AI Searching Techniques

Informed Search Techniques

Dr.Sherif Saad

Learning Objectives

- Introduce the students to different search techniques and algorithms
- Present a general approach to model and represent problems as search problems
- Demonstrate how to implement AI search techniques and algorithms

Last-Time

- Introduction to Informed Search Techniques
- Design Heuristics Functions (Heuristics Engineering)
- Best-First Search

Outlines

- A* Search Algorithm
- Greedy Algorithm
- Travel Salesman Problem

A* Search Algorithm: Description

Is an informed search algorithm belongs to the Best-First-Search Algorithms Family.

Introduced in 1968 by Peter Hart and Colleagues from Stanford as an extension of Dijkstra's Algorithm

The A* algorithm expand the node that minimize the cost to the goal. In other words select the paths that appear to lead most quickly to the solution.

A* is complete and will always find a solution, if one exists. (any conditions??)

A* Search Algorithm: Definition

The A* algorithm requires the following:

- Start node S and end Node G
- An admissible heuristic h

The A* algorithm will always select the path that minimize:

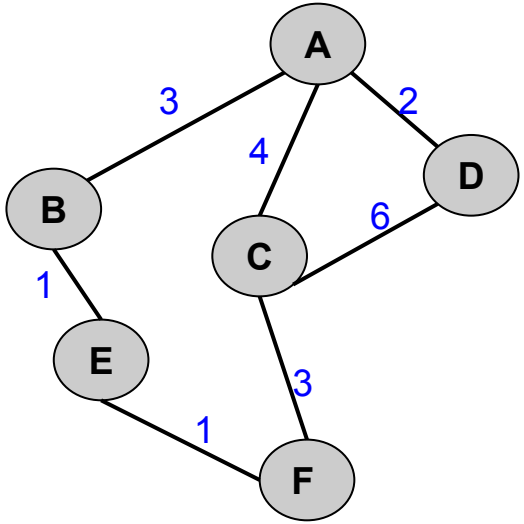
$$f(n) = g(n) + h(n)$$

- ***f(n):*** *Evaluation function that measure the cost of expanding* ***n***
- ***g(n):*** *The actual cost we spent to reach* ***n***
- ***h(n):*** *heuristic function that measure the cost of reaching the goal from* ***n***

A* Search Algorithm: Example (1)

Apply the A* Algorithm to find the shortest path from A to F given the h(n) of each node in the graph

state	H(n)
A	6
B	2
C	2
D	7
E	1
F	0

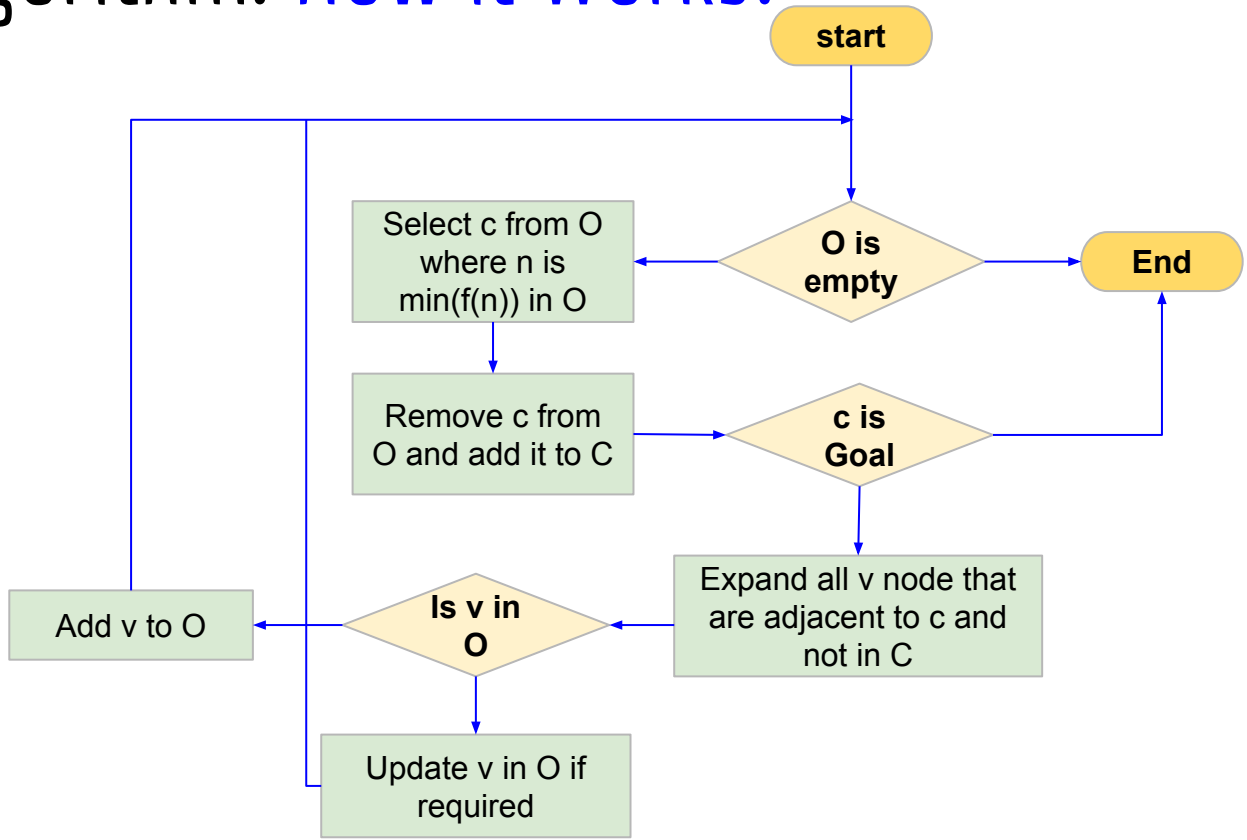


A* Search Algorithm: How it Works?

A * use 2 lists or a priority queue and a list.

O: to store nodes for expansions

C: to store node which already explored or visited



A* Search: Pseudo Code

```
def a_start_search(G,s,t,h):
    # Let Q an empty Q
    Q = Queue()
    f(s)= g(s)+h(s)
    Q.enque((s,f(s))
    while Q.empty is False:
        c = Q.pop()
        if c == t:
          return
        for w in neighbours of c in G:
            f(w) = g(w) + h(w)
            if w in Q:
                Q.update((w,f(w)))
            else:
                Q.enque((w,f(w)))
```

When A* terminate?

What is the difference between A* Search and Cheapest First Search?

Why is Cheapest First Search **not** Informed Search?

A* Search: 8 Puzzle Example

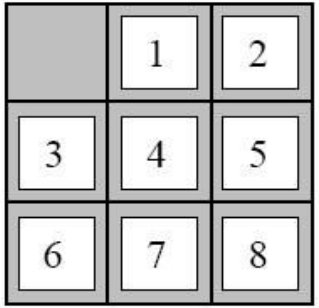
Use A* search to solve this 8 Puzzle problem.

Apply the two heuristics we discussed in the previous class.

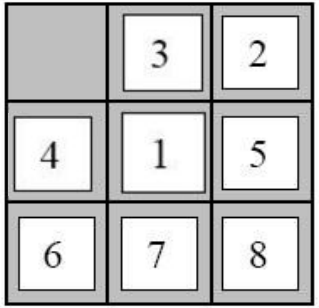
Which heuristic is better and why?

What could be a better heuristic?

When do you think A * would become optimal?



Start State



Goal State

A* Search: Application Examples

- PathFinding (video game, cybersecurity, defense)
- Speech Recognition
- NLP and Machine Translation
- Robot and Droid navigation

Greedy Search Algorithm: Definition

What is Greedy Search Algorithm?

A problem solving approach that solve the problem by making locally optimal choice at each stage with the hope of finding a global optimum.

It is an optimistic approach for problem solving.

Making the best choice at each step using the available information.

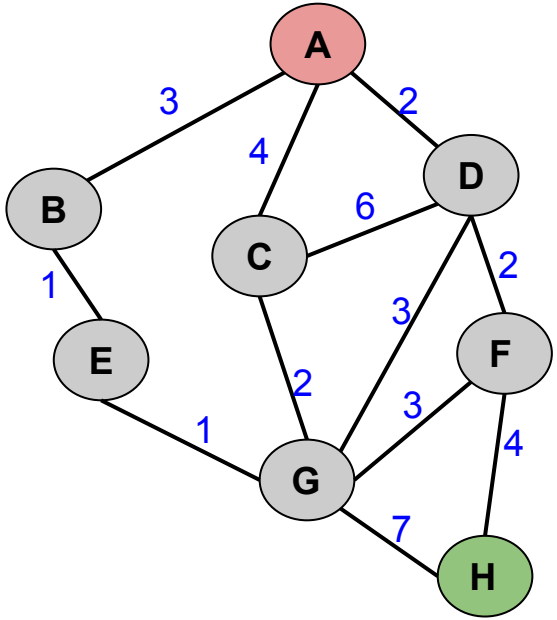
Will never reconsider a decision once made.

Greedy Search Algorithm: Example

Using greedy search find the path from node A to node H?

How is greedy search is different from A* and Cheapest Search First Algorithm?

What are the advantages of greedy search?



Greedy Search Algorithm: Main Idea

- I wish
  - I had some way to order my choices by value
  - I had some way of making the next choice.
  - greed works for my problem
- Advantages:
  - Greed is easy to implement
  - Less thinking and (cheaper), time complexity O(n^2)
- Disadvantages:
  - Design a correct greedy solution is hard
  - Verifying the the correctness of greedy is hard
  - Doesn't work for all the problems
  - *Greed is a sin** (do not do it unless you are sure you will win)

Greedy Search Algorithm: Example 2

Things to do in Vancouver. You are visiting Vancouver for one day only here is the list of the places you can visit in Vancouver during this day and the time it will take you to visit each place.
You only have 8 hours and your objective is to visit as many places as you can

Things to do	Time = 8 hours
Stanley Park	2.5 hours
Capilano Suspension Bridge	1.5 hours
Vancouver Aquarium	3 hours
Science World	2 hours
Canada Place	1 hours
English Bay	½ hours
Grouse Mountain	3 hours

Greedy Search Algorithm: Vancouver Visit

Design a greedy algorithm to maximize the number of things you can do during your visit.

Greedy Solution Outlines:

- Sort the the things you can do in an ascending order based on the time.
- Pick an item from the sorted list and do it and mark it as done
- Update your available time based on the items you marked as done
- Repeat the above steps as long as the available time is bigger that the time of the item at the head of the list

Greedy Search Algorithm: Vancouver Visit

Is the proposed greedy solution complete?

Does it find the optimal solution?

What if we add pleasure value to each place, and we want to maximize the pleasure gain based on the available time

Things to do	Time = 8 hours
Stanley Park	2.5 hours
Capilano Suspension Bridge	1.5 hours
Vancouver Aquarium	3 hours
Science World	2 hours
Canada Place	1 hours
English Bay	½ hours
Grouse Mountain	3 hours

Greedy Search Algorithm: Vancouver Visit

Use a greedy search algorithm to solve the vancouver visit considering a pleasure where you want to maximize the number of places you can visit and gain the maximum pleasure

Things to do	Time	Pleasure
Stanley Park	2.5	4
Capilano Suspension Bridge	1.5	1.5
Vancouver Aquarium	3	2
Science World	2	5
Canada Place	1	3
English Bay	0.5	1
Grouse Mountain	3	5

Greedy Search Algorithm: Vancouver Visit

Use the pleasure per unit of time as your heuristic function

Sort the places based on the heuristic measure in descending order

Then visit the places on the sorted list in order

Things to do	Time	Pleasure	Pleasure/time
Stanley Park	2.5	4	1.6
Capilano Bridge	1.5	1.5	1.0
Vancouver Aquarium	3	2	0.66
Science World	2	5	2.5
Canada Place	1	3	3
English Bay	0.5	1	2
Grouse Mountain	3	5	1.6

Greedy Search Algorithm: When??

Works very well in the following for problems with the following properties:

Optimal Substructures: the problem can be divided into similar sub problems and the global solution can be reconstructed efficiently form optimal solutions of the sub-problmes.

Greedy Property Exist: at any point you are making a choice that seems to be the best at this point. You will never have to reconsider your earlier choices

Note: no guaranteed way to recognize problems that can be solved by a greedy algorithm

Self Check Exercise: Travel Salesman Problem

A salesman who must travel between N cities. The order in which he does so is something he does not care about, as long as he visits each one during his trip, and finishes where he was at first. Each city is connected to other close by cities, or nodes, by airplanes, or by road or railway. Each of those links between the cities has one or more weights (or the cost) attached. The cost describes how "difficult" it is to traverse this edge on the graph, and may be given, for example, by the cost of an airplane ticket or train ticket, or perhaps by the length of the edge, or time required to complete the traversal. The salesman wants to keep both the travel costs, as well as the distance he travels as low as possible.

Self Check Exercise: Travel Salesman Problem

For the TSP, answer the following:

1. Formalize the problem as a search problem
    a. States, Initial State, Goal Test, Cost Function, etc
2. Design a heuristic search algorithm to solve this problem

Questions