Informed Search

Yingyu Liang

yliang@cs.wisc.edu

Computer Sciences Department

University of Wisconsin, Madison

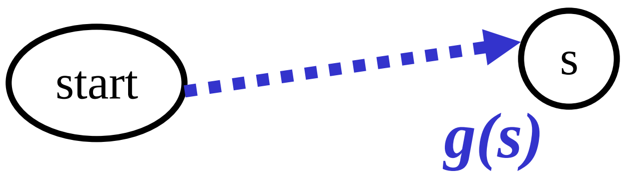
[Based on slides from Jerry Zhu, Andrew Moore http://www.cs.cmu.edu/~awm/tutorials]

Main messages

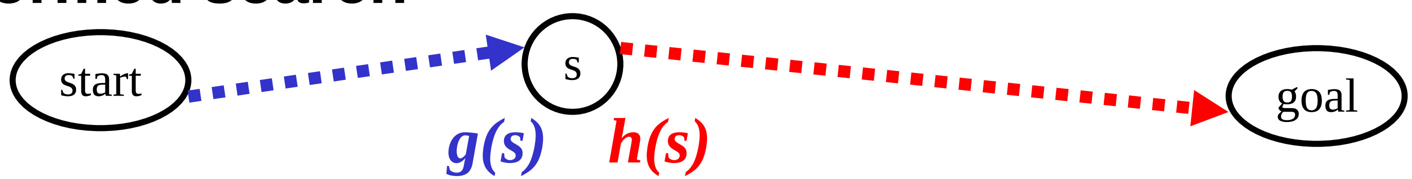
- A*. Always be optimistic.

Uninformed vs. informed search

- **Uninformed search** (BFS, uniform-cost, DFS, ID etc.)
  - Knows the actual path cost ***g(s)*** from start to a node ***s*** in the fringe, but that’s it.



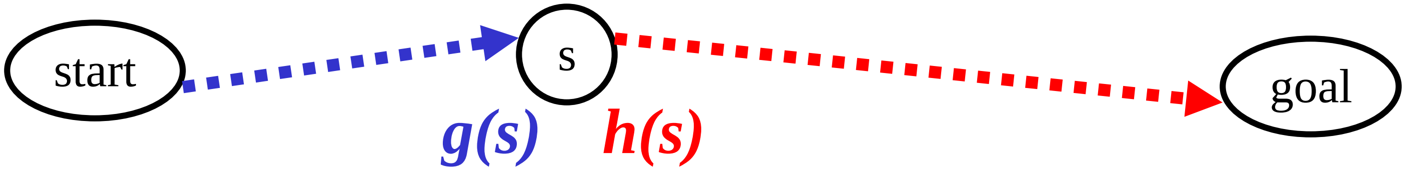
- **Informed search**



  - also has a heuristic ***h(s)*** of the cost from ***s*** to goal. (‘h’= heuristic, non-negative)
  - Can be much faster than uninformed search.

Recall: Uniform-cost search

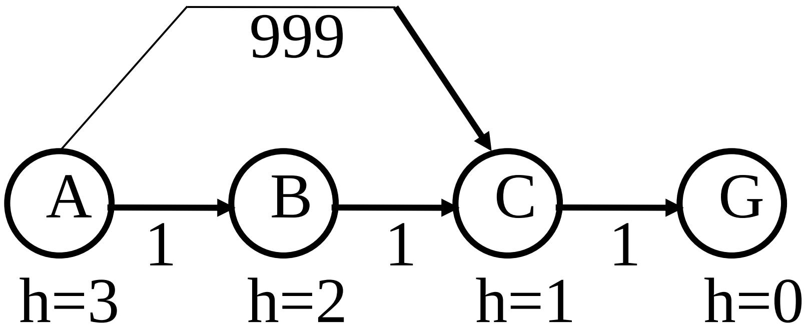
- Uniform-cost search: uninformed search when edge costs are not the same.
- Complete (will find a goal). Optimal (will find the least-cost goal).
- Always expand the node with the least ***g(s)***
  - Use a priority queue:
    - Push in states with their first-half-cost ***g(s)***
    - Pop out the state with the least ***g(s)*** first.
- Now we have an estimate of the second-half-cost ***h(s)***, how to use it?



First attempt: Best-first greedy search

- Idea 1: use ***h(s)*** instead of ***g(s)***
- Always expand the node with the least ***h(s)***
  - Use a priority queue:
    - Push in states with their second-half-cost ***h(s)***
    - Pop out the state with the least ***h(s)*** first.
- Known as “best first greedy” search
- How’s this idea?

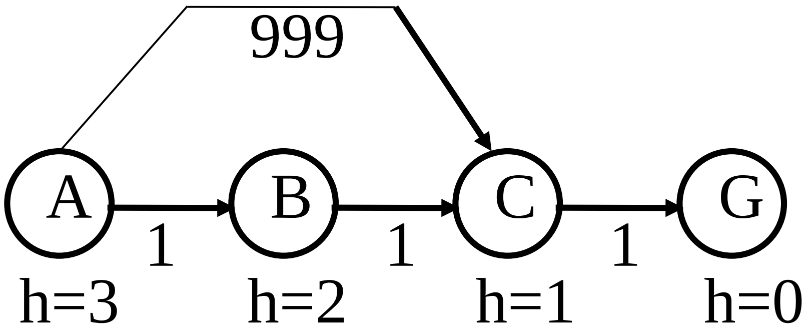
Best-first greedy search looking stupid



- It will follow the path *A*→*C*→*G* (why?)
- Obviously not optimal

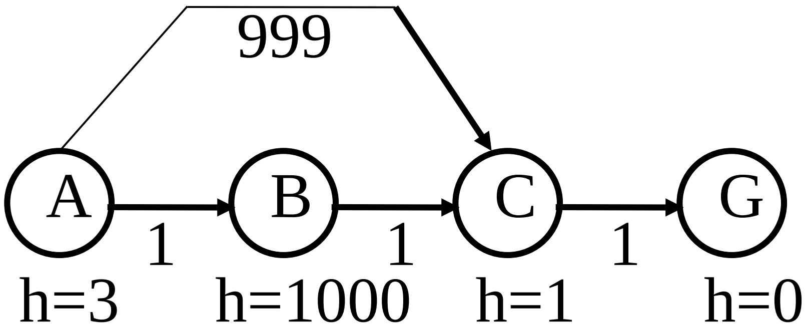
Second attempt: A search

- Idea 2: use ***g(s)***+***h(s)***
- Always expand the node with the least ***g(s)***+***h(s)***
  - Use a priority queue:
    - Push in states with their first-half-cost ***g(s)***+***h(s)***
    - Pop out the state with the least ***g(s)***+***h(s)*** first.
- Known as "A" search
- How's this idea?



- Works for this example

A search still not quite right



- A search is not optimal.

Third attempt: A* search

- Same as A search, but the heuristic function ***h()*** has to satisfy $\boldsymbol{h(s)} \le h^*(s)$, where $h^*(s)$ is the true cost from node *s* to the goal.
- Such heuristic function ***h()*** is called admissible.
  - An admissible heuristic never over-estimates

It is always optimistic

- A search with admissible ***h()*** is called A* search.

Admissible heuristic functions *h*

- 8-puzzle example

Example State

1		5
2	6	3
7	4	8

Goal State

1	2	3
4	5	6
7	8	

- Which of the following are admissible heuristics?

  •h(n)=number of tiles in wrong position
  •h(n)=0
  •h(n)=1
  •h(n)=sum of Manhattan distance between each tile and its goal location

Admissible heuristic functions *h*

- In general, which of the following are admissible heuristics? h*(n) is the true optimal cost from n to goal.
    - h(n)=h*(n)
    - h(n)=max(2,h*(n))
    - h(n)=min(2,h*(n))
    - h(n)=h*(n)-2
    - h(n)=sqrt(h*(n))

Heuristics for Admissible heuristics

- How to construct heuristic functions?

Example State

1		5
2	6	3
7	4	8

Goal State

1	2	3
4	5	6
7	8	

- Often by relaxing the constraints

  •h(n)=number of tiles in wrong position

    Allow tiles to fly to their destination in one step

  •h(n)=sum of Manhattan distance between each tile and its goal location

    Allow tiles to move on top of other tiles

“my heuristic is better than yours”

- A heuristic function h2 **dominates** h1 if for all s

  $h1(s) \le h2(s) \le h^*(s)$

- We prefer heuristic functions as close to h* as possible, but not over h*.

But

- Good heuristic function might need complex computation
- Time may be better spent, if we use a faster, simpler heuristic function and expand more nodes