorithm, explain its space complexity, time complexity, and whether it has any guarantees on the quality of the solution found.
- ▶ Given a scenario, explain whether and why it is appropriate to use an uninformed algorithm.

Learning Goals

Applications of Search

Formulating a Search Problem

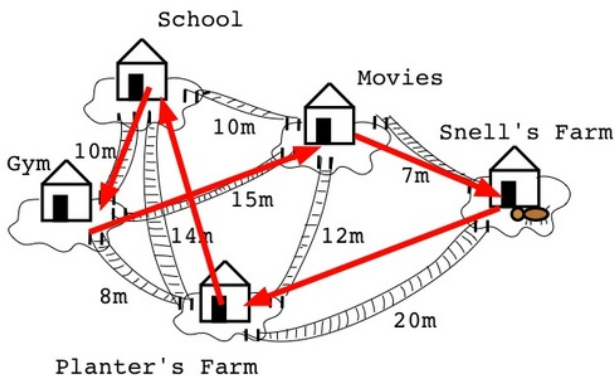
Uninformed Search Algorithms

Revisiting the Learning Goals

# Example: Transportation and Logistics

Applications of TSP: School bus routes, Service calls, Parcel pickups, DNA sequencing. More at <https://bit.ly/2i9JdIV>

## Traveling Salesperson Problem



## Example: Propositional Satisfiability

Given a propositional formula, is there a way to assign truth values to the variables to make the formula true?

$$((((a \wedge b) \vee c) \wedge d) \vee (\neg e))$$

One application: FCC spectrum auction

Check out papers by [Neil Newman](#) and Kevin Leyton-Brown.  
For example, [check this one out](#).

## Example: 8-puzzle

Initial State

5	3	
8	7	6
2	4	1

Goal State

1	2	3
4	5	6
7	8	

## Example: Hua Rong Pass Puzzle



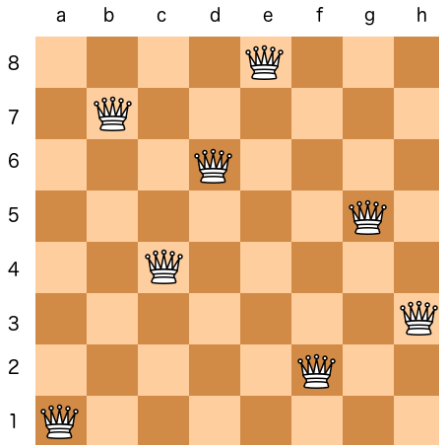


# Example: River Crossing Puzzle



<https://www.lounge.se/MissionariesCannibalsProblem>

## Example: $N$ -Queens Problem



The  $n$ -queens problem: Place  $n$  queens on an  $n \times n$  board so that no pair of queens attacks each other.

[http://yue-guo.com/wp-content/uploads/2019/02/N\\_queen.png](http://yue-guo.com/wp-content/uploads/2019/02/N_queen.png)

Learning Goals

Applications of Search

Formulating a Search Problem

Uninformed Search Algorithms

Revisiting the Learning Goals

# Why search?

We are facing a difficult problem.

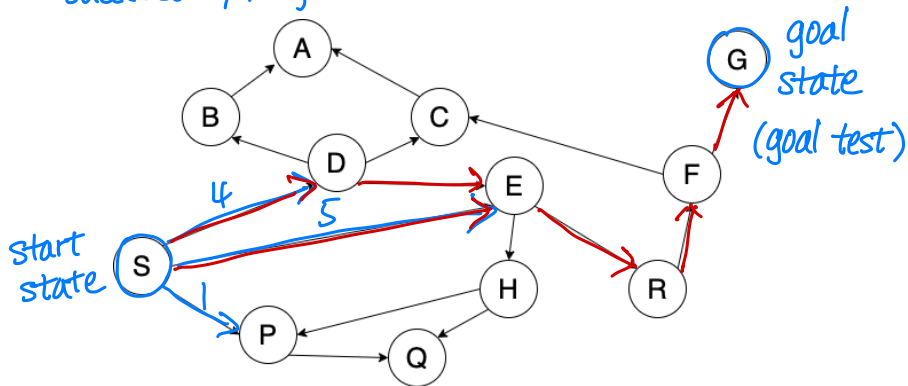
- ▶ Given a description that helps us recognize a solution
- ▶ Not given an algorithm to solve the problem

We have to search for a solution!

# Graph Searching

states  $\leftarrow$  nodes

successors / neighbours  $\leftarrow$  directed arcs



Costs  $\geq 0$

# Graph Search Formally

## Definition (Search Problem)

A **search problem** is defined by

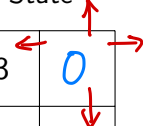
- ▶ A set of **states**
- ▶ A **start state**
- ▶ **Goal states** or a **goal test**
  - ▶ a boolean function which tells us whether a given state is a goal state
- ▶ A **successor (neighbour) function**
  - ▶ an action which takes us from one state to other states
- ▶ (Optionally) a **cost** associated with each action

A solution to this problem is a path from the start state to a goal state (optionally with the smallest total cost).

## Example: 8-Puzzle

Initial State

5	3	0
8	7	6
2	4	1



530, 876, 241

Goal State

1	2	3
4	5	6
7	8	0

123, 456, 780

## Formulating 8-Puzzle as a Search Problem

State :  $x_{00} x_{01} x_{02}$ ,  $x_{10} x_{11} x_{12}$ ,  $x_{20} x_{21} x_{22}$

$x_{ij}$  is the number in row  $i$  and column  $j$

$i, j \in \{0, 1, 2\}$ .  $x_{ij} \in \{0, \dots, 8\}$

$x_{ij} = 0$  if and only if the square is empty.

initial state: 530, 876, 241.

goal state: 123, 456, 780.

successor function: the states generated by moving the empty square left, right, up and/or down, whenever possible.

cost function: each move has a cost of 1.



## CQ: The successor function

**CQ:** Which of the following is a successor of 530, 876, 241?

(A) 350, 876, 241

(B) 536, 870, 241

(C) 537, 806, 241

(D) 538, 076, 241

530

876

241

503

876

241

536

870

241

# Why does the formulation matter?

Q: How does the state definition affect how we implement the successor function?

- ▶ The state definition can affect the amount of information needed to encode each state, and how difficult it is to implement the successor function.

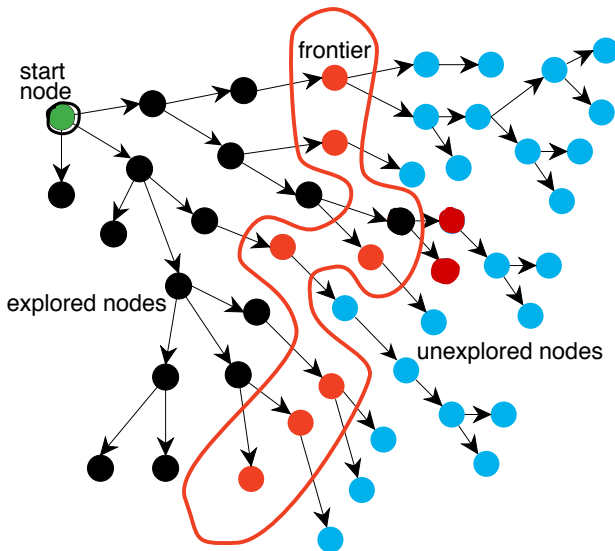
A state is defined by 8 coordinates.

$(x_i, y_i)$  are the coordinates for the tile  $i$

- ▶ The successor function can affect where  $i \in \{1, \dots, 8\}$ .  
the size and structure of the search graph.

Missionaries and Cannibals problem.

# Generating the Search Tree



# Searching for a Solution

- ▶ Construct the search tree as we explore paths incrementally from the start node.
- ▶ Maintain a frontier of paths from the start node.
- ▶ Frontier contains all the leaf nodes available for expansion.
- ▶ Expanding a node: removes it from the frontier, generating all of its neighbours and adding the neighbours to the frontier.

# Graph Search Algorithm

---

## Algorithm 1 A Generic Search Algorithm

---

```
1: procedure SEARCH(Graph, Start node  $s$ , Goal test  $goal(n)$ )
2:   frontier :=  $\{\langle s \rangle\}$ ; → defines our search strategy.
3:   while frontier is not empty do
4:     select and remove path  $\langle n_0, \dots, n_k \rangle$  from frontier;
5:     if  $goal(n_k)$  then
6:       return  $\langle n_0, \dots, n_k \rangle$ ;
7:     for every neighbour  $n$  of  $n_k$  do
8:       add  $\langle n_0, \dots, n_k, n \rangle$  to frontier;
9:   return no solution
```

Learning Goals

Applications of Search

Formulating a Search Problem

Uninformed Search Algorithms

- Depth-first search

- Breadth-first search

- Iterative-deepening search

- Lowest-cost-first search

Revisiting the Learning Goals

# Uninformed Search Algorithms

- ▶ Depth-first search
- ▶ Breadth-first search
- ▶ Iterative-deepening search
- ▶ Lowest-cost-first search

Learning Goals

Applications of Search

Formulating a Search Problem

Uninformed Search Algorithms

- Depth-first search

- Breadth-first search

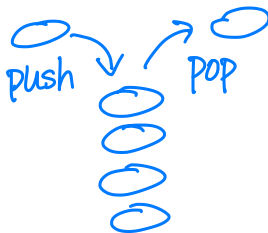
- Iterative-deepening search

- Lowest-cost-first search

Revisiting the Learning Goals



# Depth-First Search

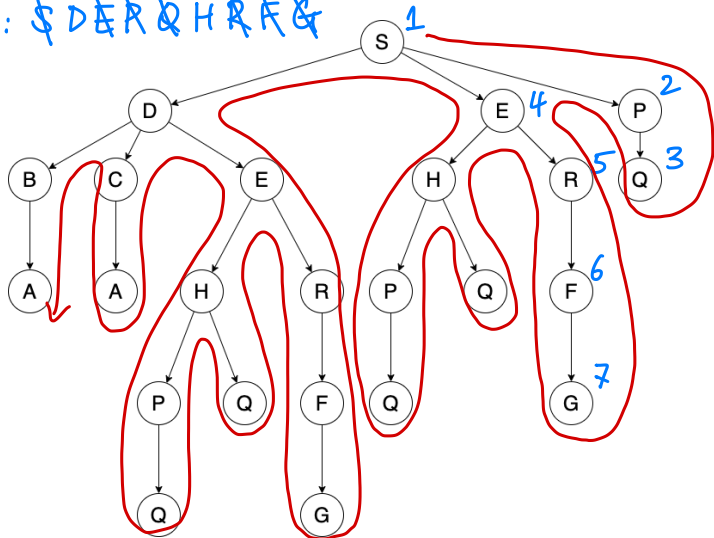


- ▶ Treats the frontier as a stack (LIFO).
- ▶ Expands the last/most recent node added to the frontier.

# Trace DFS on this search tree

We add nodes to the frontier in alphabetical order.

frontier: ~~S~~ ~~D~~ ~~E~~ ~~R~~ ~~Q~~ ~~H~~ ~~R~~ ~~F~~ ~~G~~

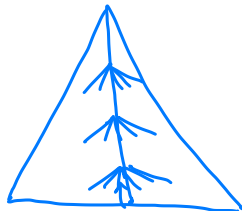


# Properties of DFS

(size of frontier)

- Space complexity?

linear in  $m \leftarrow O(bm)$



↑ maintains a path.  
at each level,  
remember all  
siblings/  
alternatives.  
↓

- Time complexity? worst case: visit all the states

$O(b^m) \rightarrow$  exponential in  $m$ .

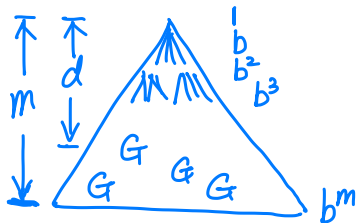
- Guaranteed to find a solution if one exists?

No. may go down an infinite path forever.

Useful quantities:

**b**: branching factor; **m**: maximum depth;

**d**: depth of the shallowest goal node



# Search w/ Cycle Pruning

---

## Algorithm 2 Search w/ Cycle Pruning

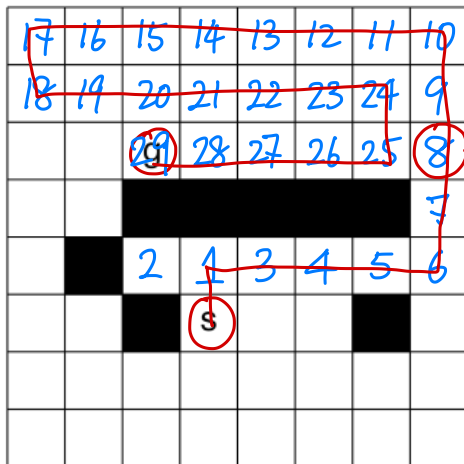
---

```
1: procedure SEARCH(Graph, Start node  $s$ , Goal test  $goal(n)$ )
2:   frontier :=  $\{\langle s \rangle\}$ ;
3:   while frontier is not empty do
4:     select and remove path  $\langle n_0, \dots, n_k \rangle$  from frontier;
5:     if  $goal(n_k)$  then
6:       return  $\langle n_0, \dots, n_k \rangle$ ;
7:     for every neighbour  $n$  of  $n_k$  do
8:       if  $n \notin \langle n_0, \dots, n_k \rangle$  then
9:         add  $\langle n_0, \dots, n_k, n \rangle$  to frontier;
10:  return no solution
```

---

## Try DFS on this grid search problem

- ▶ Expand the successors in the order: up, left, right, and down.
- ▶ Number the nodes as they are removed from the frontier.
- ▶ Use cycle pruning.



→ blind search  
we don't know  
g is on the  
left. !!

Learning Goals

Applications of Search

Formulating a Search Problem

Uninformed Search Algorithms

- Depth-first search

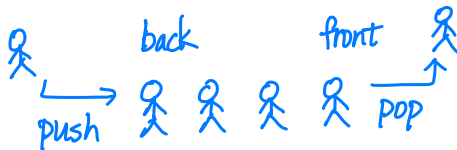
- Breadth-first search

- Iterative-deepening search

- Lowest-cost-first search

Revisiting the Learning Goals

# Breadth-First Search

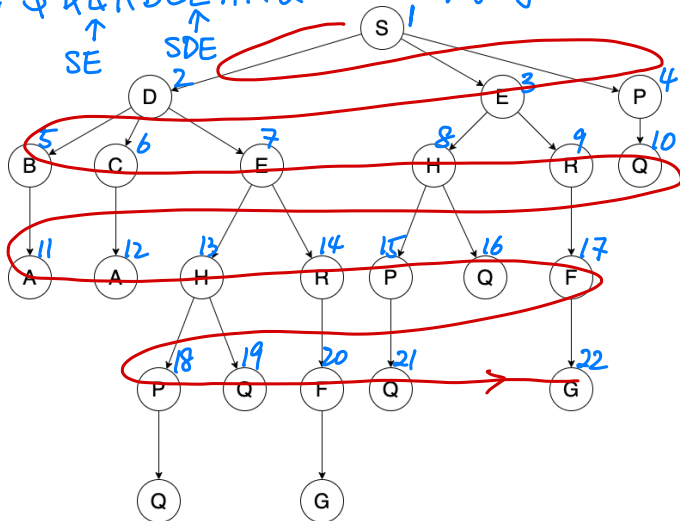


- ▶ Treats the frontier as a queue (FIFO).
- ▶ Expands the first/oldest node added to the frontier.

# Trace BFS on this search tree

We add nodes to the frontier in alphabetical order.

frontier: S D E R B C E H R Q ..... keep going





# Properties of BFS

- ▶ Space complexity? (size of the frontier)

$O(b^d)$  size of the level containing the shallowest goal node.

- ▶ Time complexity?

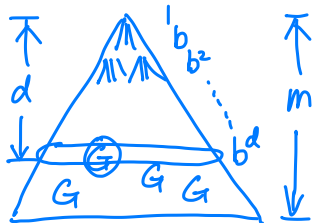
$O(b^m)$  worst case: visit all nodes in the tree.

- ▶ Guaranteed to find a solution if one exists?

Yes.

- ▶ Guaranteed to find the shallowest goal node?

Yes.



Useful quantities:

**b**: branching factor; **m**: maximum depth;

**d**: depth of the shallowest goal node

# Search w/ Multi-Path Pruning

---

**Algorithm 3** Search w/ Multi-Path Pruning

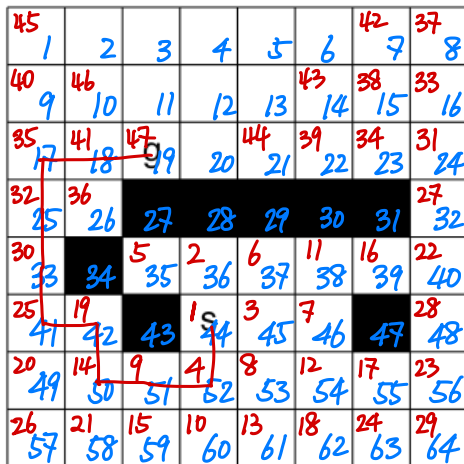
---

```
1: procedure SEARCH(Graph, Start node  $s$ , Goal test  $goal(n)$ )
2:   frontier :=  $\{\langle s \rangle\}$ ;
3:   explored :=  $\{\}$ ;
4:   while frontier is not empty do
5:     select and remove path  $\langle n_0, \dots, n_k \rangle$  from frontier;
6:     if  $n_k \notin$  explored then
7:        $\rightarrow$  add  $n_k$  to explored
8:       if  $goal(n_k)$  then
9:         return  $\langle n_0, \dots, n_k \rangle$ ;
10:      for every neighbour  $n$  of  $n_k$  do
11:        add  $\langle n_0, \dots, n_k, n \rangle$  to frontier;
12:   return no solution
```

---

## Try BFS on this grid search problem

- ▶ Expand the successors in the order: up, left, right, and down.
- ▶ Number the nodes as they are removed from the frontier.
- ▶ Use multi-path pruning. *44, 52, 51, 50, 42, 41, 33, 25, 17, 18, 19*



found the  
shortest path  
!!

# Try BFS on this grid search problem

- Expand the successors in the order: ~~up, left, right, and down.~~ *down, left, up, right*
- Number the nodes as they are removed from the frontier.
- Use multi-path pruning.

<i>42</i> 1	2	3	4	5	6	7	<i>41</i> 8
<i>37</i> 9	<i>43</i> 10	11	12	13	14	<i>40</i> 15	<i>36</i> 16
<i>33</i> 17	<i>38</i> 18	<i>44</i> 19	20	21	<i>39</i> 22	<i>35</i> 23	<i>32</i> 24
<i>31</i> 25	<i>34</i> 26	27	28	29	30	31	<i>30</i> 32
<i>28</i> 33	34	8	3	9	15	21	26
<i>24</i> 41	19	42	43	<i>1</i> 44	4	10	29
18	13	6	2	7	14	20	25
<i>22</i> 57	16	11	<i>5</i> 60	12	17	<i>23</i> 63	<i>27</i> 64

## Try BFS on this grid search problem

- ▶ Expand the successors in the order: up, left, right, and down.
- ▶ Number the nodes as they are removed from the frontier.
- ▶ Use multi-path pruning.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

# Comparing BFS and DFS

Consider the scenarios below.

Which of BFS and DFS would you choose? Why?

1. Memory is limited.
2. All solutions are deep in the tree.
3. The search graph contains cycles.
4. The branching factor is large/infinite.
5. We must find the shallowest goal node.
6. Some solutions are very shallow.

## CQ: BFS v.s. DFS

**CQ 1:** Suppose that memory is very limited.  
Which of BFS and DFS would you choose?

- (A) BFS is a better choice.
- (B) DFS is a better choice.
- (C) Both are good choices.
- (D) Neither is a good choice.

## CQ: BFS v.s. DFS

**CQ 2:** Suppose that all the solutions are deep in the search tree. Which of BFS and DFS would you choose?

- (A) BFS is a better choice.
- (B) DFS is a better choice.
- (C) Both are good choices.
- (D) Neither is a good choice.



## CQ: BFS v.s. DFS

**CQ 3:** Suppose that the search graph contains cycles. Which of BFS and DFS would you choose?

- (A) BFS is a better choice.
- (B) DFS is a better choice.
- (C) Both are good choices.
- (D) Neither is a good choice.

## CQ: BFS v.s. DFS

**CQ 4:** Suppose that the branching factor is large/infinite. Which of BFS and DFS would you choose?

- (A) BFS is a better choice.
- (B) DFS is a better choice.
- (C) Both are good choices.
- (D) Neither is a good choice.

## CQ: BFS v.s. DFS

**CQ 5:** Suppose that we must find the shallowest goal node. Which of BFS and DFS would you choose?

- (A) BFS is a better choice.
- (B) DFS is a better choice.
- (C) Both are good choices.
- (D) Neither is a good choice.

## CQ: BFS v.s. DFS

**CQ 6:** Suppose that all the solutions are very shallow. Which of BFS and DFS would you choose?

- (A) BFS is a better choice.
- (B) DFS is a better choice.
- (C) Both are good choices.
- (D) Neither is a good choice.

Learning Goals

Applications of Search

Formulating a Search Problem

Uninformed Search Algorithms

- Depth-first search

- Breadth-first search

- Iterative-deepening search**

- Lowest-cost-first search

Revisiting the Learning Goals

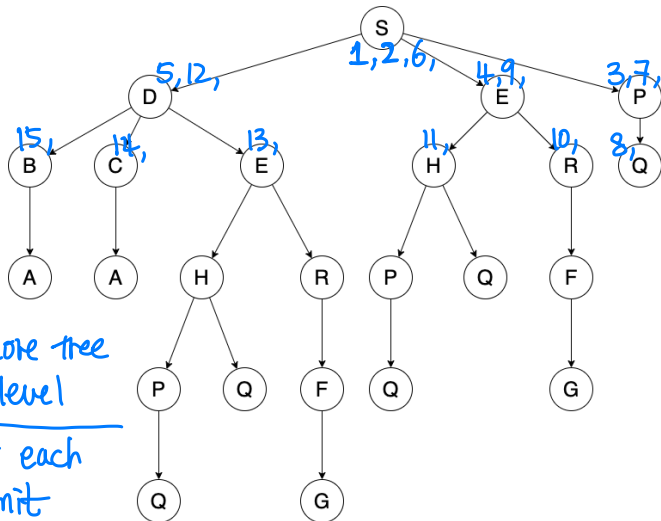
# BFS and DFS — the best of both worlds

Can we have a search algorithm  
that combines the best of BFS and DFS?

BFS	DFS
$O(b^d)$ exponential space	$O(bm)$ linear space
Guaranteed to find a solution if one exists	May get stuck on infinite paths

# Iterative-deepening search *depth limit = 1 & 3*

For every depth limit, *frontier: \$ | \$ D A R | \$ D A R Q H R B A E*  
perform depth-first search until the depth limit is reached.



*d=1*

*d=2*

*d=3*

*d=4*

*d=5*

BFS: explore tree level by level

DFS: for each depth limit

## Properties of IDS

$$\text{BFS: } b^d + b^{d-1} + b^{d-2} + \dots + b + 1 \\ = \boxed{b^d \left( \frac{b}{b-1} \right)} - \frac{1}{b-1}$$

$$\text{IDS: } b^d + 2b^{d-1} + 3b^{d-2} + \dots + db + (d+1) \\ \leq \boxed{b^d \left( \frac{b}{b-1} \right)^2} \quad \frac{b}{b-1}$$

- ▶ Space complexity:

$$O(bd) \text{ (same as DFS)}$$

- ▶ Time complexity:

similar to BFS but a bit worse

$$O\left(b^d \left( \frac{b}{b-1} \right)^2\right) \text{ because of re-computation.}$$

- ▶ Guaranteed to find a solution if one exists?

Yes. (same as BFS).

- ▶ Guaranteed to find the shallowest goal node?

Yes. (same as BFS).



Learning Goals

Applications of Search

Formulating a Search Problem

Uninformed Search Algorithms

- Depth-first search

- Breadth-first search

- Iterative-deepening search

- Lowest-cost-first search

Revisiting the Learning Goals

# What if arcs have different costs?

What if arcs have different costs?

We want an algorithm to find the optimal solution  
— the solution with the lowest total cost.

## Lowest-cost-first search

*a.k.a. Dijkstra's shortest path algorithm.*

The frontier is a priority queue ordered by path cost ( $\text{cost}(n)$ ).  
Expand the path with the lowest total cost.

- Complexity:

*space : exponential      time : exponential*

- Guaranteed to find a solution if one exists?

*Yes.*

- Guaranteed to find the optimal solution?

*Yes.*

# Revisiting the learning goals

By the end of the lecture, you should be able to

- ▶ Formulate a real world problem as a search problem.
- ▶ Trace the execution of and implement uninformed search algorithms (Breadth-first search, Depth-first search, Iterative-deepening search, and Lowest-cost-first search)
- ▶ Given an uninformed search algorithm, explain its space complexity, time complexity, and whether it has any guarantees on the quality of the solution found.
- ▶ Given a scenario, explain whether and why it is appropriate to use an uninformed algorithm.