AIML231/DATA302 —Techniques in Machine Learning

Week 10 - Search Methods in Machine Learning

Dr Qi Chen

School of Engineering and Computer Science

Victoria University of Wellington

Qi.Chen@vuw.ac.nz

Outline

- What is search in AI/machine learning
- The role of search in AI/machine learning
- Explore different types of search methods used in ML, discuss applications and limitations of each method
  - Breadth first
  - Uniform cost
  - Greedy(best-first) search
  - A* search
  - Hill climbing
  - Gradient descent
  - Simulated Annealing
  - Beam search

Search in AI/ML

- In AI/ML, search refers to the process of exploring a vast space of possible solutions/states/configuration to find an optimal or near-optimal solutions/states/configuration for a given problem

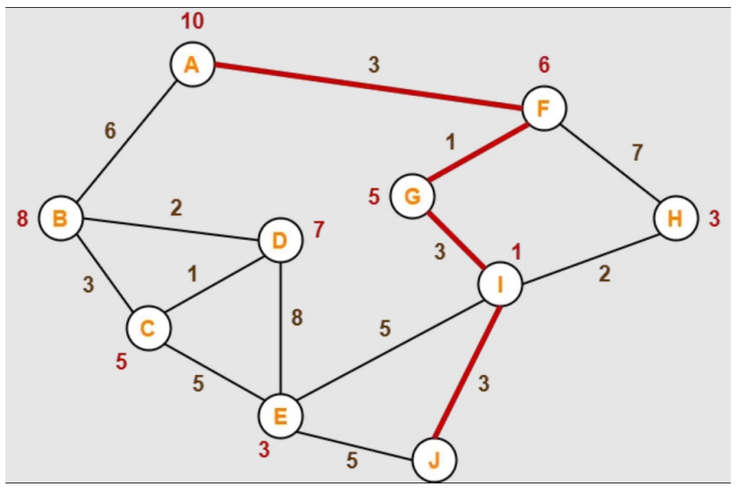
  - explore possible states in a problem space to find an optimal solution, e.g., solving puzzles, pathfinding in maps, or game playing

Initial State

1	2	3
8		4
7	6	5

Goal State

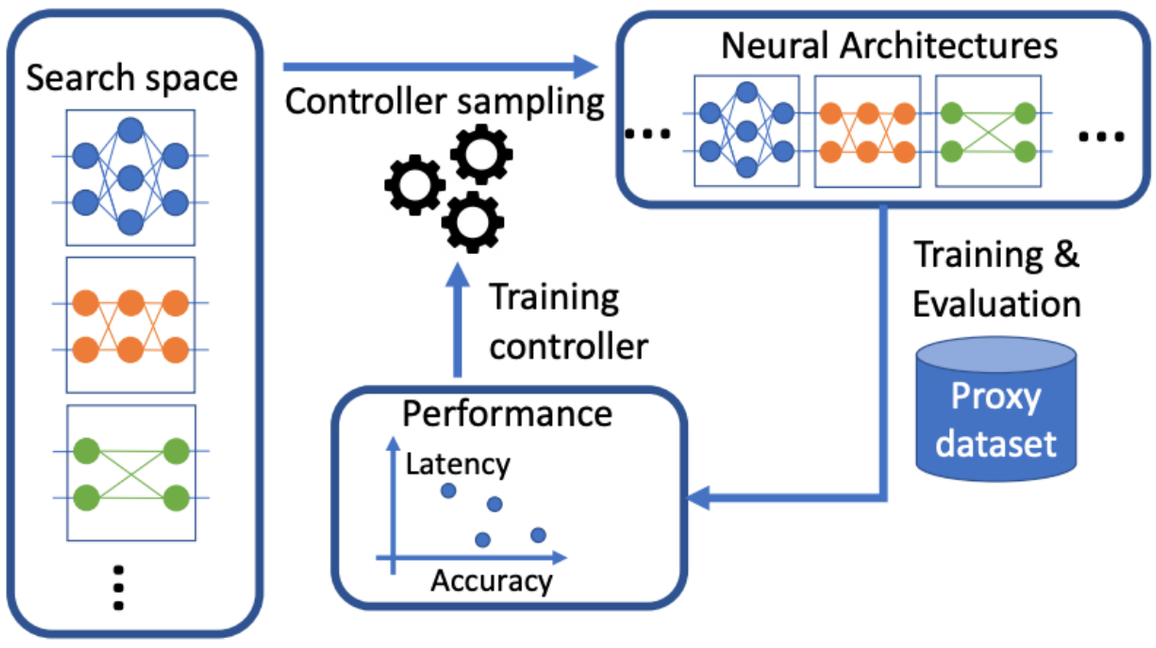
2	8	1
	4	3
7	6	5



Search in AI/ML

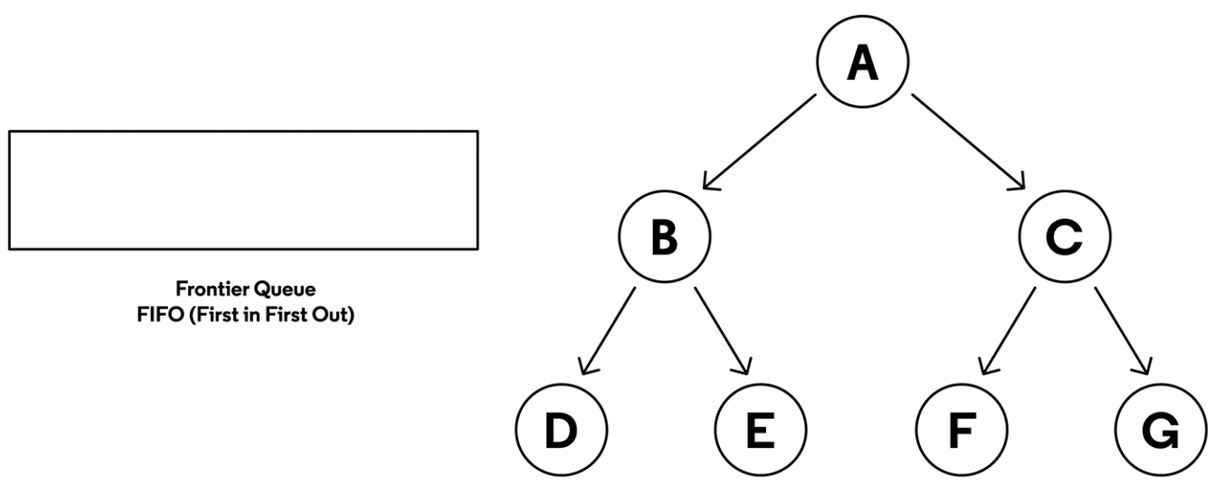
- Search is a fundamental mechanism for problem-solving and decision-making in ML

  - find the best solution from a set of possible solutions, e.g., search through different model parameters to find the ones that best fit the data, feature selection, weights optimisation/neural architecture search in NNs…



General Search Algorithm

- *Initialization:* start with an initial state or node
- *Node Expansion:* expand the node by generating successors
- Goal Test: each expanded node is checked against the goal criteria. If the goal is met, the algorithm terminates and returns the solution
- *Strategy for Node Selection:* use a specific strategy to decide which of the available nodes to explore next
- *Loop*: The process repeats with the new current node until the goal is reached or no more nodes are available for expansion

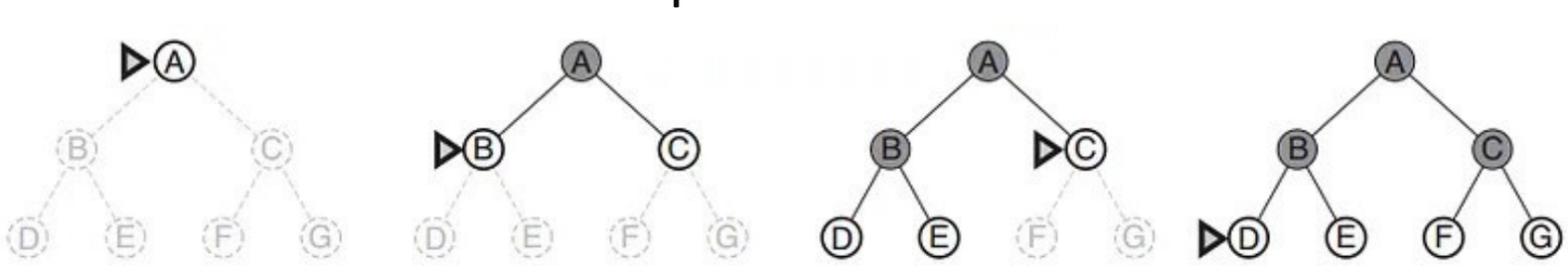


Search strategies

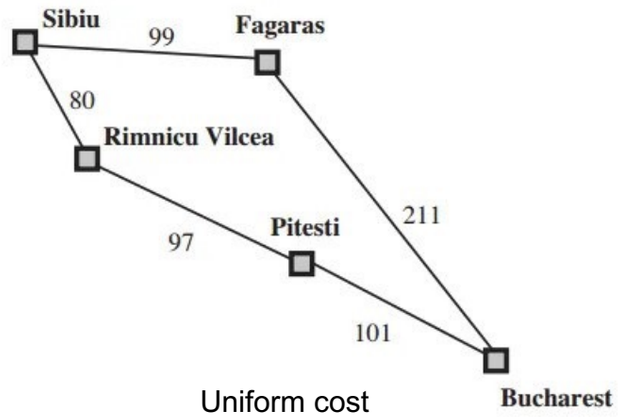
- Uninformed search(blind search)
  - Breadth first, Uniform cost, Depth first, Depth limited, Iterative deepening, Bidirectional
- Informed (Heuristic) search
  - Greedy(best-first) search
  - A* search
- Beyond classic search
  - Hill climbing
  - **Gradient descent**
  - Simulated Annealing
  - Beam search
  - Bound and bound
  - dynamic programming

Uninformed Search

- AKA, blind search, operate without any information about the number of steps or the path cost from any node in the search tree to the goal state
  - only rely on the problem's inherent configuration to make decisions about which nodes to expand



Breadth first



Uniform cost

  - explore all or most of the state space without guidance on which paths might lead more directly to the goal
  - will find a solution if one exists, and some of these algorithms are optimal
  - high time and space complexities
  - a foundational technique in AI when little to no heuristic information

Informed (Heuristic) search

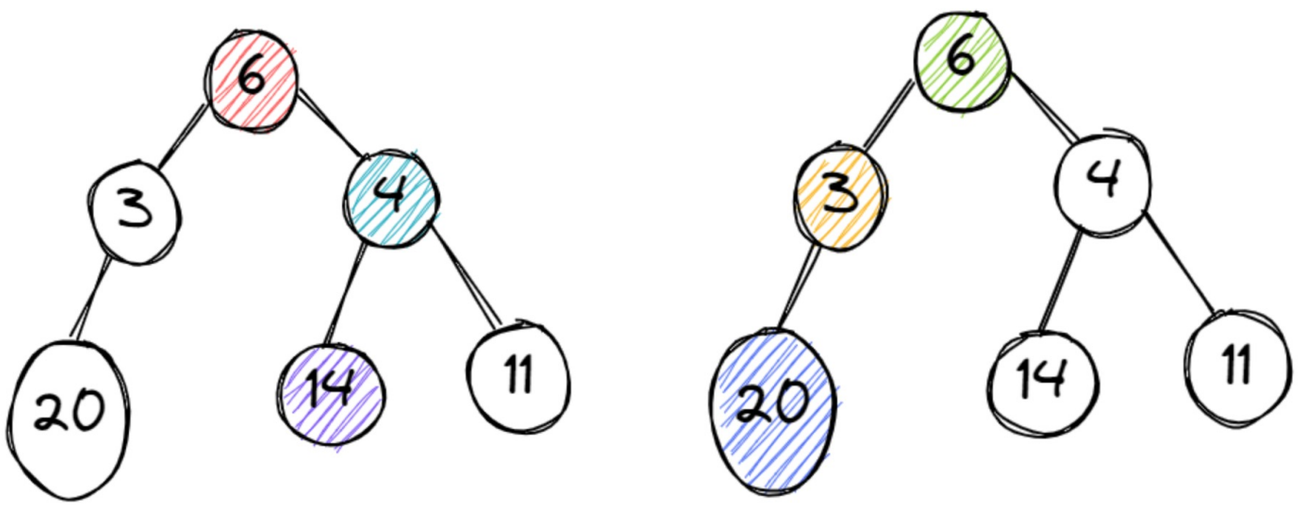
- use problem-specific knowledge, or heuristics, to guide the search process toward the goal more efficiently than uninformed search methods.
  - estimate the cost or distance to the goal, prioritize paths that are more likely to lead to a solution quickly

  - Heuristic Function: a heuristic function $h(n)$ estimates the cost or distance from node n to the goal, nodes are explored based on the estimated cost, leading to faster solutions

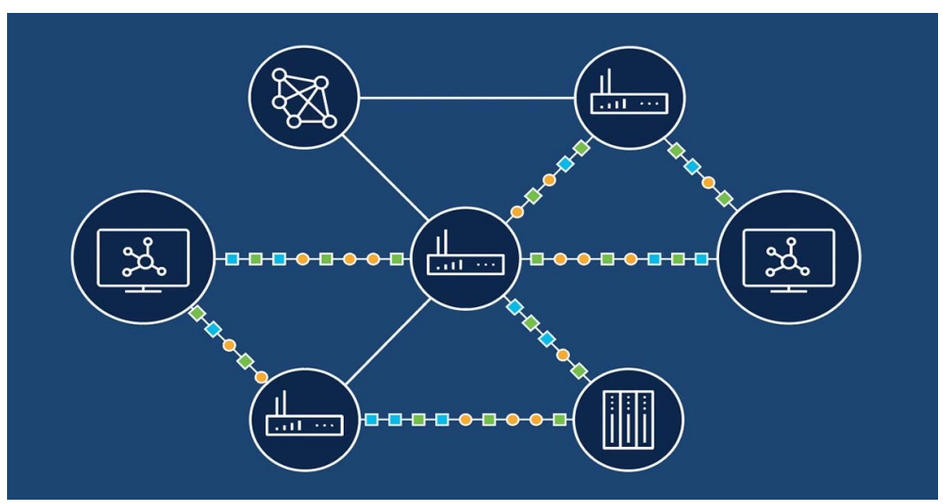
  - include the classical one: Greedy(best-first) search, A* search and more advanced: Hill climbing, Gradient descent, Simulated Annealing, Beam search

Greedy Best-First Search

- use the heuristic function $h(n)$ to choose the next node that appears closest to the goal (best first)
  - does not consider the cost from the start node but only focuses on the estimate to the goal
  - the effectiveness largely depends on the quality of the heuristic, can be very efficient with a good heuristic function
  - may not find the optimal path, even might fail to find a solution
  - use when the goal is clearly defined, when there is a reliable, effective heuristic, when the problem‘s search space is vast and computational resources are limited, when speed is more critical than accuracy



Greedy search v.s. not-greedy

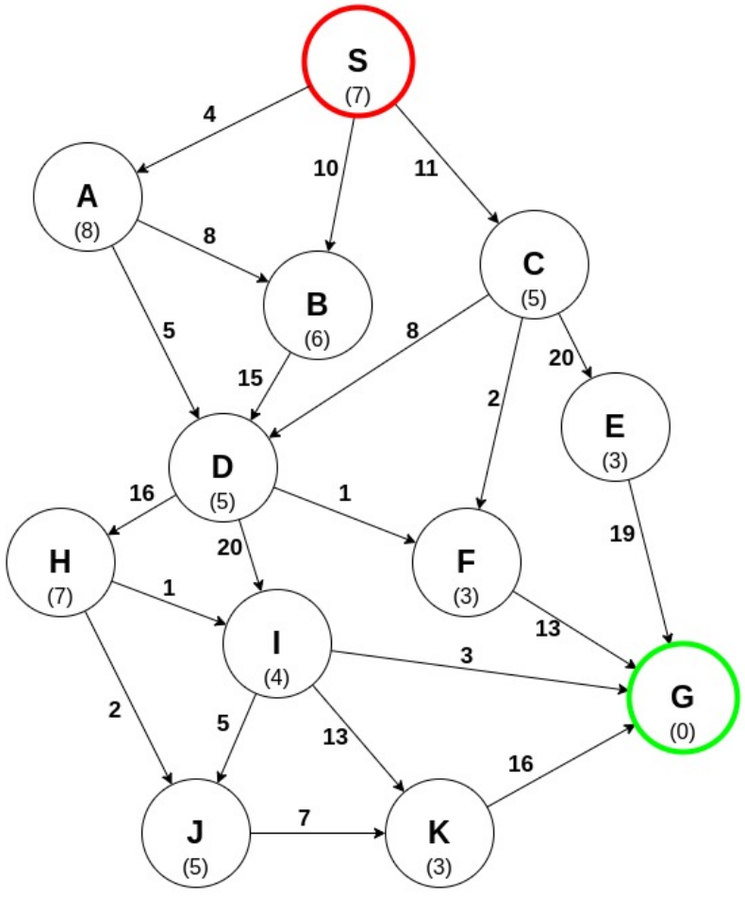


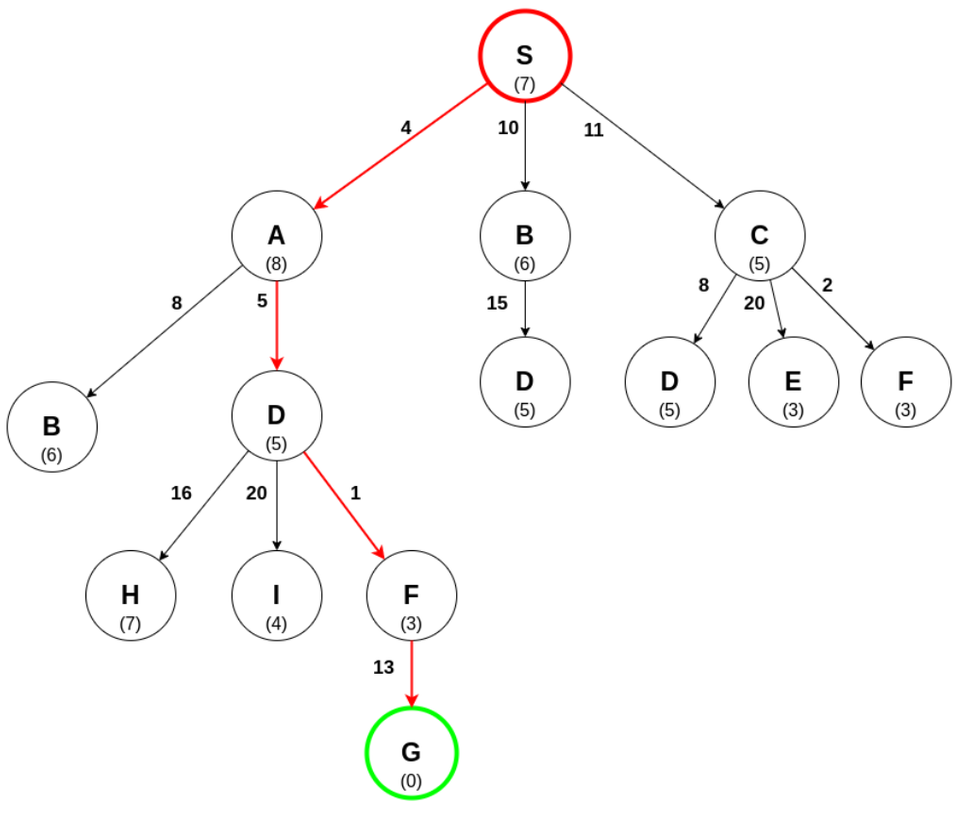
Network routing with Greedy search

https://www.freecodecamp.org/news/greedy-algorithms/

A* Search

- designed to find the most efficient path between an initial node (start) and a target node (goal)
- the estimated total cost of the cheapest solution through a node n: utilise both the cost to reach n $g(n)$ and the heuristic function $h(n)$ *to* estimate of the cost to get from that node to the goal to evaluate nodes: $f(n)=g(n)+h(n)$



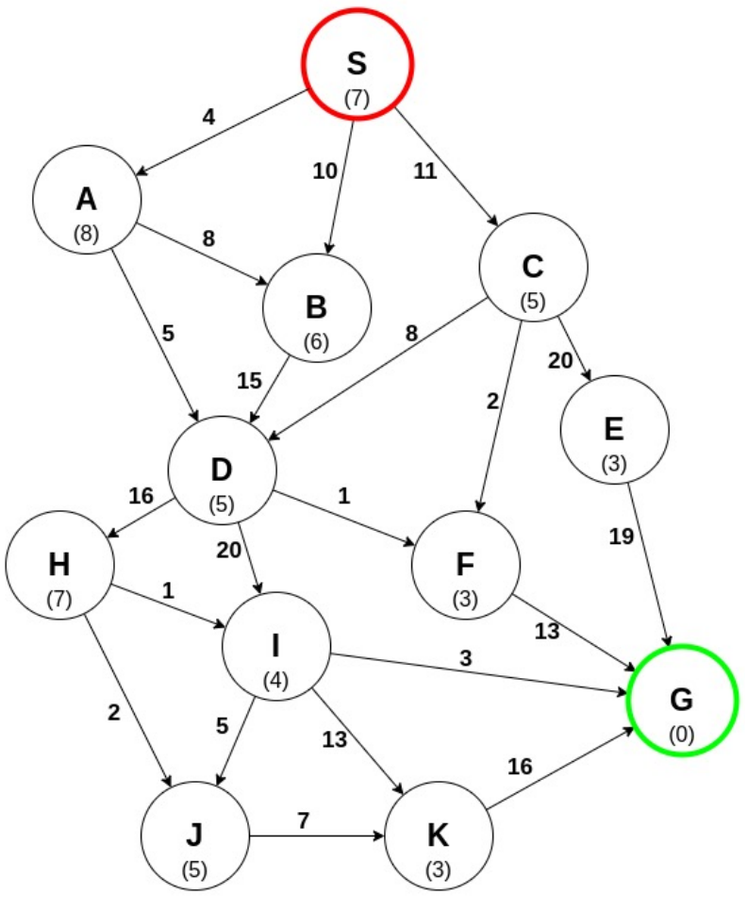


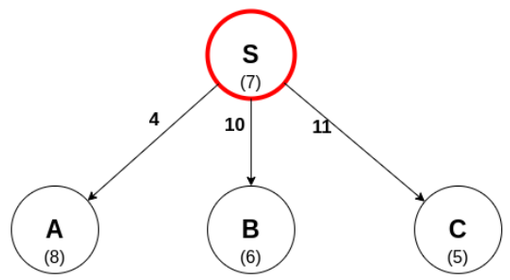
Node[cost]
I[33]
H[32]
E[36]

Closed List
S
A
D
F
B
C
G

the solution found by A* search

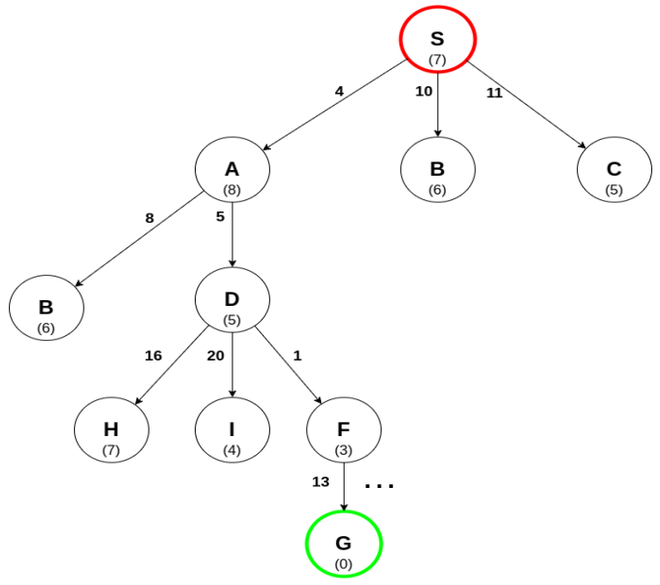
A* Search





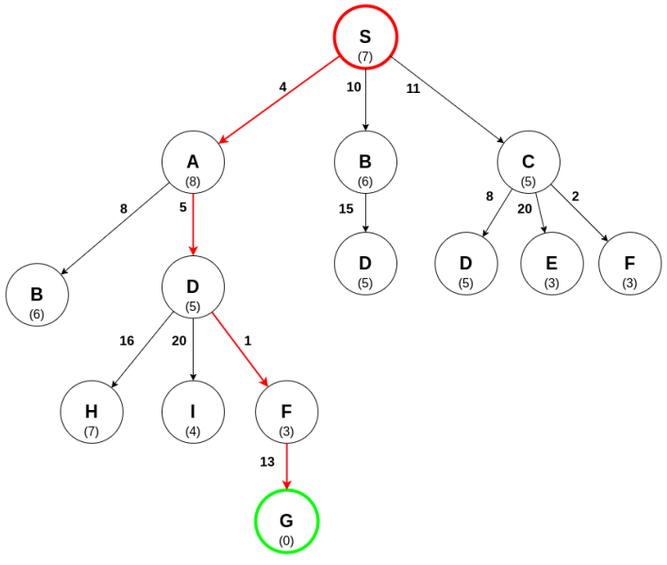
Node[cost]
A[12]
B[16]
C[16]

Closed List
S



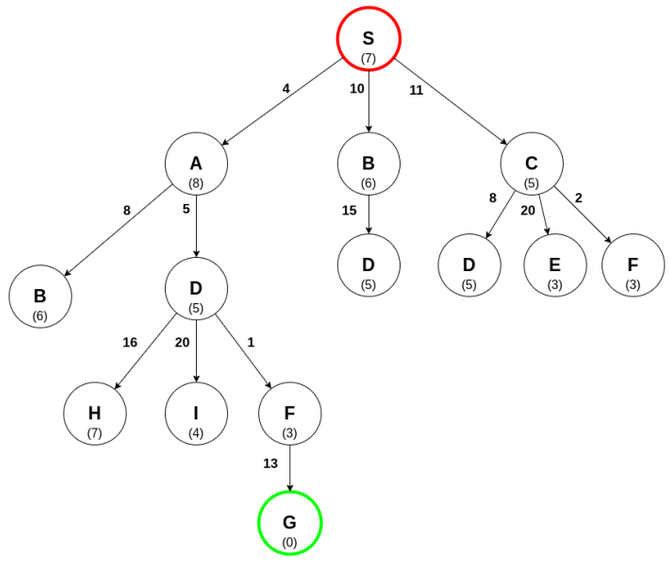
Node[cost]
B[16]
C[16]
B[18]
H[32]
I[33]
G[23]

Closed List
S
A
D
F



Node[cost]
I[33]
H[32]
E[36]

Closed List
S
A
D
F
B
C
G



Node[cost]
B[18]
G[23]
I[33]
H[32]
E[36]

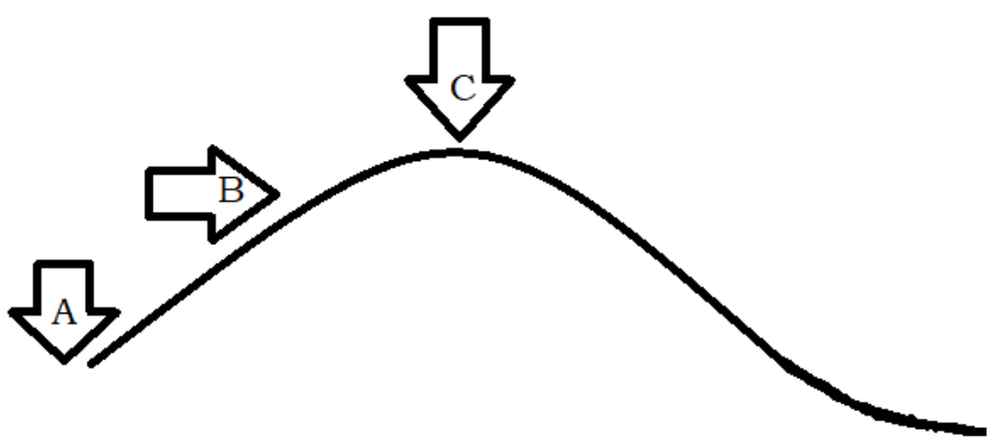
Closed List
S
A
D
F
B
C

A* Search

- heuristic has a significant impact on the performance of A*.
- common heuristics including manhattan distance, euclidean distance
- guaranteed to find the optimal solution if an admissible heuristic is used, i.e., never overestimate the cost
- efficient since search paths that appear to be more promising
- popular in many domains, including AI for games and robotics, due to its performance and accuracy
  - for navigation and avoiding obstacles where robots need to find an efficient route
  - planning tasks, planning optimal travel routes or delivery paths

Hill Climbing

- Hill Climbing iteratively improves a single solution based on neighboring solutions
  - a local search technique that aims to find the optimal solution by making small, incremental changes to a candidate solution and evaluating whether the new solution is better
- Greedy approach: moves to the best neighboring solution that improves the current state.
- use a heuristic or objective function to evaluate the solution
- well-suited for optimizing over surfaces with only one maximum

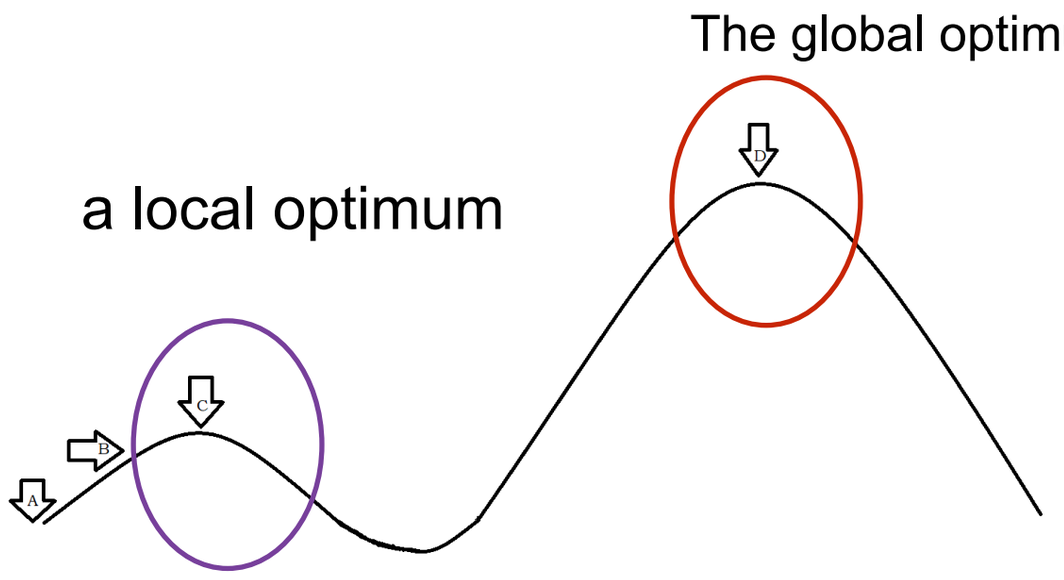


A->B->C

Hill Climbing algorithm

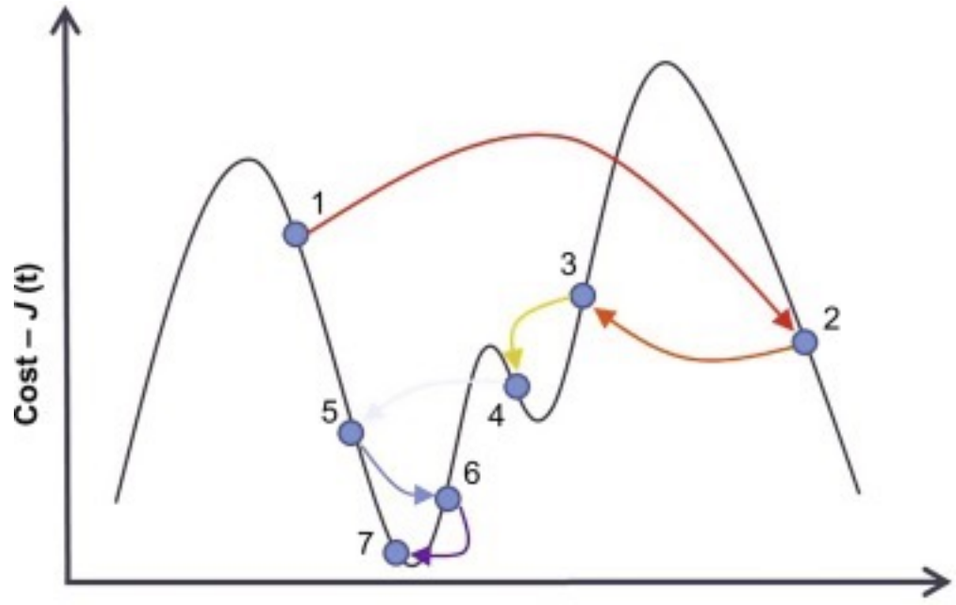
- Initilaisation: start with an initial solution, set this solution as the current solution
- Generate Neighbors: generate neighboring solutions by making small changes to the current solution
- Evaluate Neighbors: evaluate the neighboring solutions using the objective function
- Move or Stop:
  - if a better neighboring solution exists, move to that solution
  - stops if no better neighbor, the current solution is considered a local optimum

Local optima:



Simulated Annealing

- iteratively improves the current solution by randomly perturbing it and accepting the perturbation with a certain probability
  - based on the annealing process in metallurgy
  - explore the solution space by probabilistically accepting worse solutions
  - main advantage - escape from local minima and converge to a global minimum
  - the probability of accepting a new solution depends on a temperature parameter and the difference in cost (or energy) between solutions. The temperature parameter decreases over time

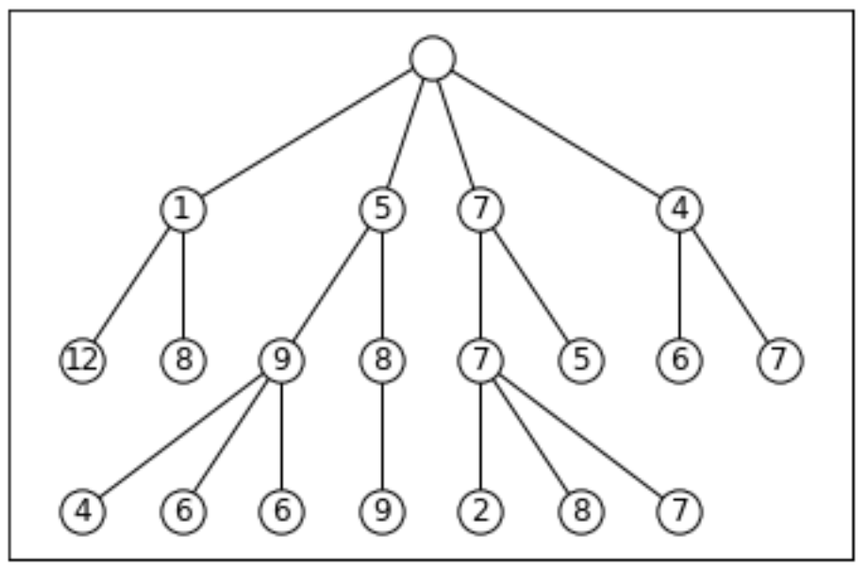


Simulated Annealing algorithm

- Initialization: define an initial solution and an initial temperature T, set the cooling schedule parameters (cooling factor and final temperature)
- Iterative Improvement:
  - While the stopping criterion is not met:
    - generate a neighboring solution
    - calculate the change in the objective function (ΔE) between the current and neighboring solutions
    - if the new solution is better (ΔE<0), move to the new solution.
    - if the new solution is worse ($\Delta E \geq 0$), accept the solution with a probability
- Cooling Schedule: Gradually reduce the temperature according to the cooling schedule

Beam search

- extend breadth-first search, generate all possible successor states and keep only the "best" k candidates, i.e., "beam"
  - reduce the search space significantly while still allowing a broad exploration of paths
  - the beam width k is configurable
  - applications in robot navigating, a maps app searching for the best route, game-playing systems



Beam search

- Initialization: define the beam width w, initialize a list of active candidates (beam) with the initial node
- Expand Candidates:
  - For each node in the beam:
    - generate all possible successors
    - evaluate each successor using a heuristic function
- Select Best Candidates: sort all successors based on their heuristic scores, retain only the top w successors
- Check for Goal State: if a goal state is found among the beam, return the solution, otherwise, continue expanding
- Iterate or Stop:
  - if no candidates remain in the beam or a stopping condition is met, terminate the search.
  - otherwise, repeat from Step 2

Summary

- Uninformed vs. Informed Search- Uninformed search strategies do not use any knowledge about the goal's location or nature, while informed search strategies use heuristics, thereby potentially increasing the efficiency of the search
- Each strategy has its context and application depending on the problem‘s requirements, e.g., in surfaces with local maxima and global maxima, compare with hill climbing, simulated annealing is more suitable since it avoid being trapped in local minima by allowing occasional moves in the wrong direction