ME 536

Week 11: Excuse me,
how can I ... ?

How can I:

solve this math problem ?

make an optimal decision ?

beat that guy in chess ?

go there ?

prove that ?

…

Find the croc :

Find → Search

1- You **should** know when you find it

2- Scan

Randomly

Systematically

How can I: get there ?

Get where?

From where?

What are the alternatives?

Let's go: from S to G

A lot to think about:

Cost of moving?

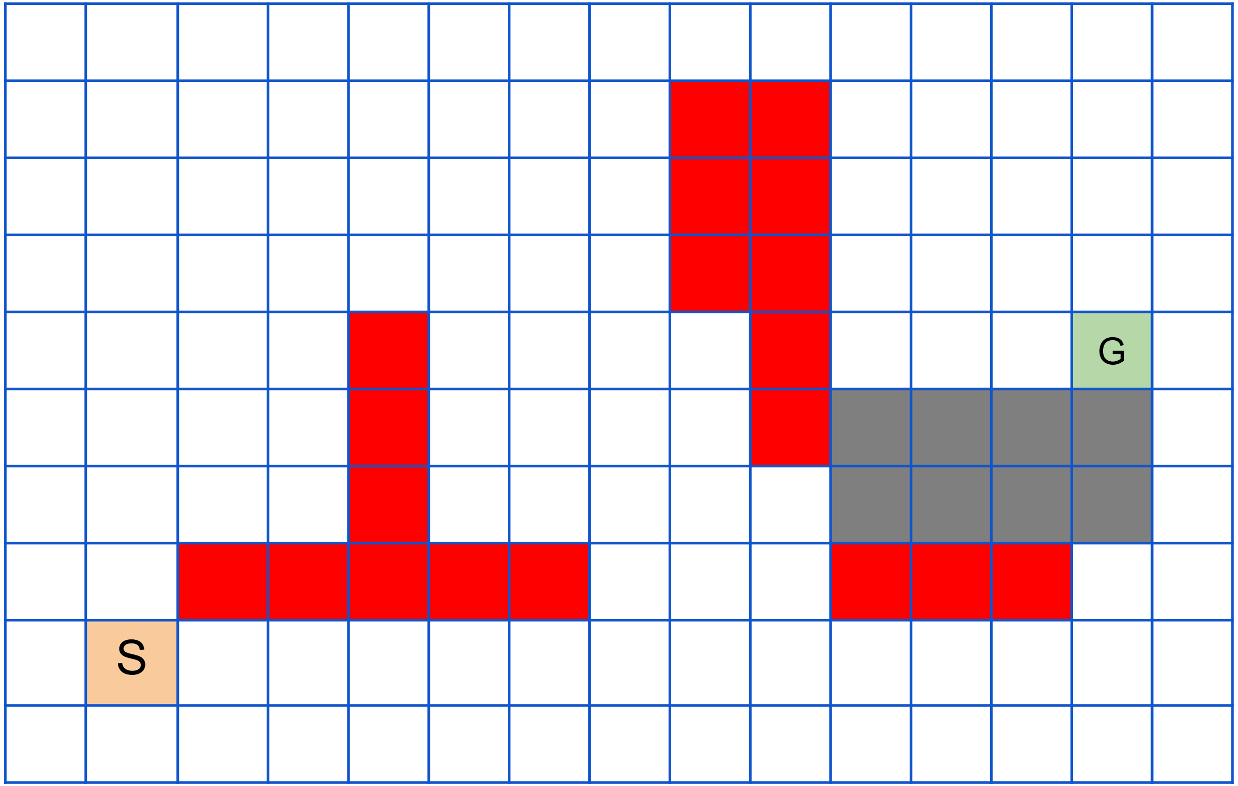
Directions to move?

Direction to choose?

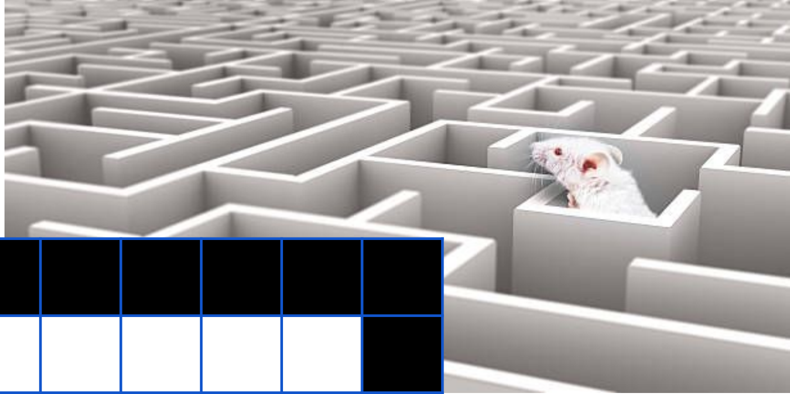
Completely blind

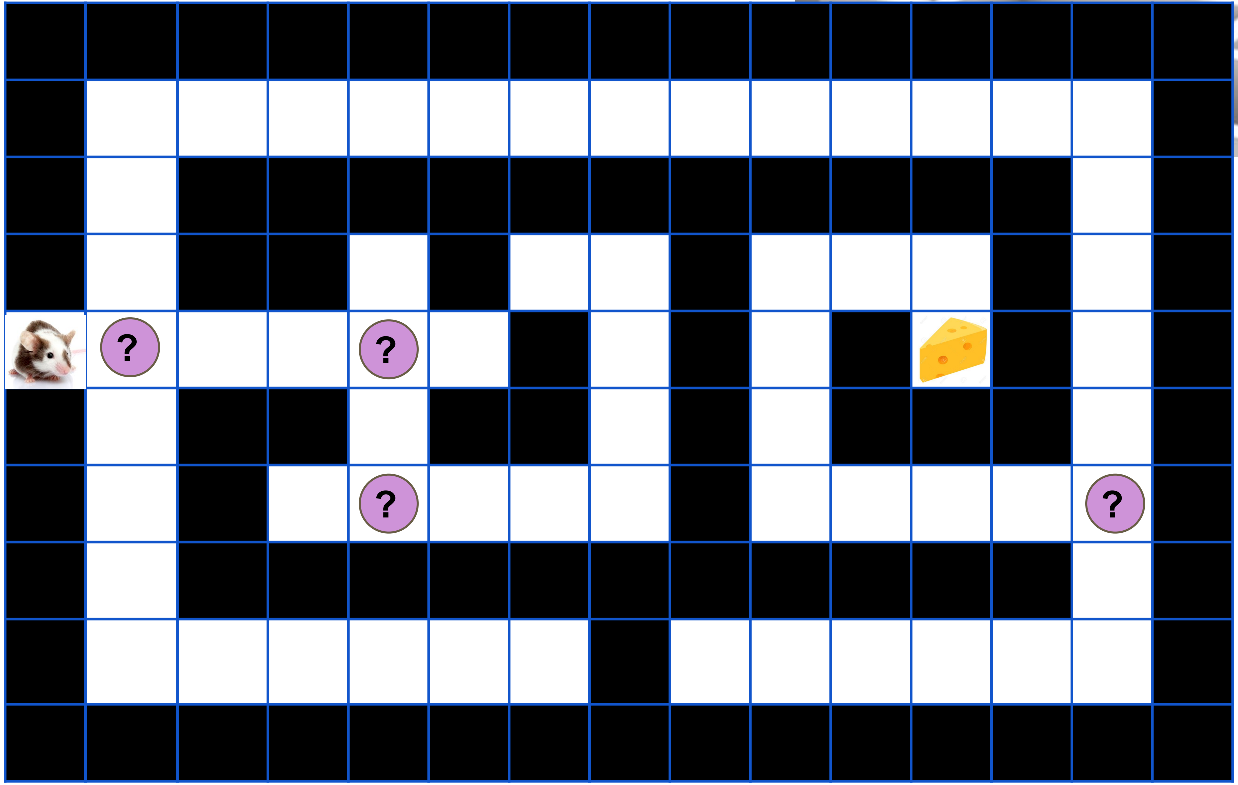
Partially blind

Not blind at all



Let’s go: run for the cheese





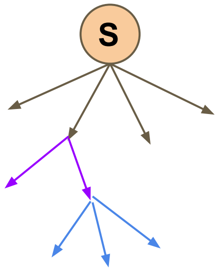
Terminology:

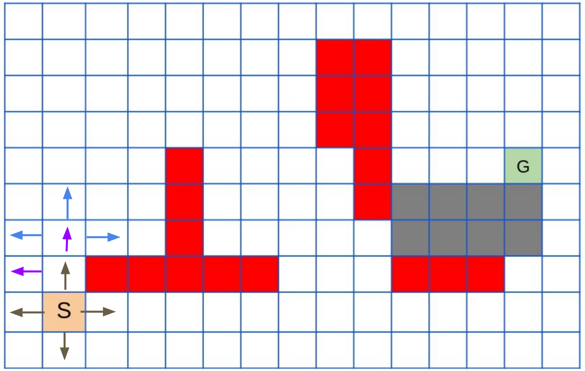
- The **initial state** is the entry point to the **search space**.
- An **operator** (**successor function** $S(\ )$) returns the set of **reachable states** from state x by a single action given x.
  - $S(\ )$ should avoid repeated and unreachable states
- A **path** is a sequence of actions leading *from one state to another*.
- A **solution** is a **path** between the *initial and goal states*.

Terminology:

- A **path cost** (*online cost*) function is a function that assigns a cost to a path
- **Search Cost** (*offline cost*) is defined with time and memory required to find a solution / to make a goal reaching plan.
- **Total cost** of the search is the sum of the path cost and search costs.

Search Strategy





- Determines *which states* should be **expanded** and **checked *next***.
- Generally ends up building a so called: **search tree**.
- The root of the tree is at the *initial state*.

Search Strategy: How good is it?

Can be evaluated based on:

- **Completeness**: is the strategy guaranteed to find a solution when there is one?
- **Time complexity**: how long does it take to find a solution?
- **Space complexity**: how much memory does it need to perform the search?
- **Optimality**: does the strategy find the highest quality solution when there are several alternative solutions?

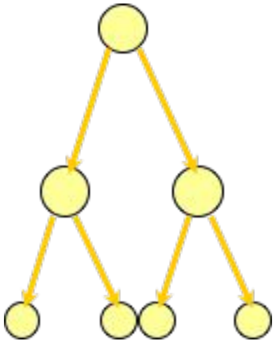
Search Where:

List

11
54
32
22
27
99
19
...

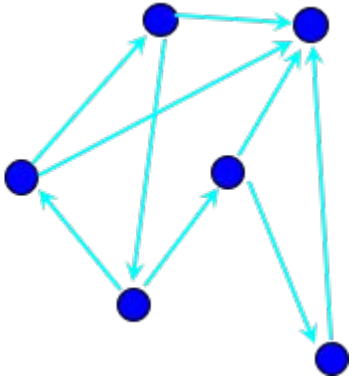
- Is the list sorted?

Tree



- Branching factor b
- b= 2 is a binary tree

Graph



Informed vs Uninformed Search: what is the catch?

So cliche: *number guessing game*

Guess the number I have in my mind that is between 0-100

Uninformed or
a.k.a **BLIND** search

1
NO
2
NO
3
NO
4
NO
...

50
Higher
75
Lower
62
Higher
68
Lower
...

Informed search

Uninformed Search Strategies

- Only binary evaluation is possible after every action
- Distinguished by the order in which nodes are expanded:
  - Breadth-first
  - Depth-first
  - Depth-limited
  - Uniform cost
  - Bidirectional

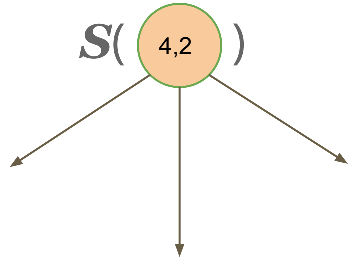
Breadth-First Search: Fish in the shallows first

Initial State S

Goal State G

Unpassable

1,1	1,2	1,3	1,4	1,5
2,1	2,2	2,3	2,4	2,5
3,1	3,2	3,3	3,4	3,5
4,1	4,2	4,3	4,4	4,5
5,1	5,2	5,3	5,4	5,5



<u>Assume:</u>

- movement in **4-neighbors**
- CCW starting from 6 o'clock

Breadth-First Search: Fish in the shallows first

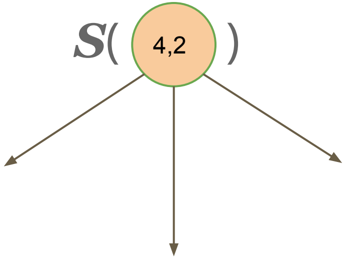
Initial State S

Goal State G

Unpassable

1,1	1,2	1,3	1,4	1,5
2,1	2,2	2,3	2,4	2,5
3,1	3,2	3,3	3,4	3,5
4,1	4,2	4,3	4,4	4,5
5,1	5,2	5,3	5,4	5,5

Fill in the search tree



Assume:

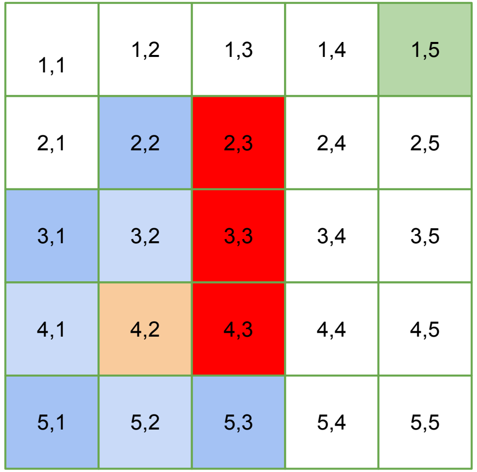
- movement in **4-neighbors**
- CCW starting from 6 o'clock

Breadth-First Search: Fish in the shallows first

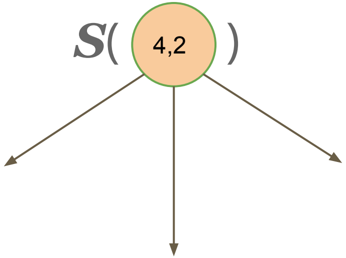
Initial State S

Goal State G

Unpassable



Fill in the search tree

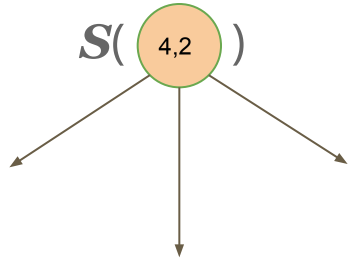


<u>Assume:</u>

- movement in **4-neighbors**
- CCW starting from 6 o'clock

Breadth-First Search: Fish in the shallows first

Fill in the search tree



Initial State S

Goal State G

Unpassable

1,1	1,2	1,3	1,4	1,5
2,1	2,2	2,3	2,4	2,5
3,1	3,2	3,3	3,4	3,5
4,1	4,2	4,3	4,4	4,5
5,1	5,2	5,3	5,4	5,5

<u>Assume:</u>

- movement in **4-neighbors**
- CCW starting from 6 o'clock

Breadth-First Search: Fish in the shallows first

- If there is a solution, *breadth-first search* is **guaranteed to find** it.
- If there are *several solutions*, it will **find the shallowest one**.
- Space and time complexities are the similar since *all leaf nodes must be maintained in the memory*.
- The time complexity is O($\boldsymbol{b^d}$)
- The space complexity is O($\boldsymbol{b^d}$)

Where $\boldsymbol{d}$ is the depth of the search tree

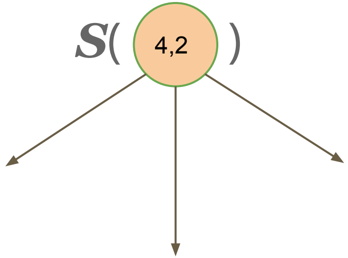
Problem: Cost

Depth-First Search: Check for fish until...

Initial State S

Goal State G

Unpassable



Assume:

- movement in **4-neighbors**
- CCW starting from 6 o'clock

Depth-First Search: Check for fish until...

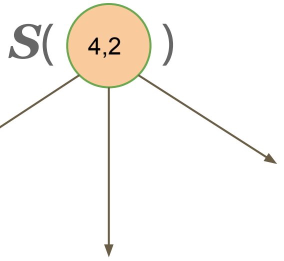
Initial State S

Goal State G

Unpassable

1,1	1,2	1,3	1,4	1,5
2,1	2,2	2,3	2,4	2,5
3,1	3,2	3,3	3,4	3,5
4,1	4,2	4,3	4,4	4,5
5,1	5,2	5,3	5,4	5,5

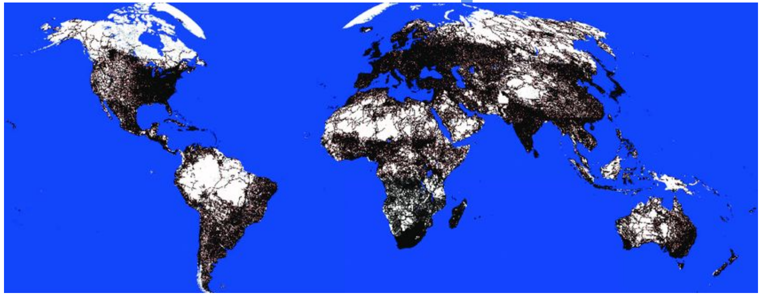
Fill in the search tree



Assume:

- movement in **4-neighbors**
- **CCW starting from 6 o'clock**
    - **Consider different alternatives only to see ?**

Depth-First Search: Fish in the depths first



- **Major drawback**: some problems have *very deep* or even *infinite-depth* search trees.
- Space complexity is also superior to *breadth-first search* since only the current branch has to be kept in memory.
- The time complexity is O($\boldsymbol{b^d}$)
- The space complexity is O($\boldsymbol{bd}$)

Where $\boldsymbol{d}$ is the depth of the search tree

Depth-Limited Search: Quitters may win

- Improve *depth-first search* by ***limiting the depth to search***
    - *Pre-determine a max-depth*
    - *Assume bottom when max-depth reached*
    - *Increase max-depth if no solution found*

Uniform Cost Search: Not all states are created equal

- Uniform cost search modifies the breadth-first search strategy by always expanding the **lowest-cost node on the fringe** rather than the lowest-depth.
- Cost of each node is measured by a function:
  - $g(\boldsymbol{n})$: the path cost of reaching state $\boldsymbol{n}$ from the initial state.
- Path cost is generally assumed to be non-negative: $g(\boldsymbol{n}) \geq 0$
- If costs are constant, $g(\boldsymbol{n})$ gives the path length between $\boldsymbol{n}$ and the initial state.

Uniform Cost Search: Not all states are created equal

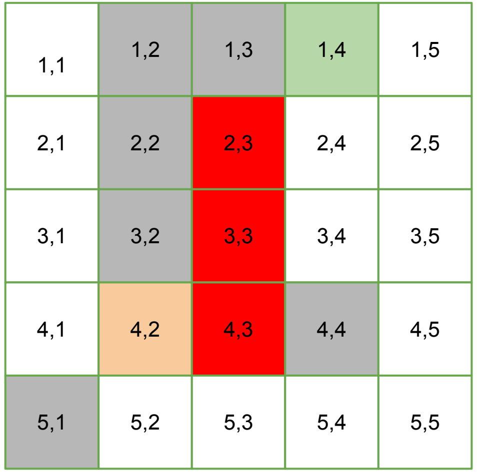
Initial State S

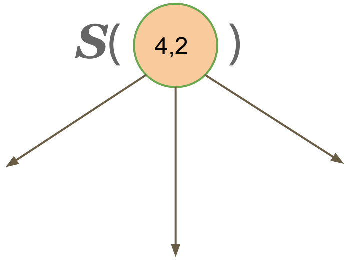
Goal State G

Unpassable

Cost of 1

Cost of 2





<u>Assume:</u>

- movement in **4-neighbors**
- ~~CCW starting from 6 o'clock~~

Uniform Cost Search - Exercise: Draw the search tree

Initial State S

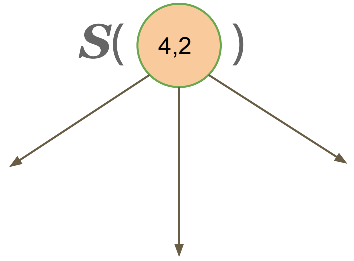
Goal State G

Unpassable

Cost of 1

Cost of 2

1,1	1,2	1,3	1,4	1,5
2,1	2,2	2,3	2,4	2,5
3,1	3,2	3,3	3,4	3,5
4,1	4,2	4,3	4,4	4,5
5,1	5,2	5,3	5,4	5,5



<u>Assume:</u>

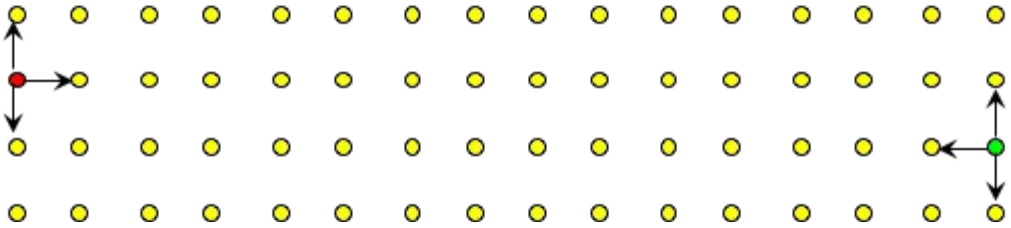
- movement in **4-neighbors**
- Let cost of 2 be 3 or 4

Uniform Cost Search: Not all states are created equal

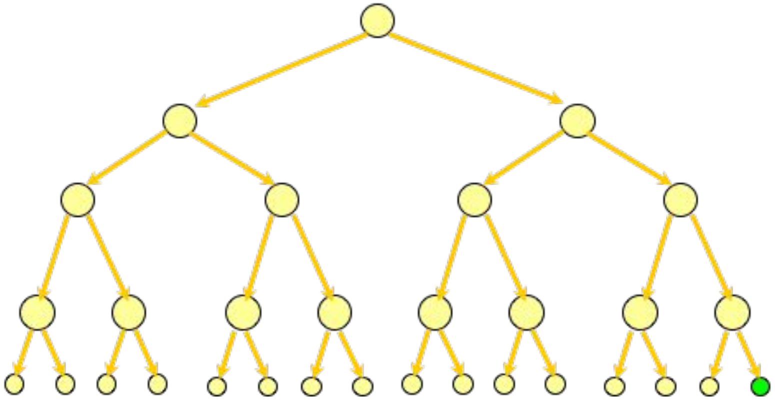
- Finds the **lowest-cost** *or* **cheapest** solution if cost does not decrease:
    - $g(\boldsymbol{S}(\boldsymbol{n})) \geq g(\boldsymbol{n})$

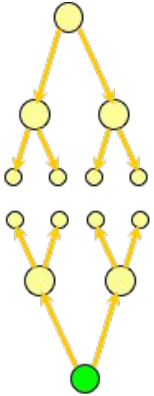
- Finds the **cheapest** solution without exploiting the whole search space / tree

Bi-directional Search:



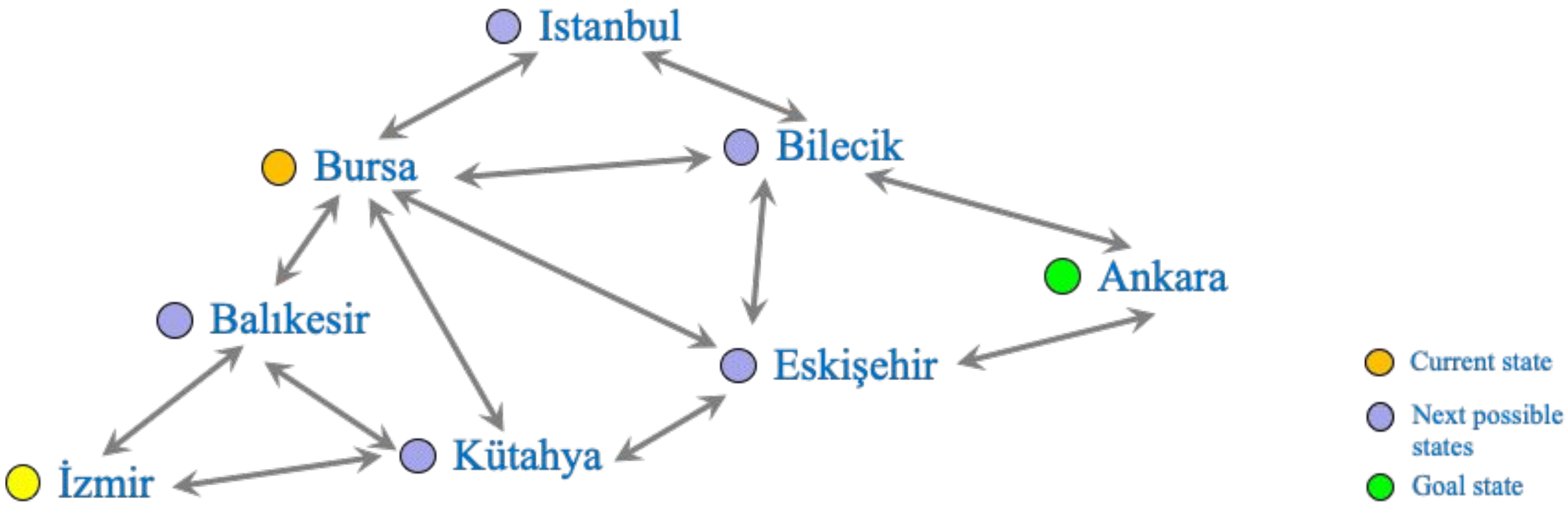
- A way to reduce space complexity
- Solution is found when 2 concurrent search trees meet at one node
- Note that how you can use bi-directional search with *different strategies*





Informed Search: Informed about what ?

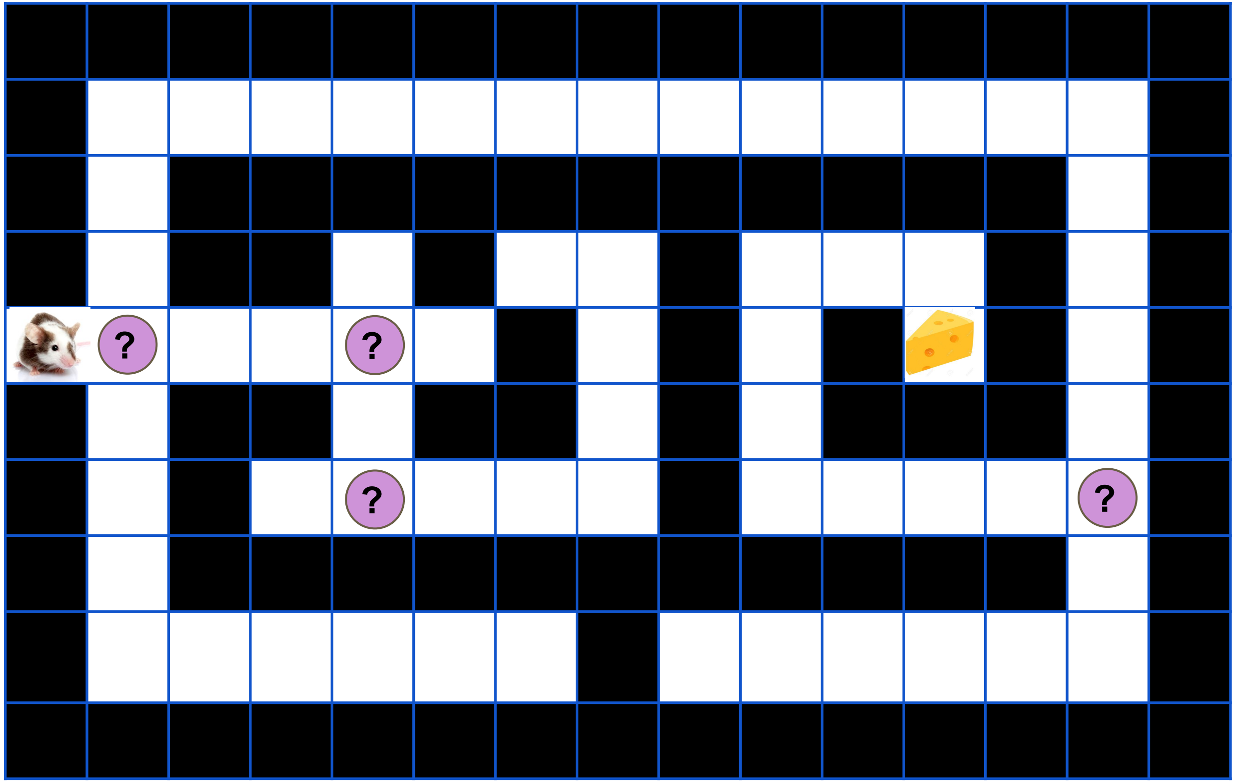
- If **information** or *even a hunch* about **cost to goal** is available
  - *Uninformed search* methods become *inefficient*



Informed Search: Not all *goals* are created equal

- *Assume cost is:*
  - *direct distance to goal*
  - *smell of the goal*

*If you **do NOT know the whole world**, can you have a very accurate cost estimate?*



Heuristic Function: Gut feeling, a hunch, ...

For many problems cost of reaching a goal can be **estimated but cannot be determined** (*otherwise the problem is already solved* :)

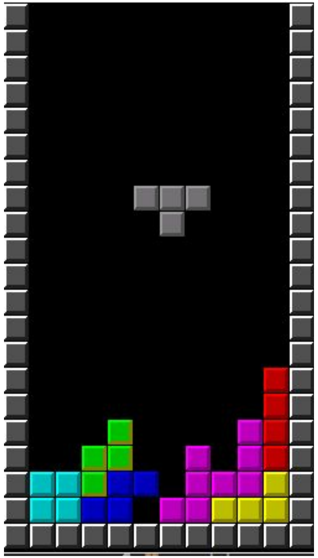
A **heuristic function** estimates the cost of reaching the goal given the current state $\boldsymbol{n}$:

$$\mathrm{h}(\boldsymbol{n}) = \mathrm{f}(\boldsymbol{n}, \textit{goal\_state})$$

Writing a **heuristic function** is not necessarily easy

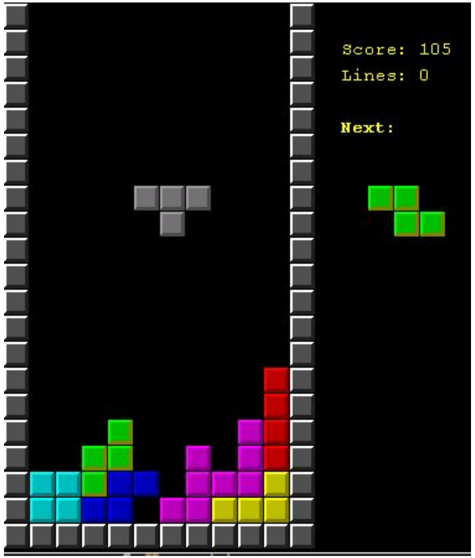
Informed Search: Not all *goals* are created equal

- What is my goal anyway? Short- & long-term?



Informed Search: Not all *goals* are created equal

- What is my goal anyway? Short- & long-term?
- *I am informed* about the next piece then what?



Greedy Search: Best First Search

Best in the sense of your *heuristic function*.

- Expand the *most opportunistic node* next: i.e. the node that is evaluated best based on the *heuristic function*.
- Reminds uniform cost search (most opportunistic vs cheapest)
- Incomplete and not optimal
- very deep or infinite depth search trees are problematic
- good choice of $\mathrm{h}(\boldsymbol{n})$ might substantially improve performance
- bad choice of $\mathrm{h}(\boldsymbol{n})$ might trigger a death-roll

Greedy: with *caution* $\longrightarrow$ *Educated Greed*

Best of Uniform-cost & Greedy Search:

- Uniform-cost search:

  Optimal and complete but not goal oriented

  $g(\boldsymbol{n}) = f(\boldsymbol{n}, \textit{initial\_state})$

- Greedy Search:

  Neither optimal nor complete but goal oriented

  $h(\boldsymbol{n}) = f(\boldsymbol{n}, \textit{goal\_state})$

A* search : Best of two worlds

Cost function:

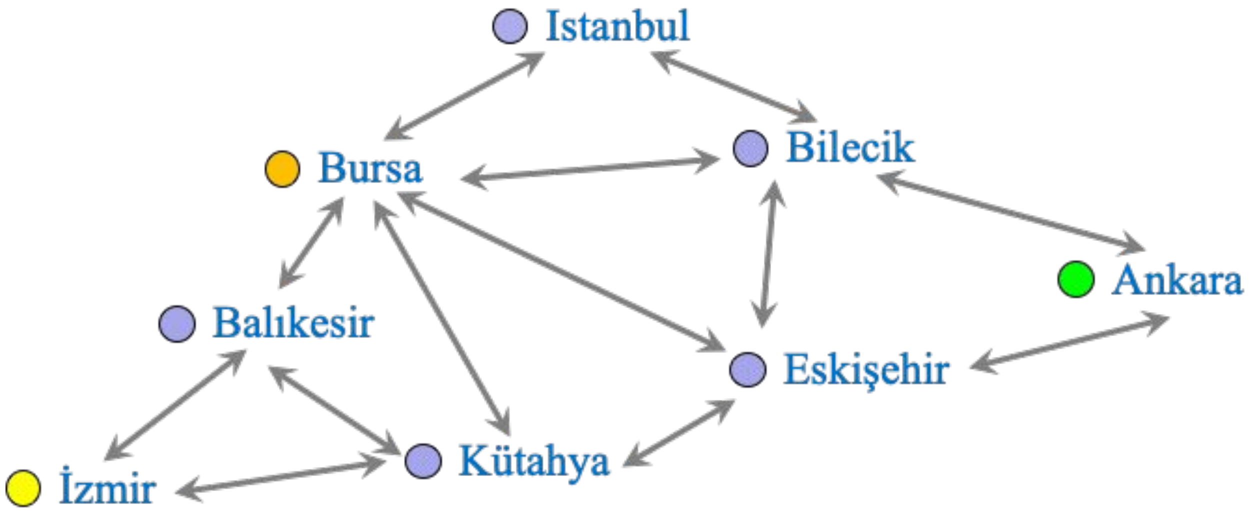
$$f(\boldsymbol{n}) = g(\boldsymbol{n}) + h(\boldsymbol{n})$$

For an admissible heuristic $\mathbf{h}(\boldsymbol{n})$ solution is optimal

$\mathbf{h}(\boldsymbol{n})$ is admissible if it never over-estimates the cost to goal

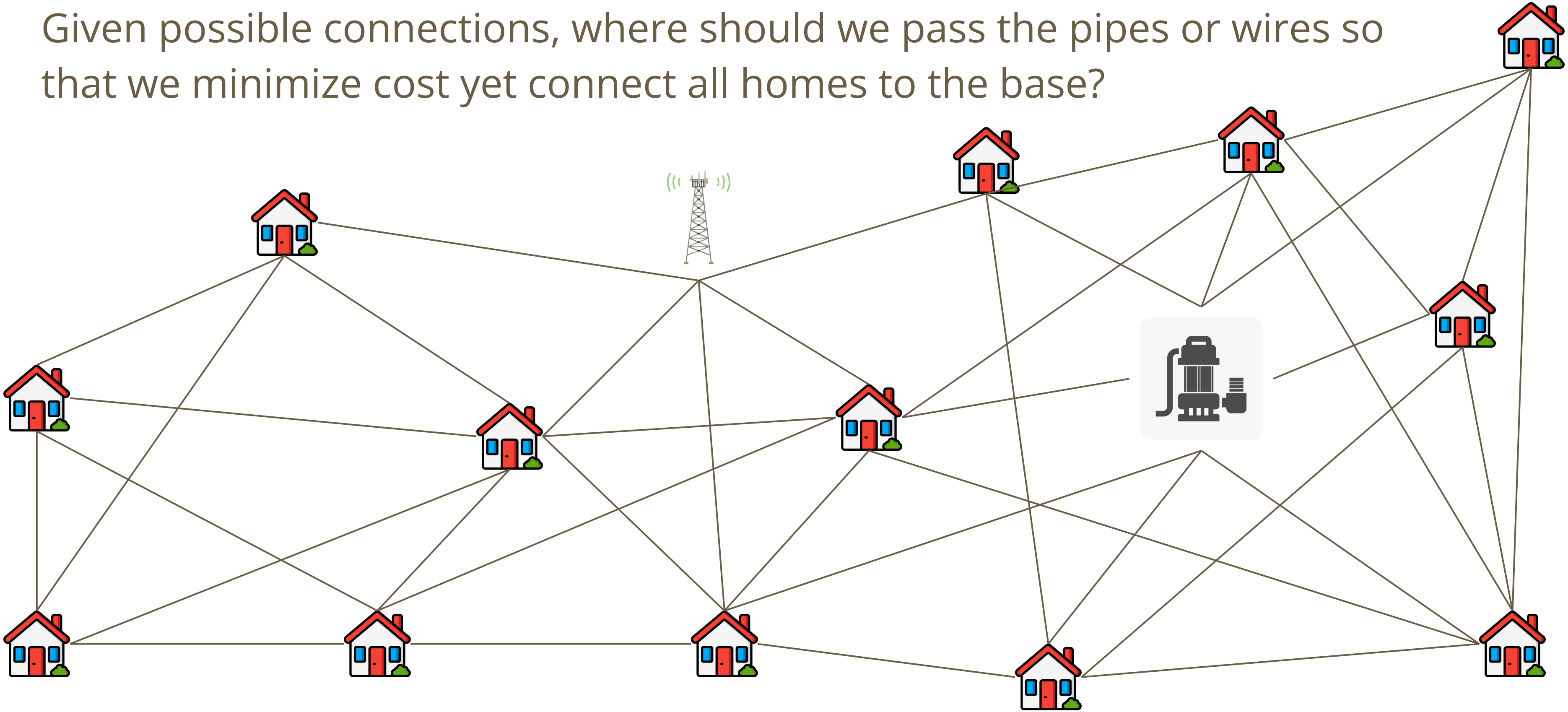
Recall: Not always have a tree!

- How to search over a graph?
- Bursa to Ankara? Which way?
- Trees emerge!



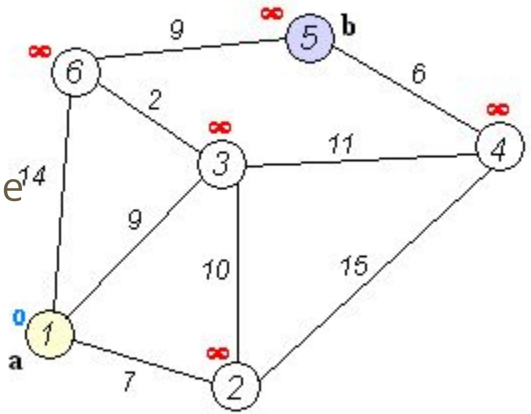
Let's pump it or wire it up? Directed vs Undirected?

Given possible connections, where should we pass the pipes or wires so that we minimize cost yet connect all homes to the base?



Famous *di*-Graph Search Methods: G={V,E,w}

All work on weighted *directed*-graphs



- Dijkstra's- *time complexity* $O(|V| \cdot log|V|+|E|)$
  - Find shortest path to all other nodes from the starting node
  - No negative edge costs are allowed
  - Very much like uniform-cost **search on a tree**
- Bellman-Ford- *time complexity* $O(|V| \cdot |E|)$
  - Find shortest path to all other nodes from the starting node
  - Negative edge costs are welcomed!
  - Negative cycles are NOT!
- Floyd Warshall - *time complexity* $O(V^3)$
  - Find shortest path between any two nodes
  - Negative edge costs are welcomed!

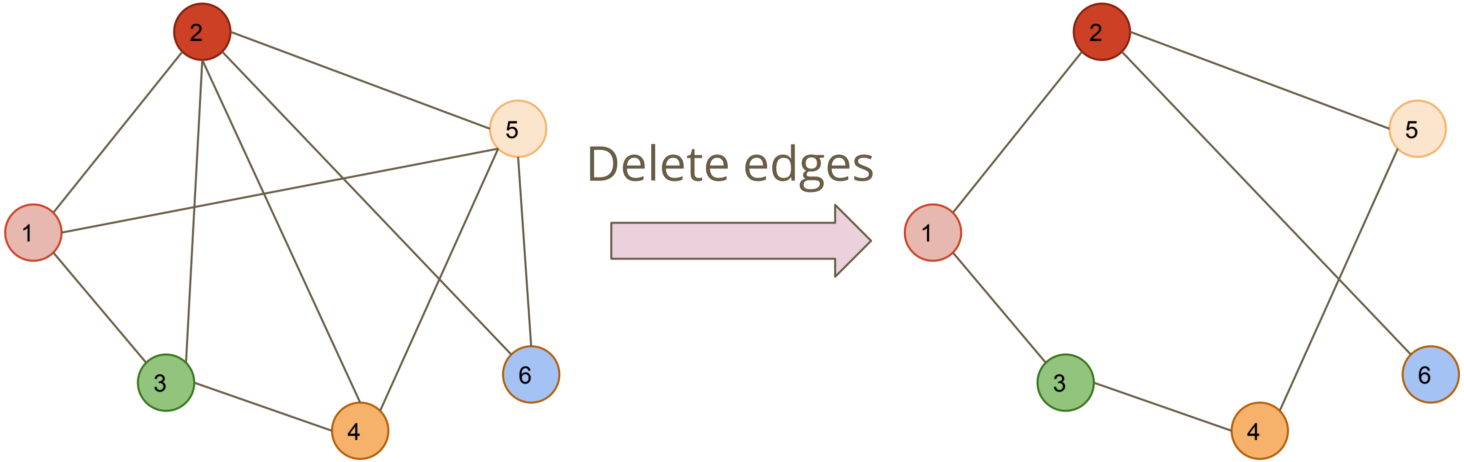
Details left for self study...

Tree from a Graph

Can you extract a tree from a graph $G=\{V,E\}$?

A spanning tree is a sub-graph containing all nodes without cycles

Number of edges in the spanning tree = $|V|$-1



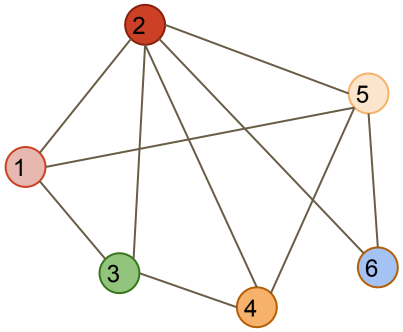
Let’s Delete one more !!!

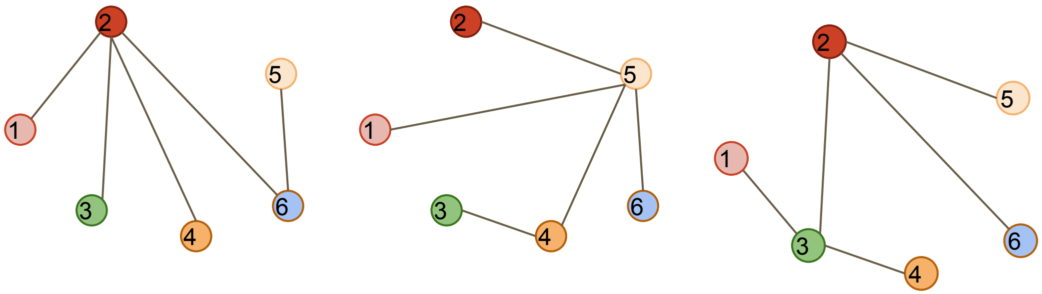
But which one?

Forest from a Graph

Can you extract a tree from a graph G={V,E} ?

Many spanning trees are possible!



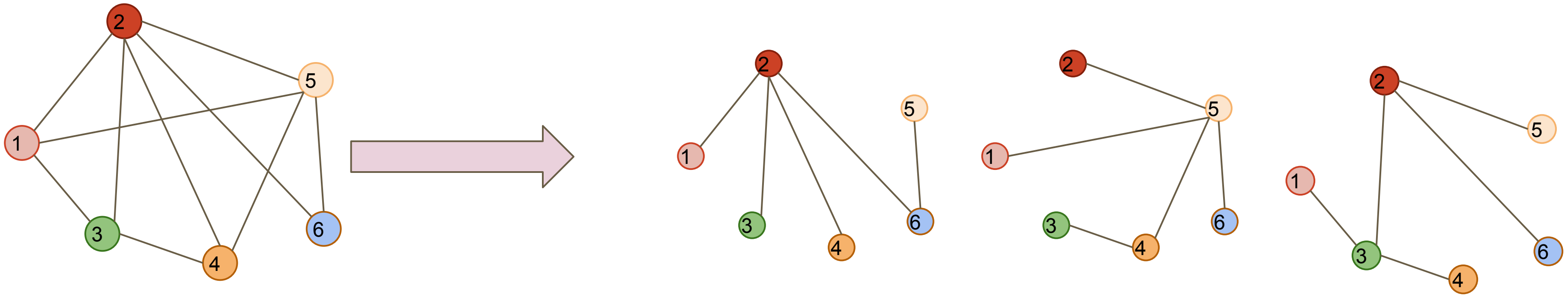


Forest from a Graph : Which one has minimum cost?

Can you extract a tree from a graph $G=\{V,E,w\}$?

Generate all trees and select one?

Or better?



Forest from a Graph: cheapest ones in G={V,E,w}

A minimum (minimal) spanning tree (MST) is a spanning tree with minimal total edge cost

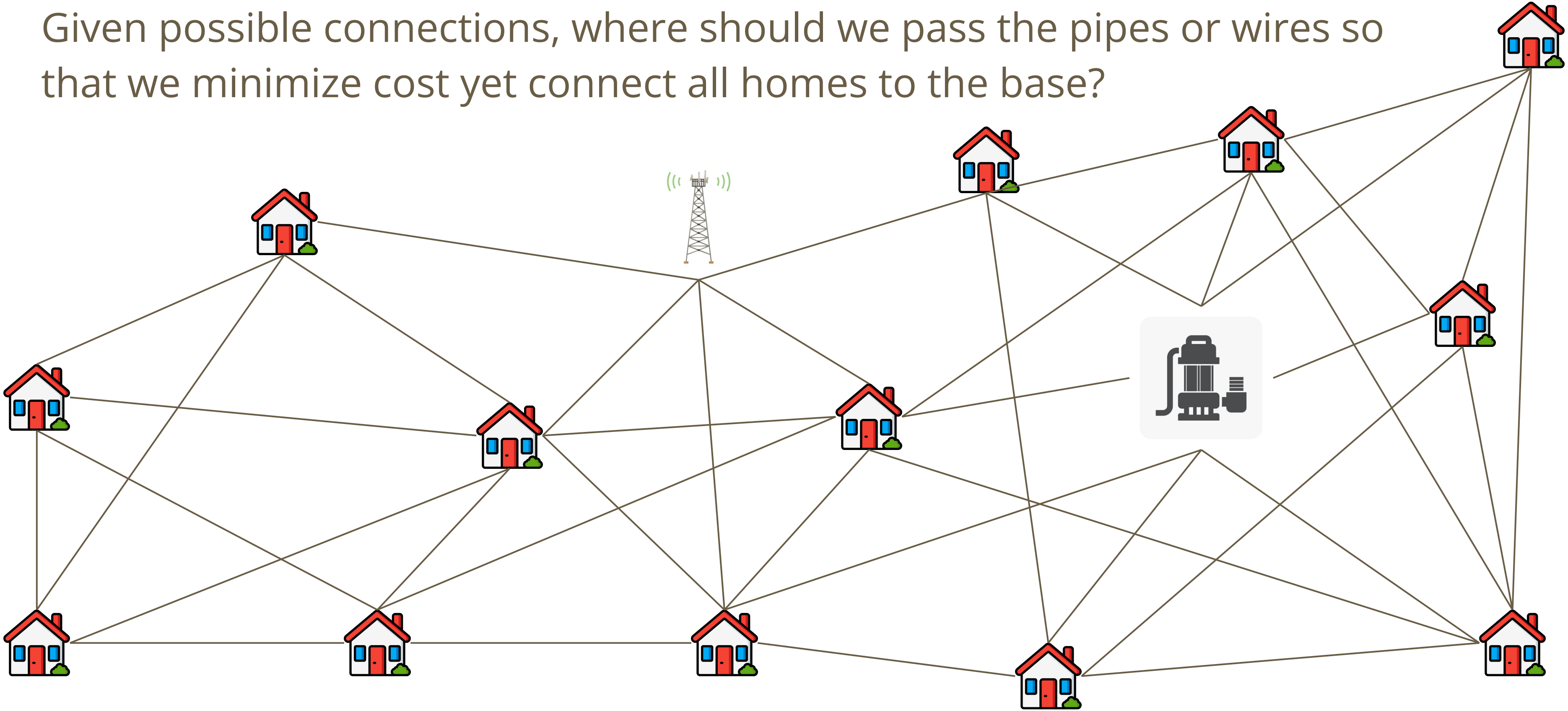
MST is not necessarily unique

- Undirected
  - Prim's → better on dense graphs - feels like uniform cost search
  - Kruskal's → better on sparse graphs - feels like greedy search
- Directed
  - Edmonds'

Details left for self study...

Can there be other trees? Other than MST?

Given possible connections, where should we pass the pipes or wires so that we minimize cost yet connect all homes to the base?



How about you are not the only actor?

- Multiplayer games
- Actions are probabilistic
- Closed-World assumption totally fails
- …

Minimax: Minimum damage control

	B_1	B_2	B_3	B_4
A_1	5	-2	-1	0
A_2	-3	4	1	-2
A_3	9	-5	4	-3

Payoff Matrix

At any point in the game it is A's turn

A can take 3 actions

In return B can take 4 actions

In the Payoff matrix: + is gain, - is loss

Which action should A choose to minimize potential damage / loss?

What if B is not that clever?

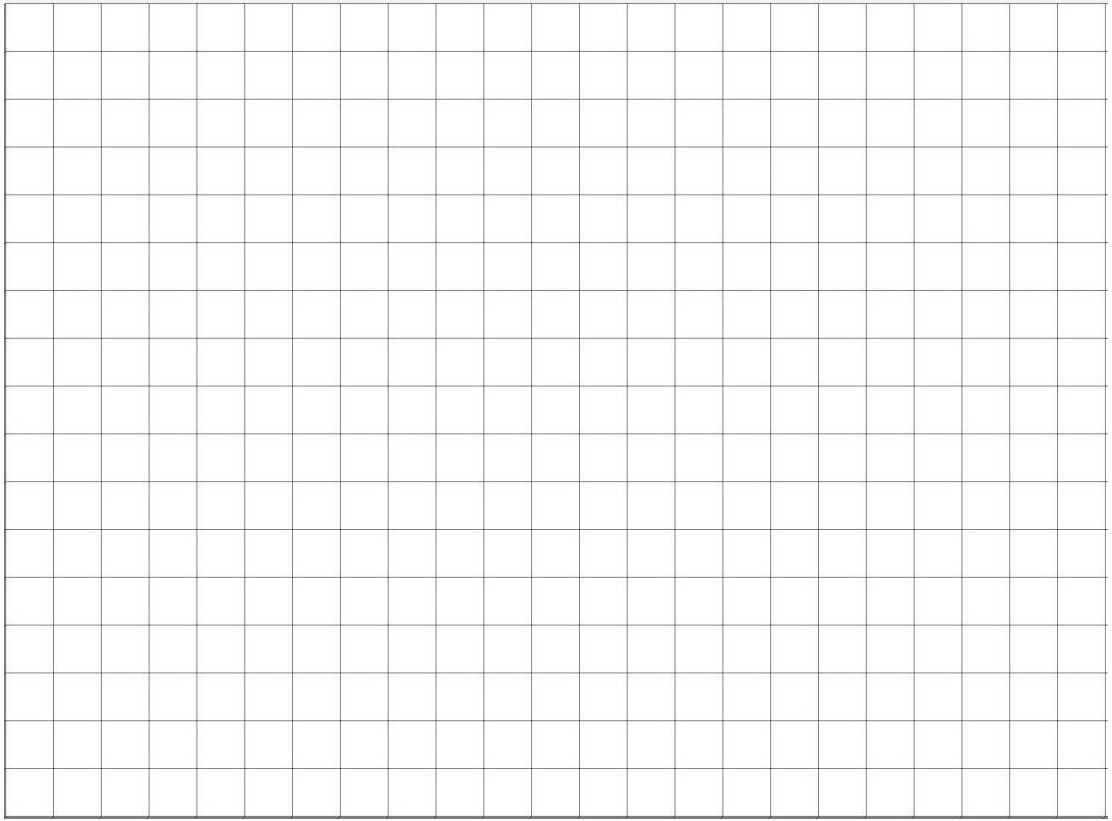
Perhaps: Assign probabilities to each possible action by B?

Design & Search: Search to Design

Search in a high dimensional space

States are not easily quantifiable

Tools of search might be useful in design



to be continued...