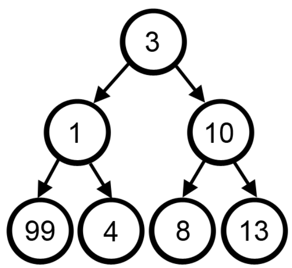
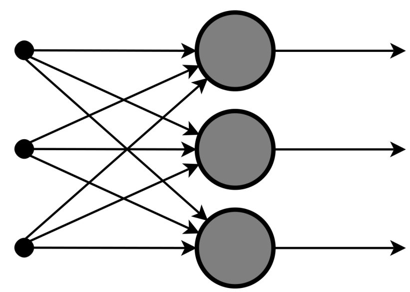
CSE 40171: Artificial Intelligence

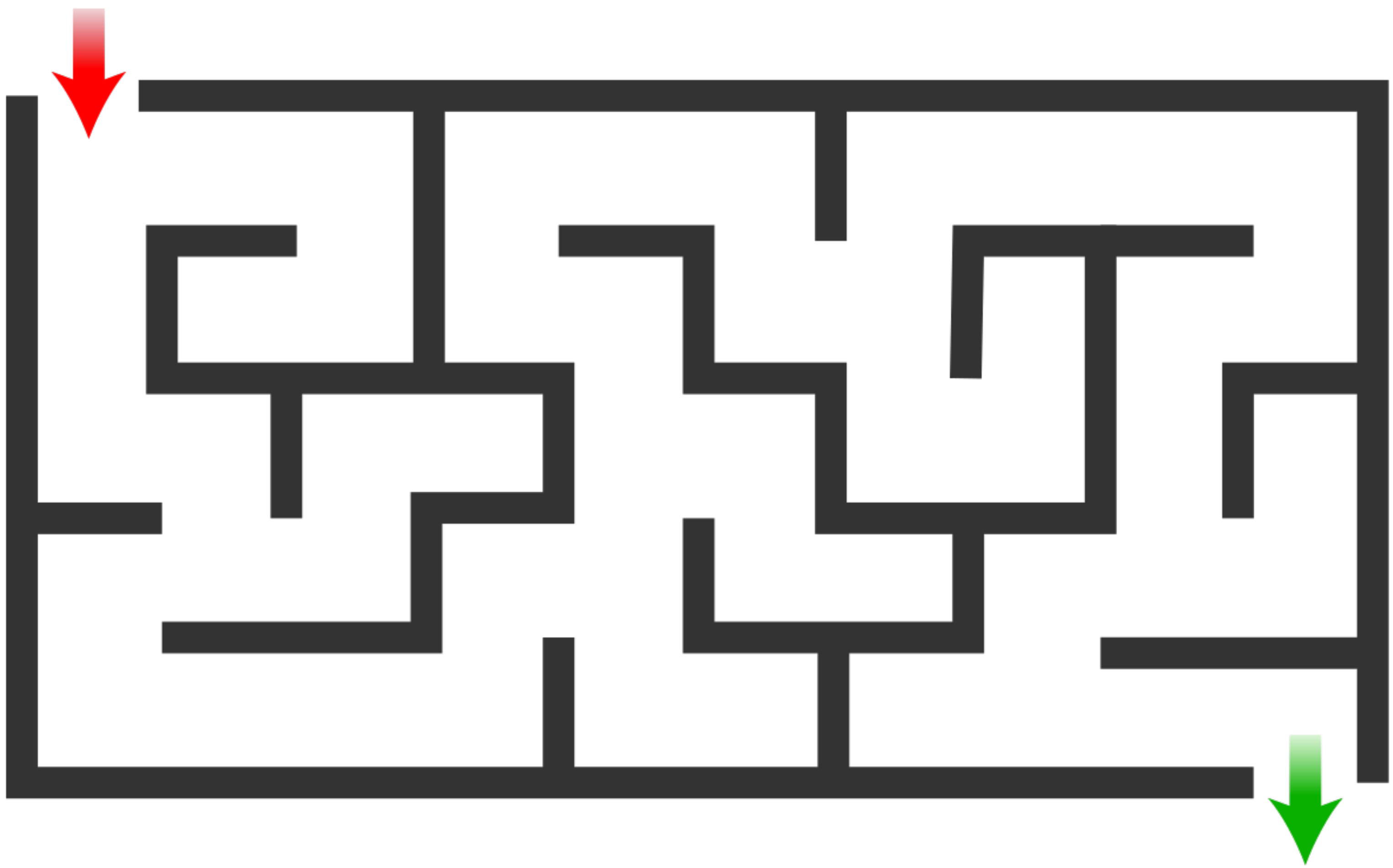




Informed Search: A* Search

Homework #2 has been released
It is due at 11:59PM on 9/30

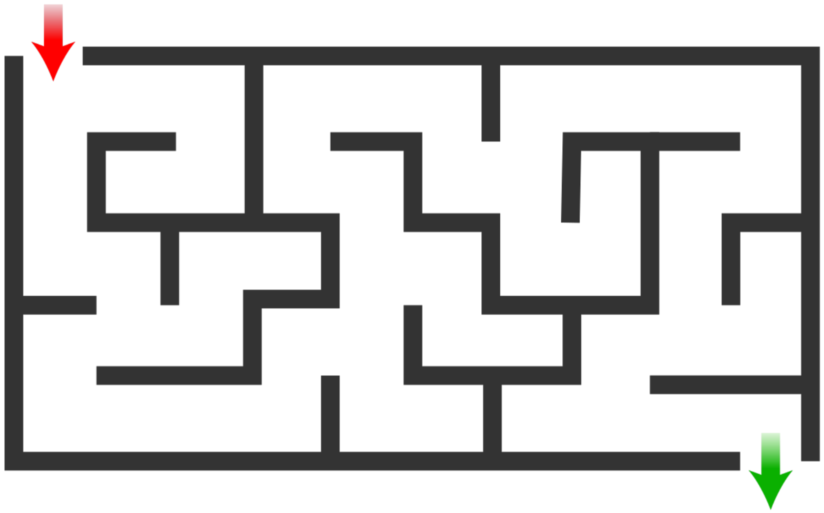
Quick Recap: Search



Quick Recap: Search

Search problem:

- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test



Search tree:

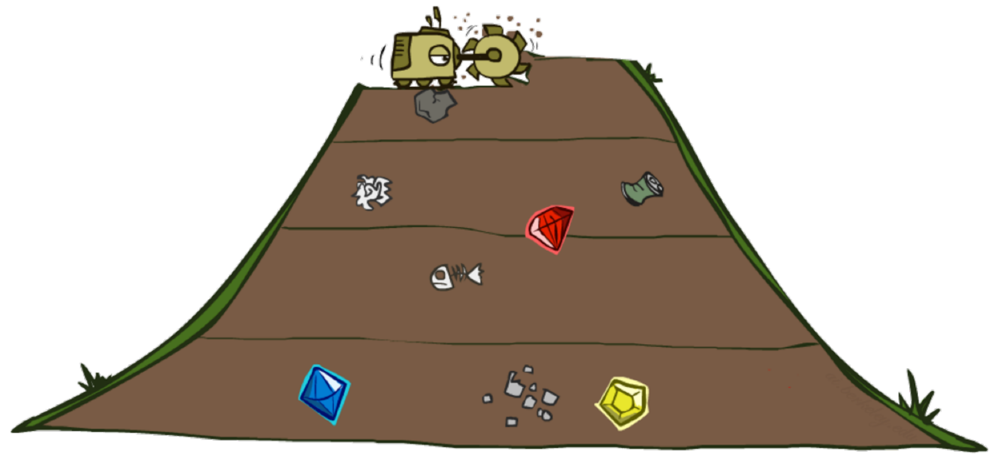
- Vertices: represent plans for reaching states
- Plans have costs (sum of action costs)

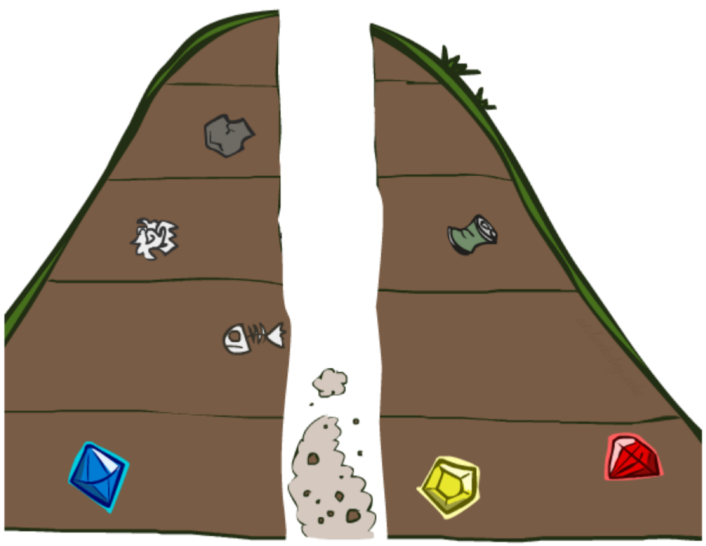
Search algorithm:

- Systematically builds a search tree
- Chooses an ordering of the fringe (unexplored nodes)
- Optimal: finds least-cost plans

Quick Recap: Search

Breadth-First Search



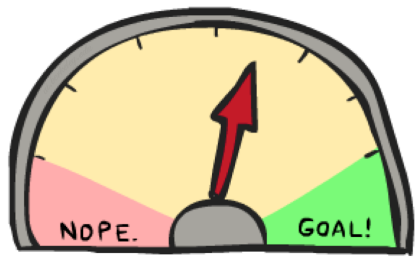


Depth-First Search

Image credit: Dan Klein and Pieter Abbeel, UC Berkeley CS 188

What was wrong with uninformed search?

- Did not make use of problem-specific knowledge beyond the definition of the problem itself
- Was not efficient



Informed Search

Image credit: Dan Klein and Pieter Abbeel, UC Berkeley CS 188

New Aspects of Informed Search

The general approach we will consider is **best-first search**

- Instance of TREE-SEARCH or GRAPH-SEARCH

A vertex is selected for expansion based on an **evaluation function**, $f(v)$

- vertex v with the lowest evaluation is expanded first

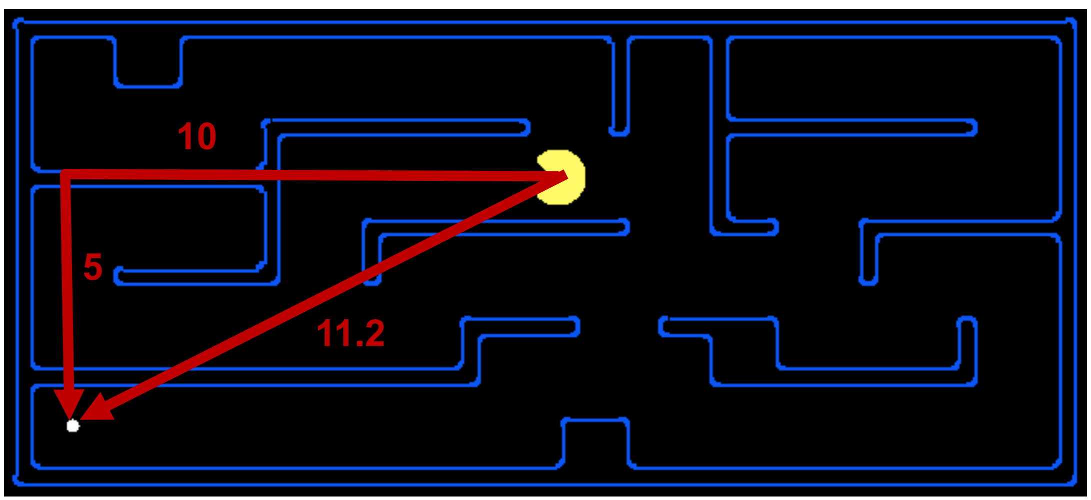
Most best-first search algorithms include as a component of a heuristic function, $h(v)$

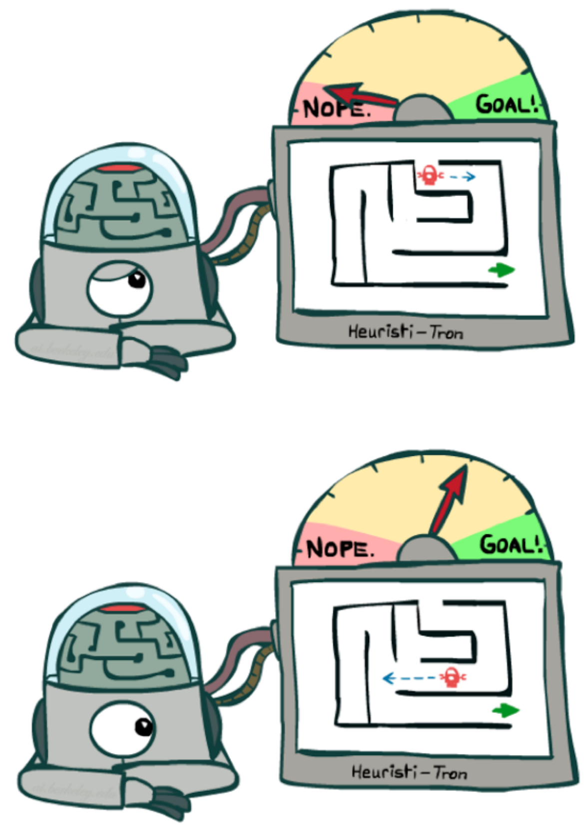
- $h(v)$ = estimated cost of the cheapest path from the site at vertex v to a goal state

Search Heuristics

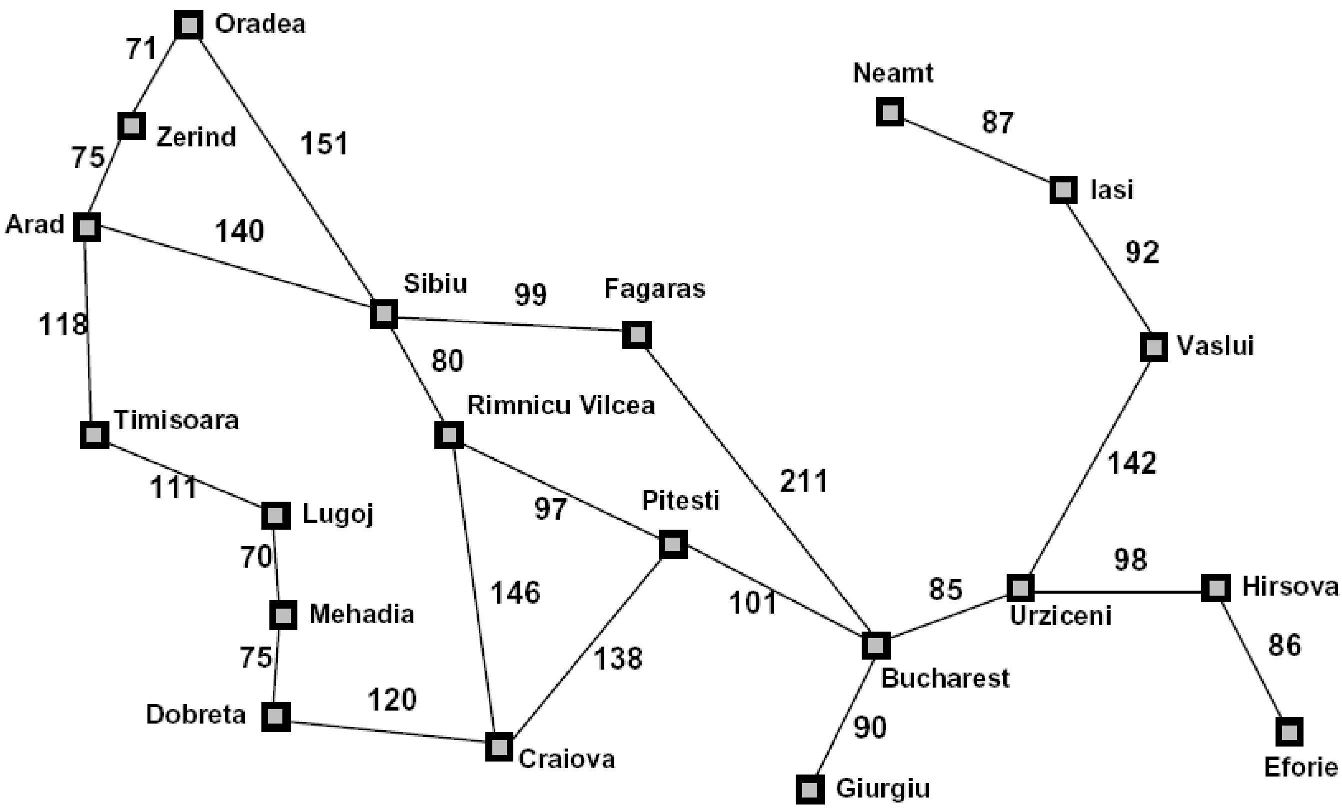
A heuristic in this context is:

- A function that estimates how close a state is to a goal
- Designed for a particular search problem
- Examples: Manhattan distance, Euclidean distance for path planning





Slide credit: Dan Klein and Pieter Abbeel, UC Berkeley CS 188



Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

$h(v)$

Image Credit: Russel and Norvig

Greedy Best-First Search

Image credit: Dan Klein and Pieter Abbeel, UC Berkeley CS 188

Greedy Best-First Search

Strategy: expand the vertex that is closest to the goal

Assumption: this is likely to lead to a solution quickly

Heuristic Function: $f(v) = h(v)$

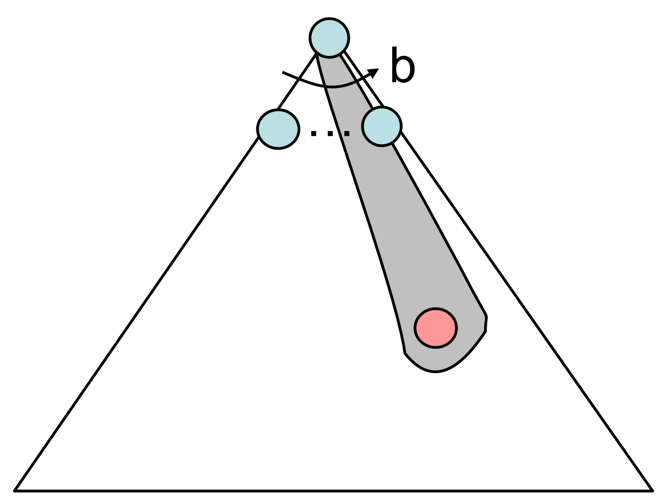
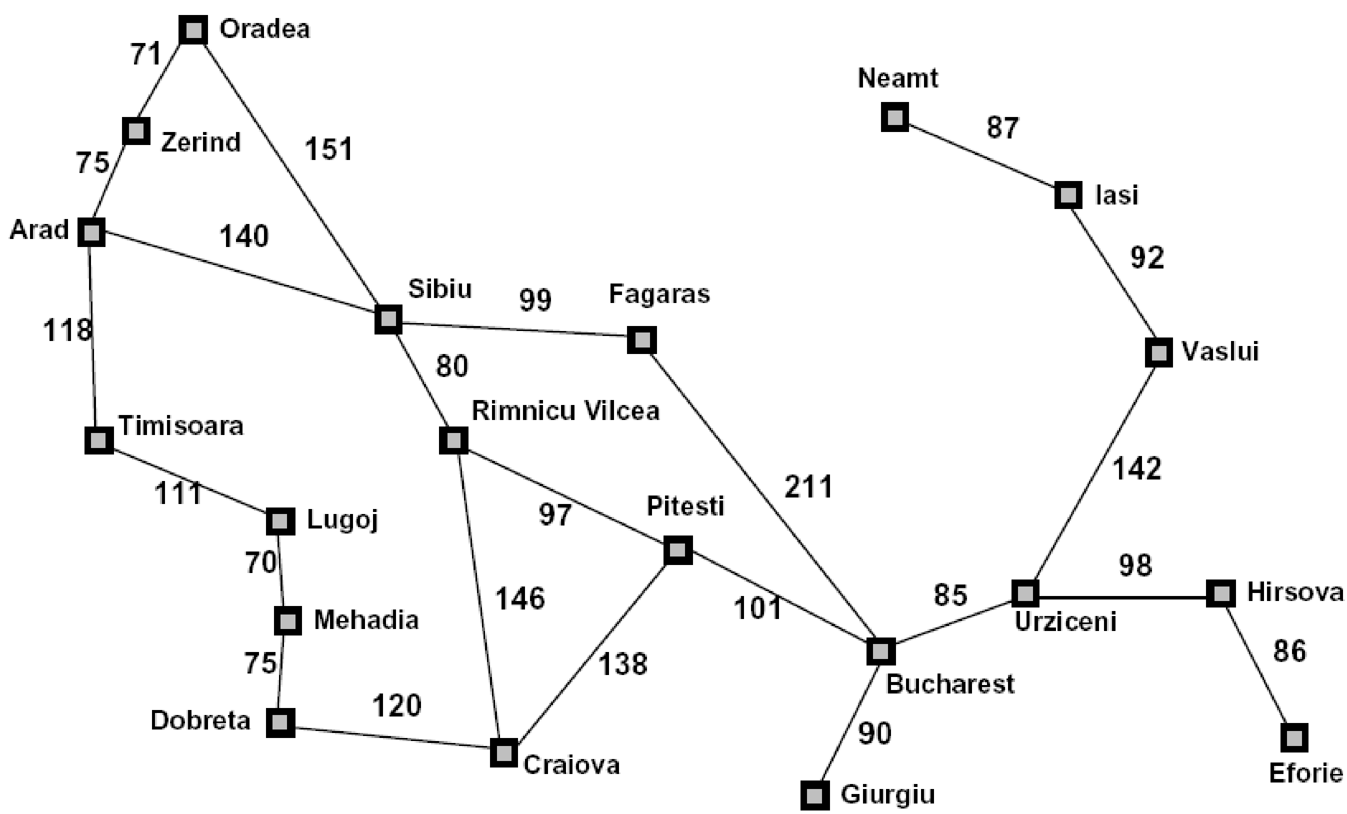


Image credit: Dan Klein and Pieter Abbeel, UC Berkeley CS 188

Example with the following heuristic



Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

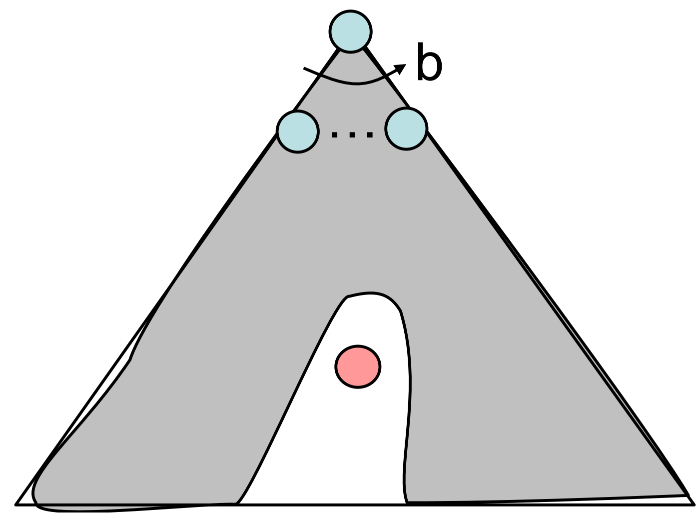
$h(v)$

Image Credit: Russel and Norvig

Practical Problems with Greedy Best-First Search

A common case: best first takes you straight to the wrong goal

The worst case: like a badly guided depth-first search



Analysis of Greedy Best-First Search

Assume a uniform tree where every state has b successors

Completeness: incomplete in a finite state space (just like depth-first search)

Optimality: the algorithm is not optimal

- In our example, we found the path Arad → Sibiu → Fagaras → Bucharest. But this is 32KM *longer* than the path going from Arad → Sibiu → Rimnicu Vilcea → Pitesti → Bucharest.

Time complexity: $O(b^m)$, where m is the maximum depth of the search space

Space complexity: $O(b^m)$

A* Search

Image credit: Dan Klein and Pieter Abbeel, UC Berkeley CS 188

A* Search

A* search is the most widely known form of best-first search

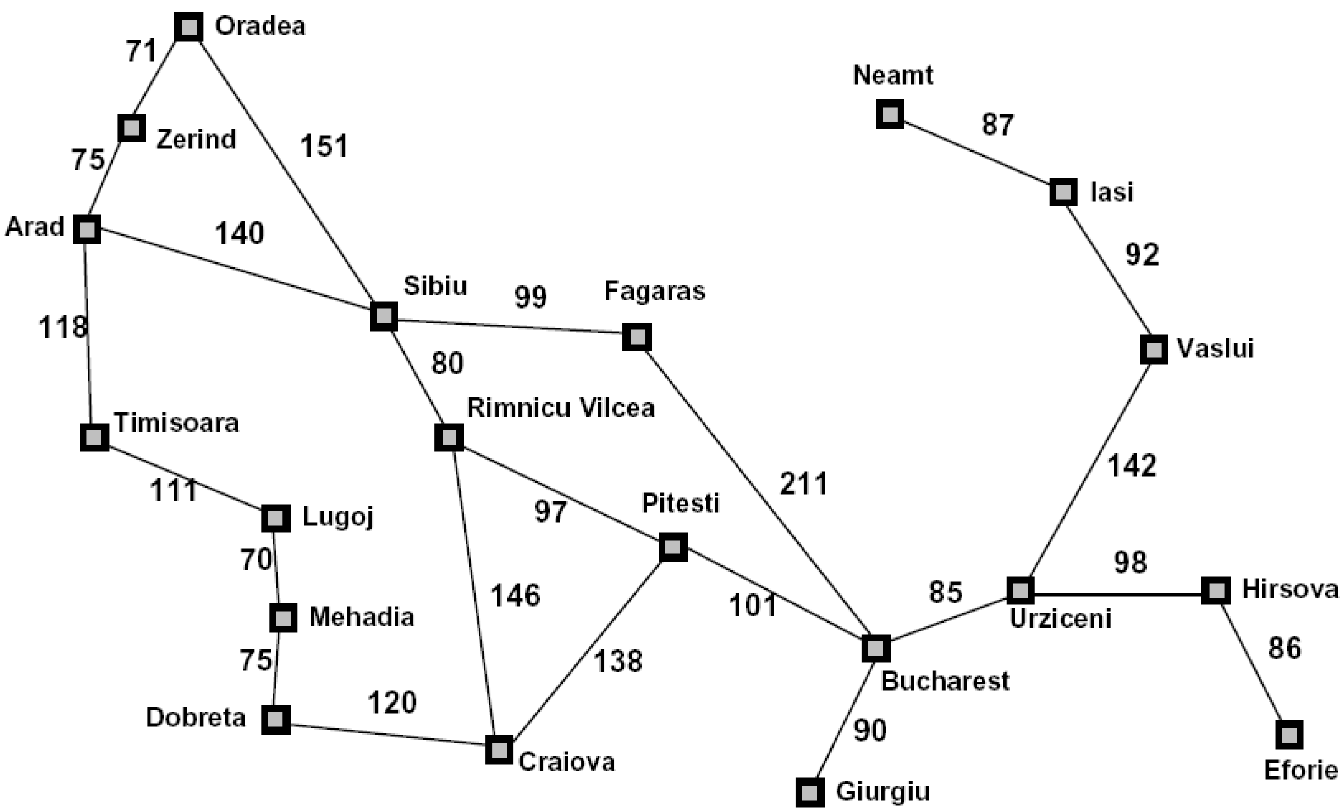
$g(v)$: the cost to reach the vertex

$h(v)$: the cost to get from the vertex to the goal

Vertices are evaluated via: $f(v) = g(v) + h(v)$

i.e., $f(v) =$ estimated cost of the cheapest solution through v

Example with the following heuristic

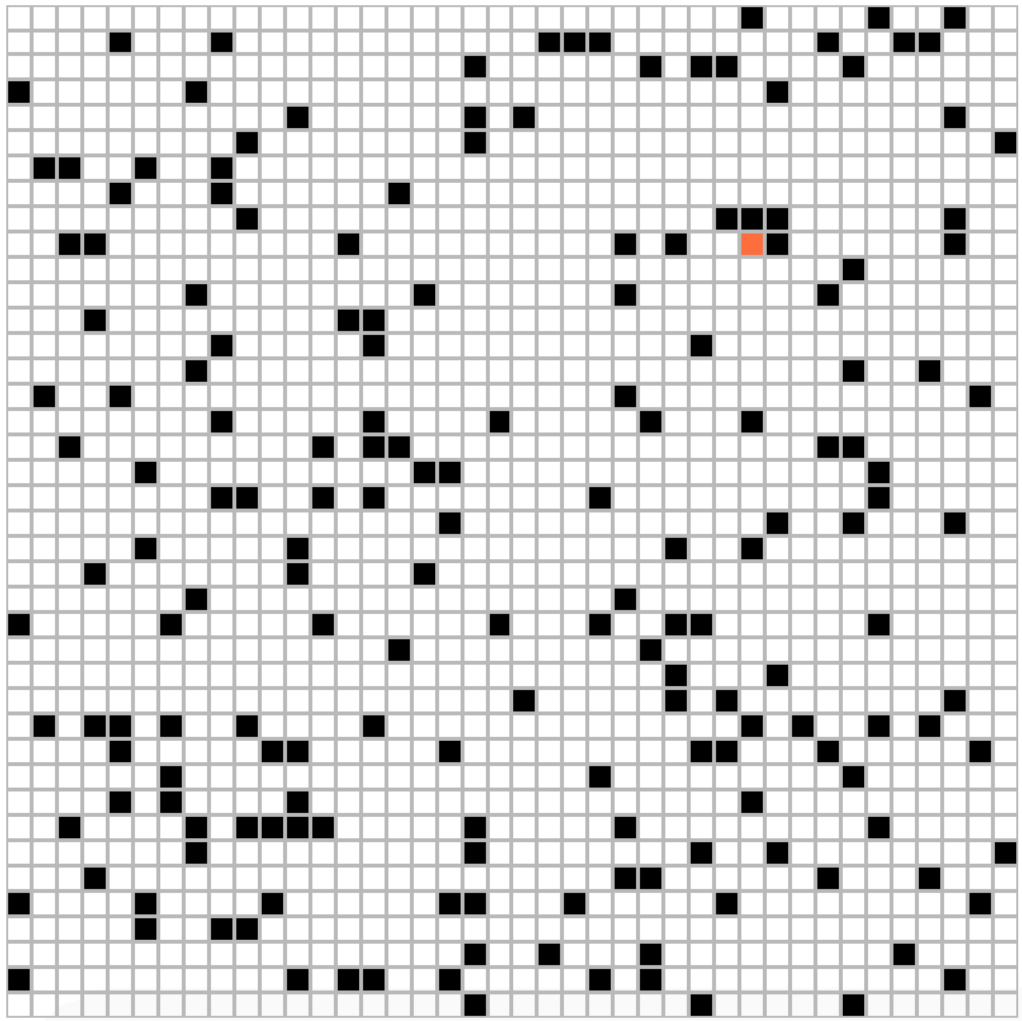


Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

***g*(*v*): path costs in graph** ***h*(*v*)**

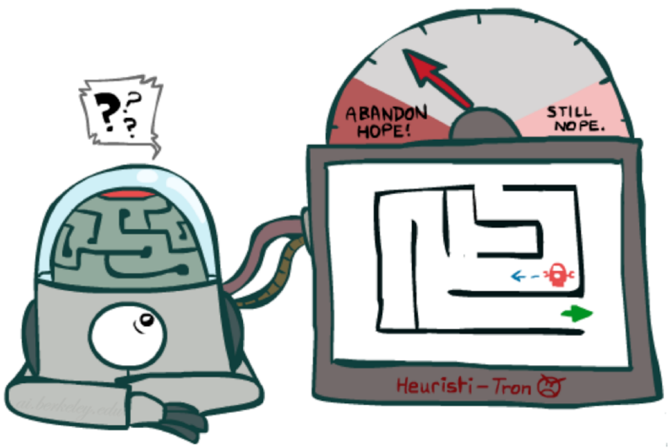
Image Credit: Russel and Norvig

Visual Example of A* Search

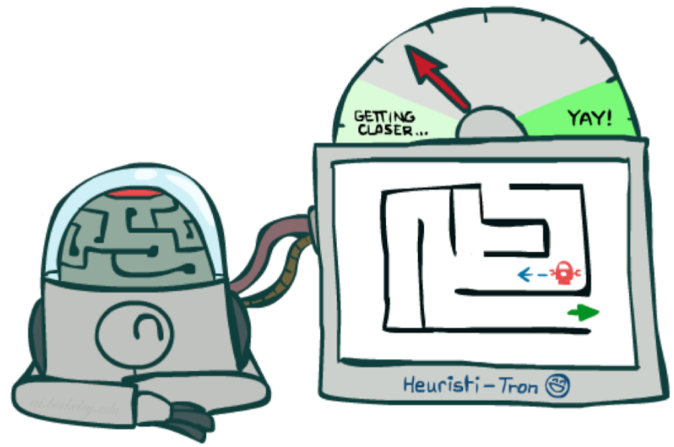


https://bgrins.github.io/javascript-astar/demo/

Conditions for Optimality in TREE-SEARCH: Admissibility



Inadmissible (pessimistic) heuristics break optimality by trapping good plans on the fringe



Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs

Slide credit: Dan Klein and Pieter Abbeel, UC Berkeley CS 188

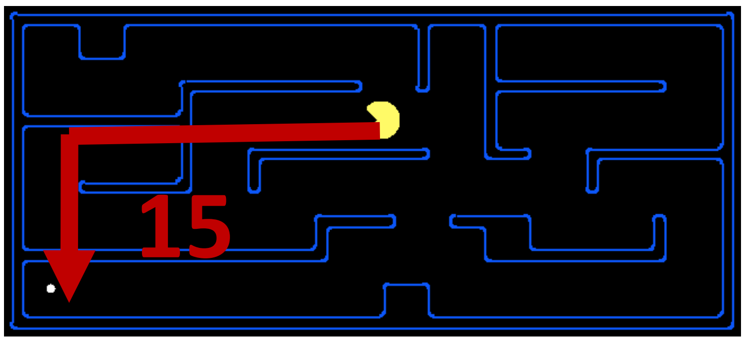
Conditions for Optimality in TREE-SEARCH: Admissibility

A heuristic h is admissible (optimistic) if:

$$0 \leq h(v) \leq h^*(v)$$

where $h^*(v)$ is the true cost to a nearest goal

Example:



Coming up with admissible heuristics is most of what is involved in using A* in practice.

Slide credit: Dan Klein and Pieter Abbeel, UC Berkeley CS 188

Conditions for Optimality in GRAPH-SEARCH: Consistency

$h(v)$ is consistent if:

For every vertex v and every successor v' of v generated by any action a, the estimated cost of reaching the goal from v is not greater than the step cost of getting v' plus the estimated cost of reaching the goal from v'

$h(v) \leq c(v, a, v') + h(v')$ ← **Form of the triangle inequality**

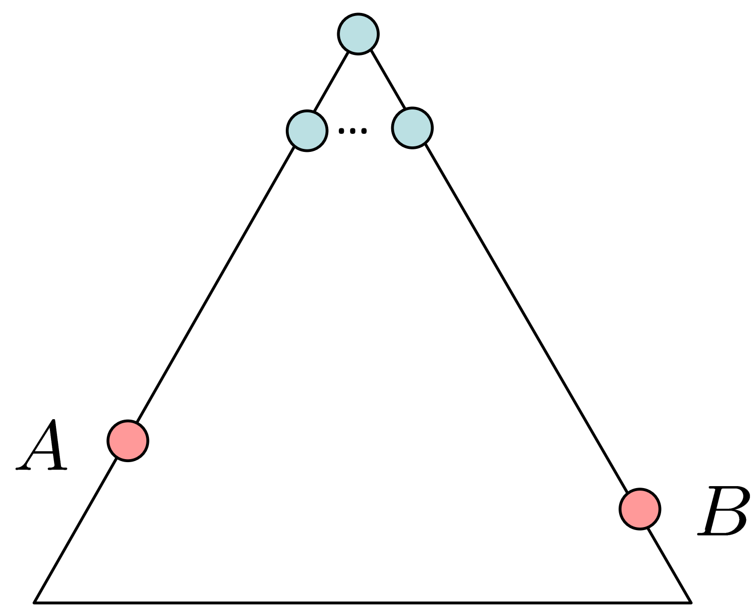
Optimality of A* TREE-SEARCH

Assume:

- A is an optimal goal vertex
- B is a suboptimal goal vertex
- h is admissible

Claim:

- A will exit the fringe before B

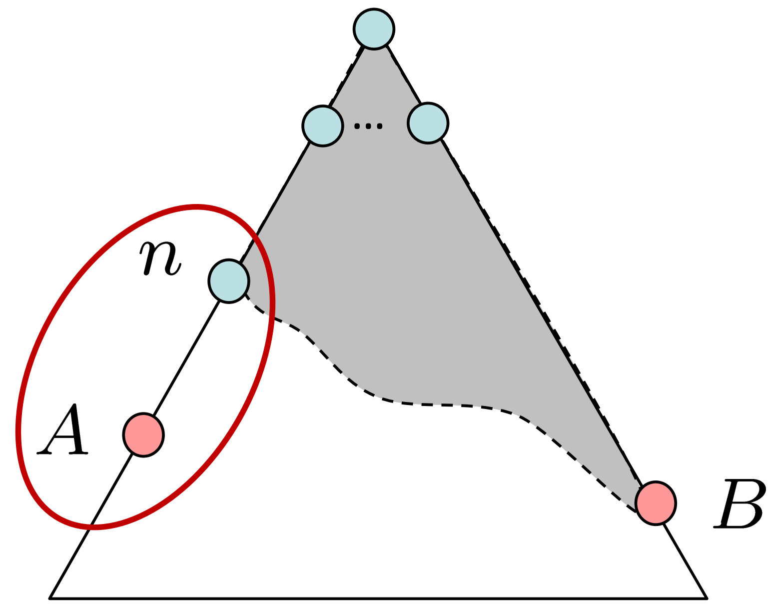


Slide credit: Dan Klein and Pieter Abbeel, UC Berkeley CS 188

Optimality of A* TREE-SEARCH

Proof:

- Assume B is on the fringe
- Some ancestor n of A is on the fringe, too (possibly A!)
- Claim: n will be expanded before B



1. $f(n)$ is less or equal to $f(A)$

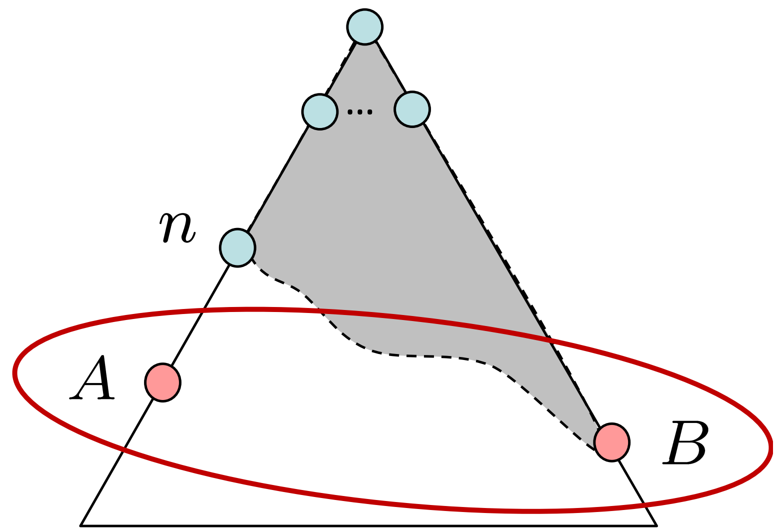
$f(n) = g(n) + h(n)$	Definition of f-cost
$f(n) \leq g(A)$	Admissibility of h
$g(A) = f(A)$	$h = 0$ at a goal

Slide credit: Dan Klein and Pieter Abbeel, UC Berkeley CS 188

Optimality of A* TREE-SEARCH

Proof:

- Assume B is on the fringe
- Some ancestor n of A is on the fringe, too (possibly A!)
- Claim: n will be expanded before B



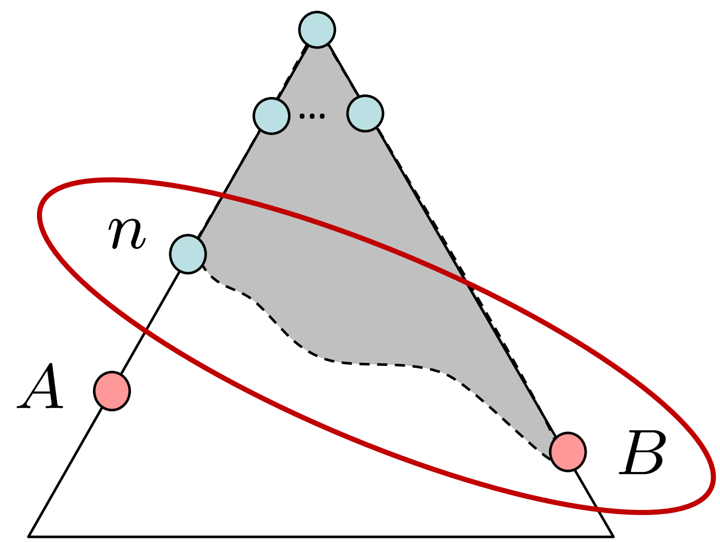
1. $f(n)$ is less or equal to $f(A)$
2. $f(A)$ is less than $f(B)$

$g(A) < g(B)$	B is suboptimal
$f(A) < f(B)$	$h = 0$ at a goal

Optimality of A* TREE-SEARCH

Proof:

- Assume B is on the fringe
- Some ancestor n of A is on the fringe, too (possibly A!)
- Claim: n will be expanded before B



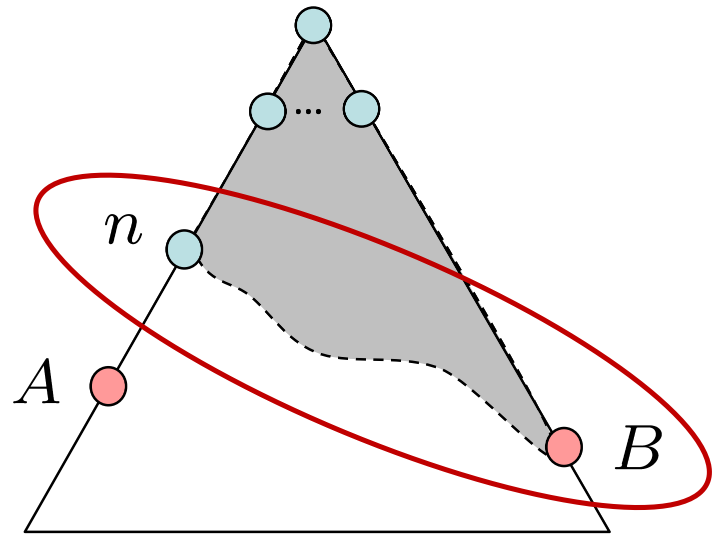
1. $f(n)$ is less or equal to $f(A)$
2. $f(A)$ is less than $f(B)$
3. n expands before B

$$f(n) \leq f(A) < f(B)$$

Slide credit: Dan Klein and Pieter Abbeel, UC Berkeley CS 188

Optimality of A* TREE-SEARCH

- All ancestors of *A* expand before *B*
- *A* expands before *B*
- A* search is optimal



Complexity and Completeness of A*

Depth of tree: d

Assume a uniform tree where every state has b successors

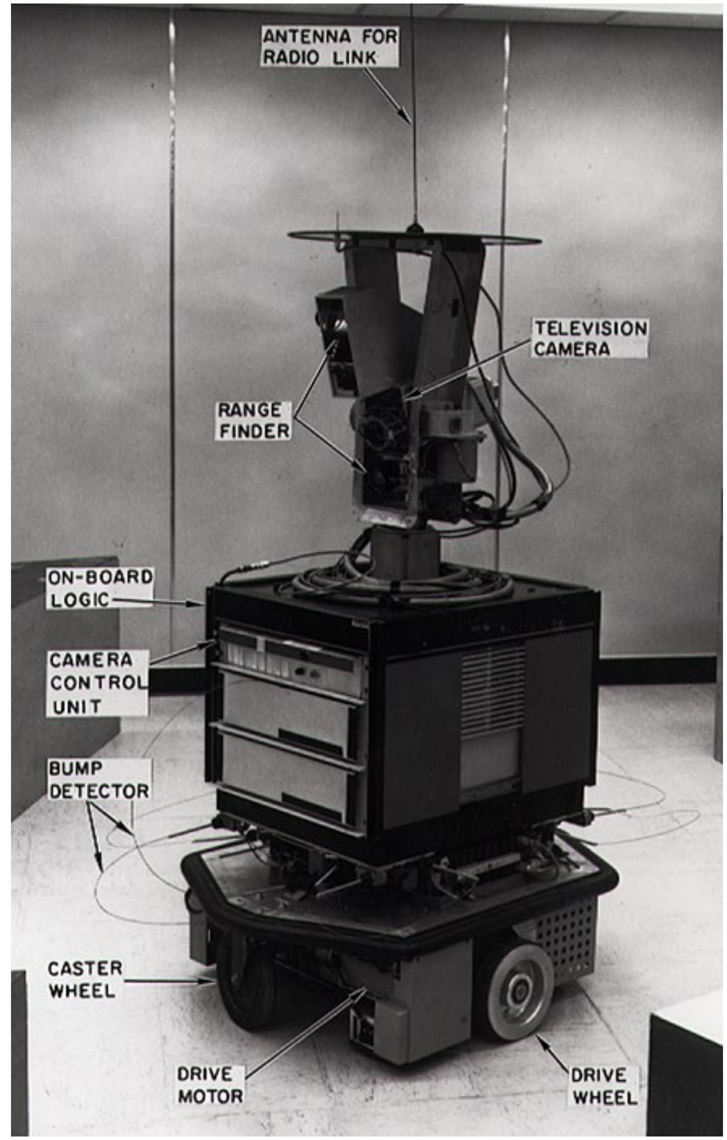
Completeness: A* search is complete

Time complexity: $O(|E|) = O(b^d)$

Space complexity: $O(|V|) = O(b^d)$

Applications of A* Search

Originally developed as a path planner for *Shakey the Robot* at Stanford (1968)



Applications of A* Search

games-like-age-of-empires-798x350 (cc) BY 2.0 Siddhartha Thota

Commonly used in games where terrain is mapped to a grid