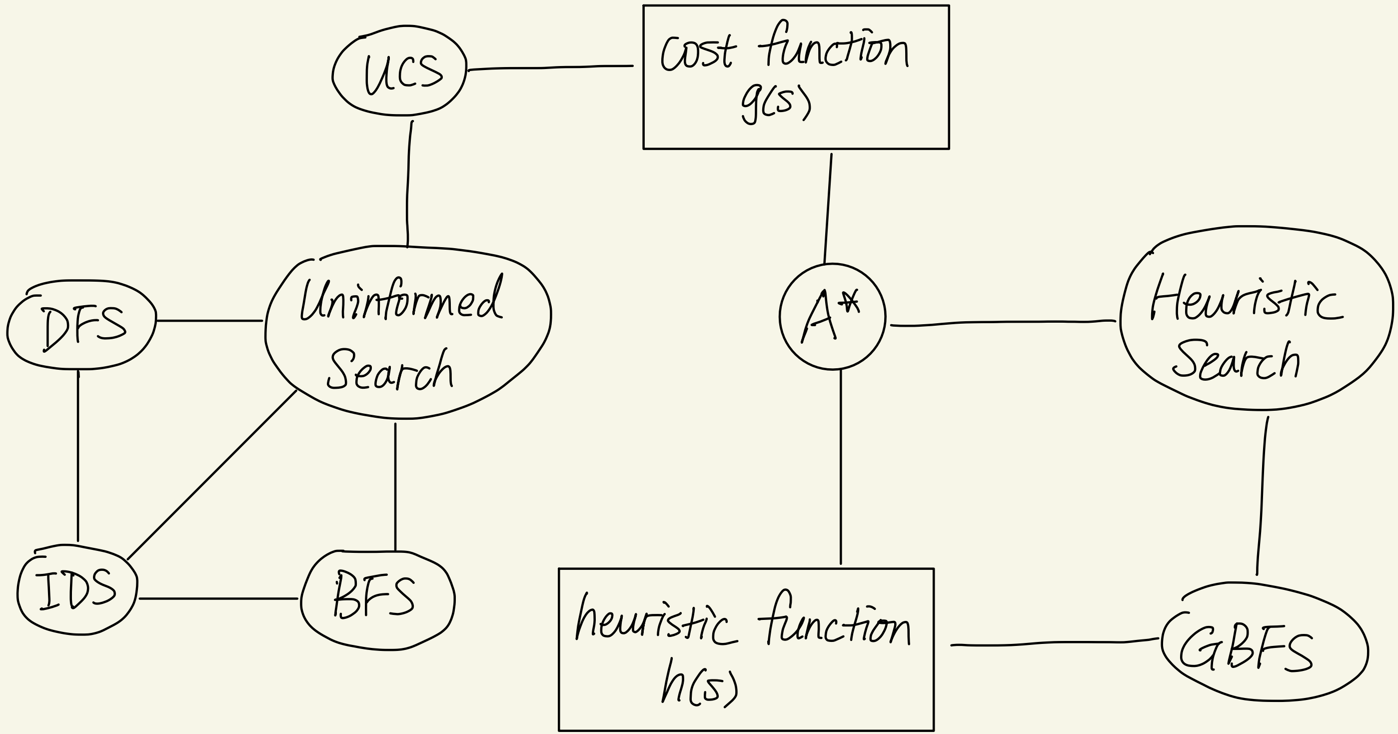
Search

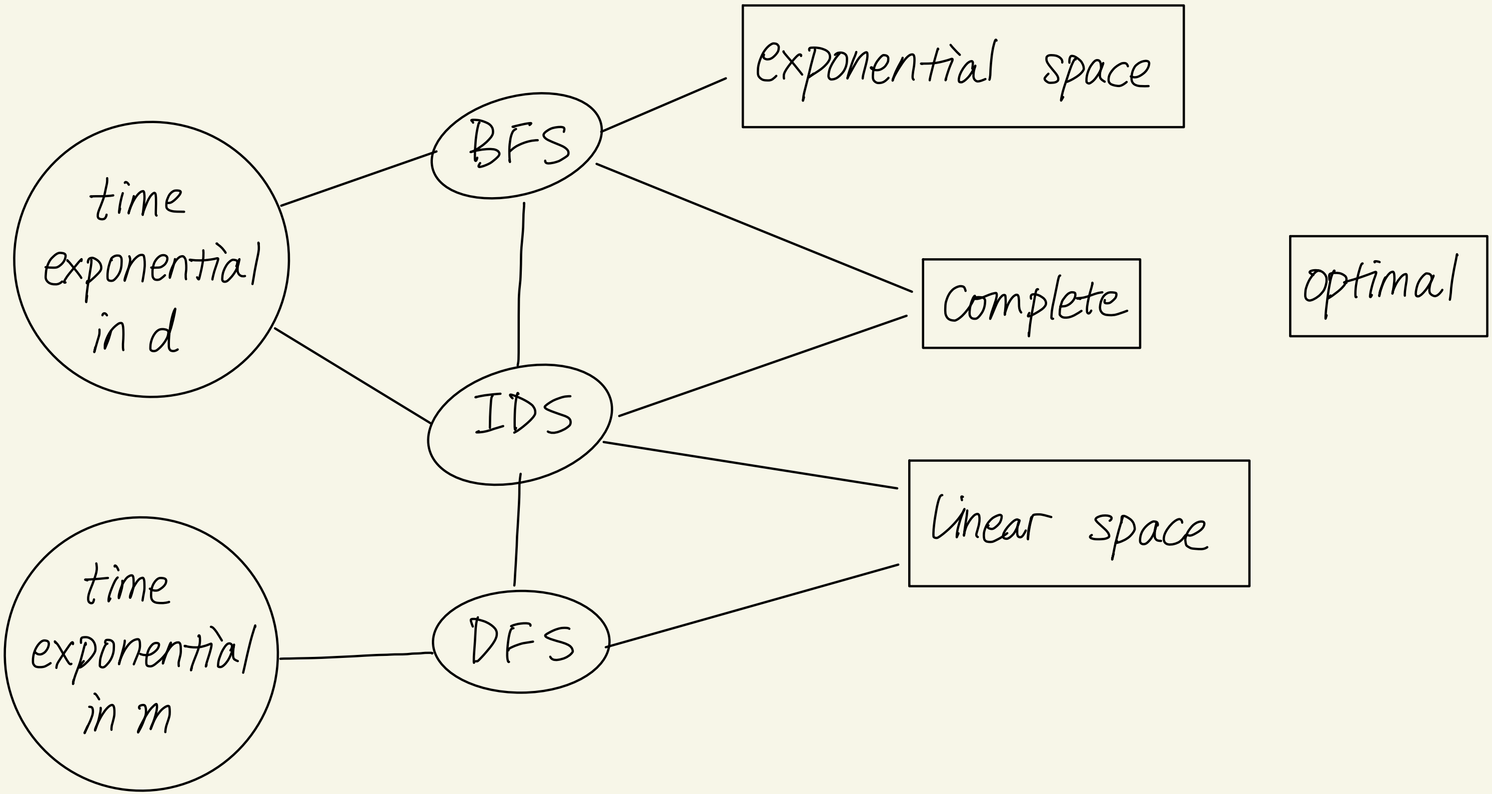
Summary and Questions

by Alice Gao ^_^

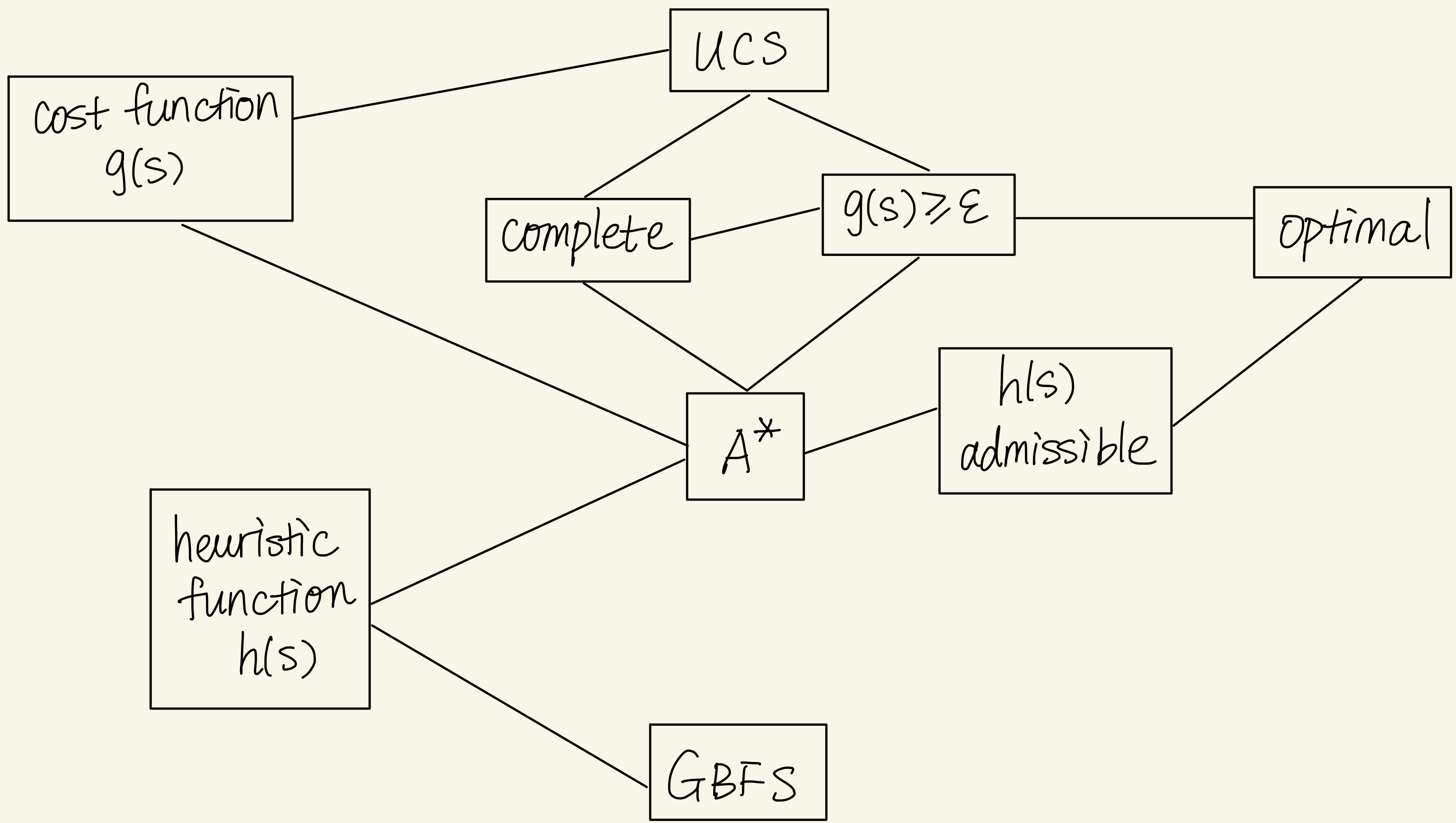
Uninformed v.s. Heuristic Search



Properties of BFS, DFS, and IDS



Properties of UCS, GBFS, A* Search.



Uninformed Search.

(Blind).

Depth - First Search.

- strategy : frontier is a stack (LIFO)
- complete ? No.

  stuck on infinite paths.
- optimal ? No.

  not optimizing costs.
- space : linear in m.

  remembers one path.
- time : exponential in m.

  visits the entire tree in the worst case.

Breadth-First Search.

- strategy: removes oldest state added.
- complete? Yes.

  guaranteed to find shallowest goal node.
- optimal? No.

  not optimizing costs.
- space: exponential in d.

  # of nodes on level d is exponential in d.
- time: exponential in d.

  visits top d levels in the worst case.

Iterative - Deepening Search.

- strategy : for depth limit l from 0 to ∞,
  perform DFS until depth limit l.
- complete ? Yes. ($\approx$ BFS)
  terminates at level d.
- optimal ? No. ($\approx$ BFS & DFS)
  not optimizing costs.
- space : linear in d. ($\approx$ DFS)
  remembers one path of length at most d.
- time : exponential in d. ($\approx$ BFS)

Multiple-Path Pruning

- why? find a solution faster.
  unnecessary to store multiple paths to same node.
- how?
  use an explored set to remember visited states.
- effects on space complexity:
  - ↑ space complexity to exponential
- effects on optimality:
  - UCS w/ multi-path pruning is still optimal.
  - A* w/ multi-path pruning may not be optimal.

Uniform-Cost Search.

- strategy: remove path w/ smallest total cost.
- complete? Yes.

  edge cost is not arbitrarily small.
- optimal? Yes.
- space: exponential in C^*/ε C^*: optimal path cost. ε: minimum edge cost.
- time:

Multi-path pruning.

why? only need to find one path to each state.
how? keep the first path found to each state.
discard all other paths found.

effects on algorithm properties.

- DFS : linear space → exponential space.
- BFS : exponential space remains the same.
- UCS : still optimal.
- A* : may not be optimal depending on $h(n)$.

Heuristic Search

Greedy Best-First Search.

- Strategy: expand state w/ smallest heuristic value.
- Complete? No.

  can get stuck in a cycle.
- optimal? No.

  can return a suboptimal solution first.
- space and time: exponential.

1. Construct a search graph s.t. GBFS does not terminate.

2. Construct a search graph s.t. GBFS does not return the optimal solution.

A* Search

- strategy: expand state w/ smallest f value.

$$f(s) = \underset{\substack{\uparrow \\ \text{cost}}}{g(s)} + \underset{\substack{\uparrow \\ \text{heuristic.}}}{h(s)}$$

- complete? Yes under mild conditions.
- optimal? Yes if $h(s)$ is admissible.
- space and time: exponential.

(A* is optimal)

1. Construct a search graph w/ inadmissible heuristic and show that A* search does not find the otimal solution.

or

(A* is optimally efficient.)

2. Construct a search graph w/ admissible heuristic and show that
   ① UCS finds the optimal solution.
   ② A* finds the optimal solution.
   ③ A* visits fewer states than UCS.

Constructing Heuristics.

The cost of the optimal solution to the relaxed problem
= an admissible heuristic for the original problem.

Dominating Heuristics.

$h_1(n)$ dominates $h_2(n)$ iff

(1) $h_1(n) \geq h_2(n) \quad \forall n.$

(2) $h_1(n) > h_2(n) \quad \exists n.$

A* with multi-path pruning

- if heuristic is admissible but not consistent, A* w/ multi-path pruning is not optimal.

Definition of Consistent Heuristic.

① $h(n)$ is admissible, and

② if there is an edge from n_1 to n_2,

$$h(n_1) - h(n_2) \le g(n_1, n_2).$$

- if heuristic is consistent, A* w/ multi-path pruning is optimal.

1. Given a search graph, verify that a heuristic function is admissible.

2. Given a search graph, verify that a heuristic function is consistent.

3. Construct a search graph and a heuristic that is admissible but not consistent, show that A* w/ multi-path pruning is not optimal.