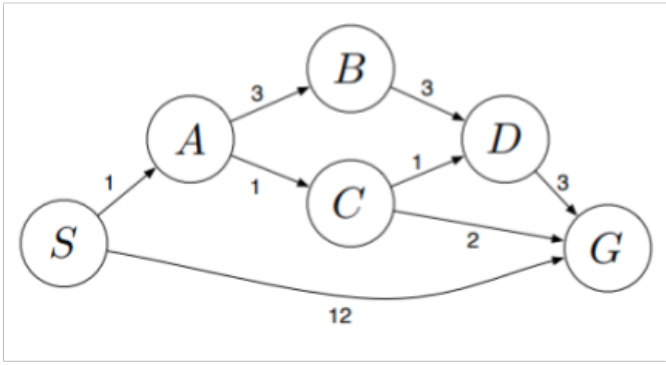
Informed Search

IFN680: Artificial Intelligence and Machine Learning
Week 9

1 Aims

In this workshop, you will implement and test some of the search algorithms introduced during the lectures, specifically informed search methods. You will do this both manually by hand and in Python.

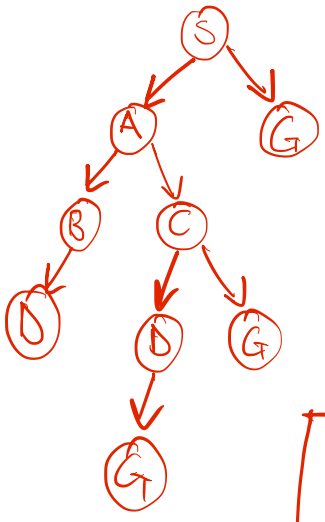
2 Informed Search Algorithms in Action



State	h_0
S	3
A	2
B	6
C	1
D	3
G	0

Answer the following questions about the search problem shown above. Break any ties alphabetically. For the questions that ask for a path, please give your answers in the form 'S - A - D - G.'

a) What path would uniform cost graph search return for this search problem?

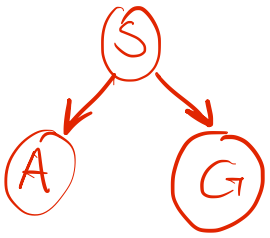


Returned Path:
S-A-C-G

Fringe Cost

~~S~~
~~S→A~~ ~~1~~
S→G 12
~~S→A→B~~ ~~4~~
~~S→A→C~~ ~~2~~
~~S→A→C→D~~ ~~3~~
S→A→C→G 4 ✓✓
S-A-C-D-G 6
S-A-B-D 7

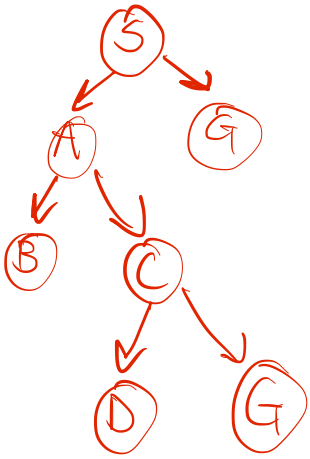
b) What path would greedy search, return for this search problem?



Fringe	Cost
~~S~~	
S-A	2
S-G	0 ✓✓

Path Returned:
S-G

b) What path would A* graph search, using the provided heuristic, return for this search problem?



Fringe	Cost
~~S~~	
~~S-A~~	~~2+1=3~~
S-G	0+12=12
S-A-B	6+4=10
~~S-A-C~~	~~1+2=3~~
S-A-C-D	3+3=6
S-A-C-G	0+4=4 ✓

Path Returned
S-A-C-G

Note how we explored less nodes than UCS.

c) Consider the heuristics for this problem shown in the table below.

State	h_1	h_2
S	5	4
A	3	2
B	6	6
C	2	1
D	3	3
G	0	0

1. Is h1 admissible? NO
2. Is h1 consistent? NO
3. Is h2 admissible? YES
4. Is h2 consistent? NO

An admissible heuristic must underestimate or be equal to the true cost.
A consistent heuristic must satisfy $h(N) - h(L) \leq \text{path}(N \to L)$ for all paths and nodes N and L.

$h1$ overestimates the cost $S \to G$ as 5 when it is 4, so it is inadmissible.
$h1$ is not consistent because $h(S) - h(A) \leq \text{path}(S \to A)$ is violated as $5 - 3 \leq 1$.
$h2$ does not overestimate costs and is admissible.
$h2$ is not consistent because $h(S) - h(A) \leq \text{path}(S \to A)$ is violated as $4 - 2 \leq 1$.

3 Pancake Problem Puzzle

In the **Pancake Problem**, you are given a stack of pancakes, each of a different size. The objective is to sort this stack so that the pancakes are in order of size, with the smallest at the top and the largest at the bottom. The only operation you are allowed to perform is to insert a spatula under any pancake in the stack and flip all the pancakes above the spatula. The challenge is to sort the stack with the minimum number of such flips. You can play with this puzzle here.

Solving the pancake puzzle involves finding a sequence of flips that achieves the desired order with the least number of flips. This is an optimization problem that can be quite difficult, particularly as the number of pancakes increases. Let's see how we can apply A* search to solve this problem:

- **State representation:** Each state can be represented by a permutation of integers representing the pancakes, with the value of the integer corresponding to the size of the pancake.
- **Initial state and goal state:** The initial state is the initial configuration of pancakes e.g `(2,0,1,3,4)` The goal state is a sorted array of integerse.g `(4,3,2,1,0)`
- **Actions:** An action is a flip, defined by the position of the spatula. If the stack has n pancakes, there are n-1 possible flips.
- **Transition model:** The transition model describes the result of performing an action (a flip) on a state (a pancake configuration). This can be defined by a function that takes a state and an action, and returns a new state.
- **Cost function:** The cost function g(n) is the number of flips taken to reach the current state from the initial state. Each flip has a cost of 1.
- **Heuristic function:** The heuristic function h(n) is an estimate of the number of flips required to sort the pancakes from the current state. There are different ways to define this heuristic. A simple example is the number of pancakes that are not in their correct position or the largest pancake that is out of place.

The aim of this challenge is to design and implement an informed AI search agent that can solve the Pancake Problem.

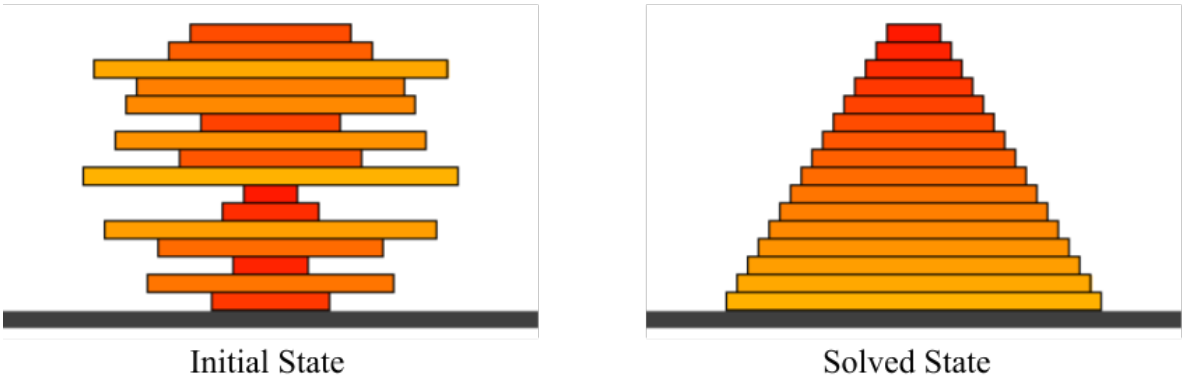


Figure 1: Pancake Problem Puzzle.

3.1 Task

Complete the missing functions in the `pancake_puzzle.ipynb` which uses functions derived from the `search.Problem` class to solve the "pancake puzzle" introduced in the lecture. Implement different search heuristic functions `h` for this problem and compare their performance.

Additionally, you must compare the number of steps taken by BFS and A* search.