Heuristic Search:

BestFS and A*

CPSC 322 Lecture 8

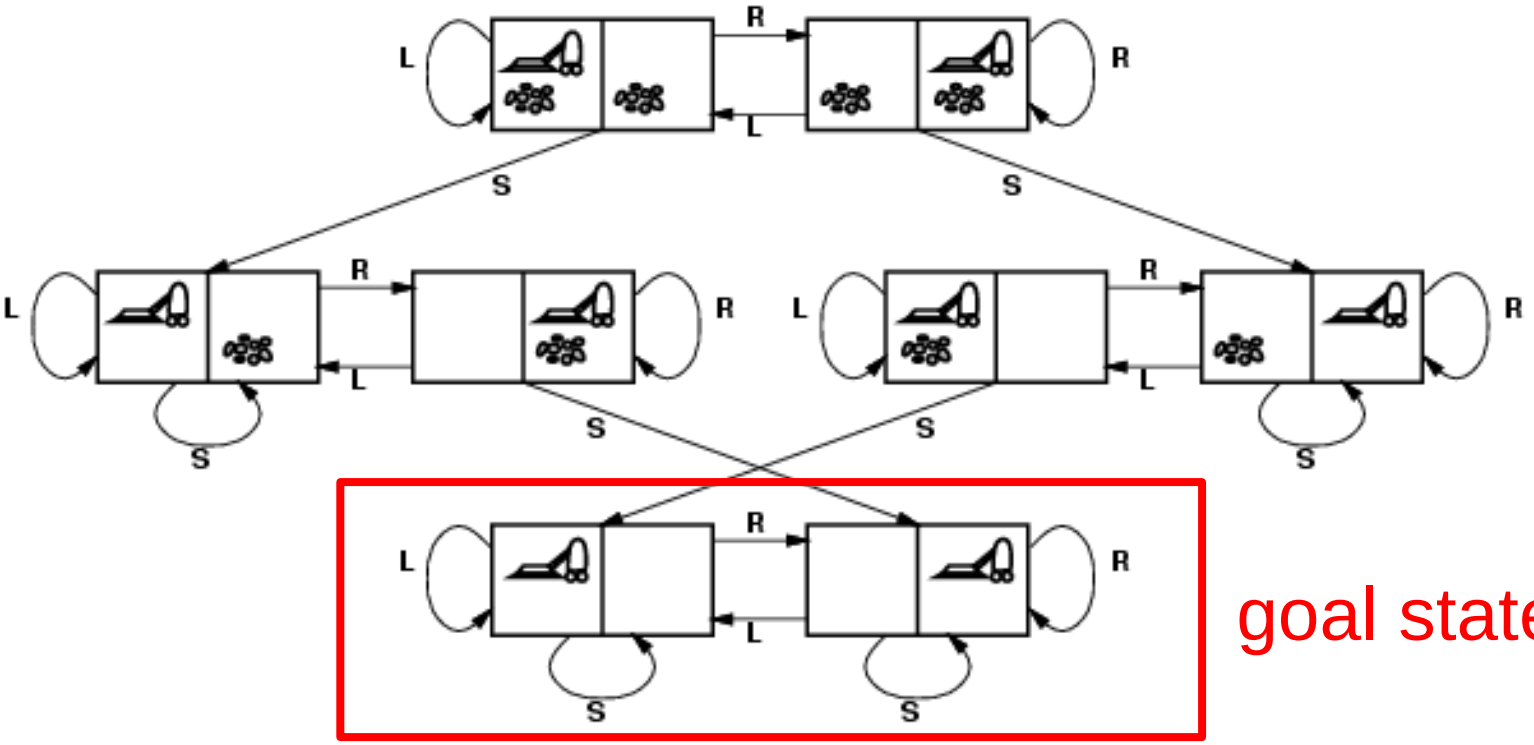
Lecture Overview

- **Recap / Finish Heuristic Function**
- **Best First Search**
- A*

Strategies for Creating Heuristics

- Calculate the solution cost for a **relaxed** version of the problem
  - eg. No walls, no movement constraints
- Use (optimal) search to find the solution cost for a **subproblem**
  - eg. 8-puzzle, but replace some numbers with blank tiles

Admissible heuristic for Vacuum world?



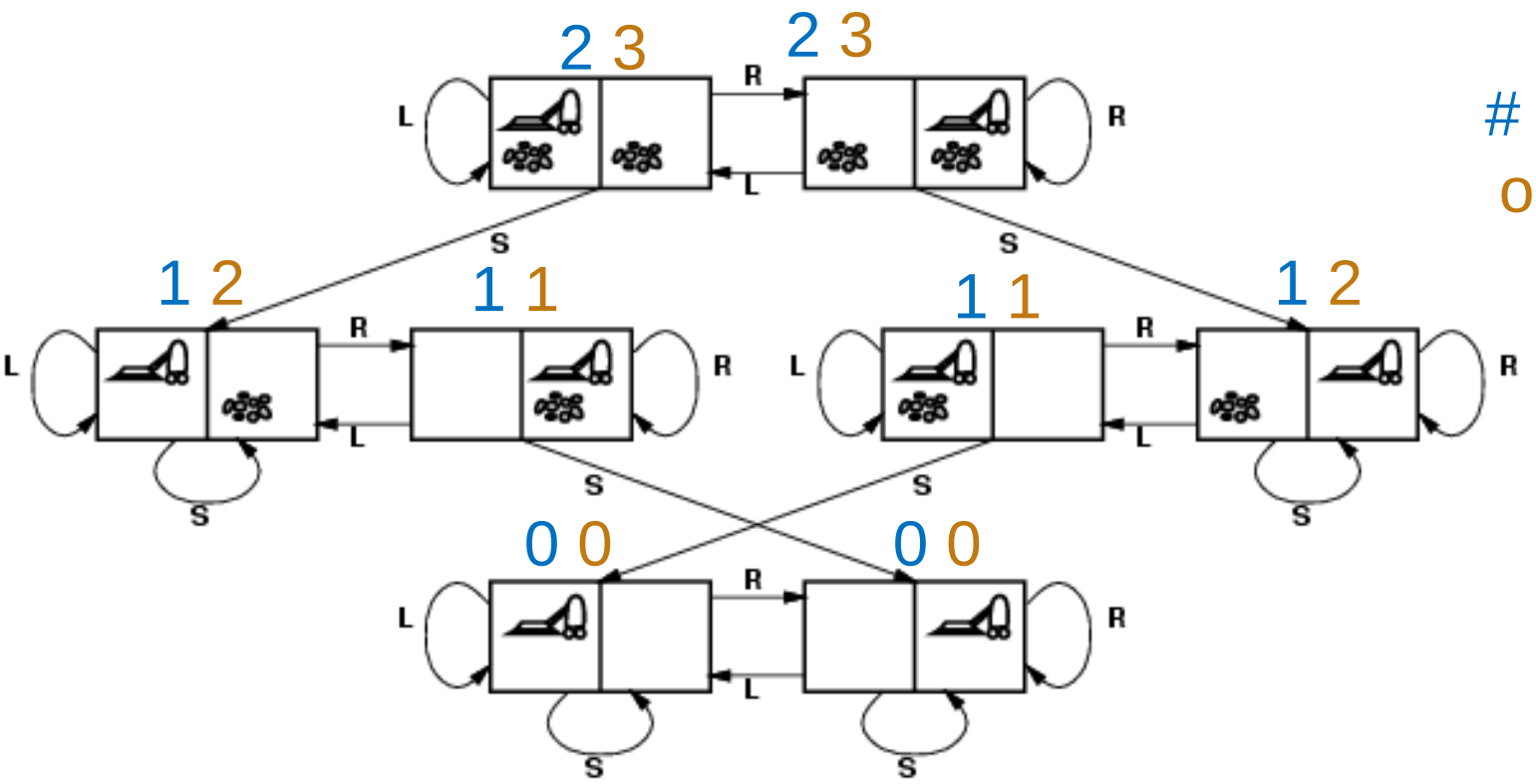
goal states

states? Where it is dirty and robot location

actions? *Left*, *Right*, *Suck*

Possible goal test? no dirt at all locations

Admissible heuristic for Vacuum world?



dirty rooms
optimal cost

states? Where it is dirty and robot location

actions? *Left*, *Right*, *Suck*

Possible goal test? no dirt at all locations

Learning Goals for today's class

- **Define/read/write/trace/debug** & **Compare** different search algorithms
  - With / Without cost
  - Informed / Uninformed

- Formally prove A* optimality.

Lecture Overview

- Recap Heuristic Function
- **Best First Search**
- A*

Best-First Search

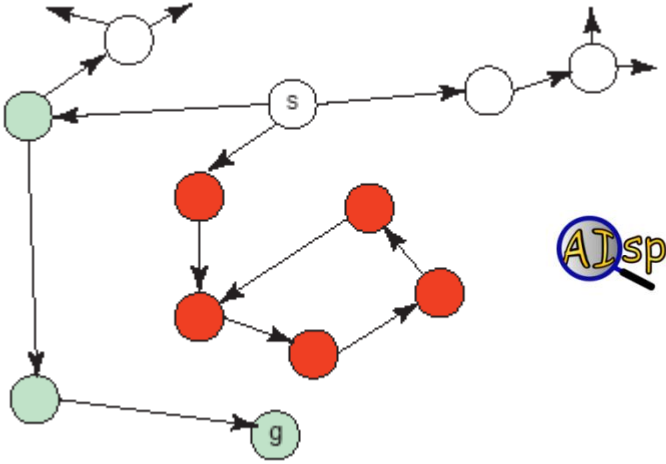
- **Idea:** select the path whose end is closest to a goal according to the heuristic function.

- **Best-First search** selects a path on the frontier with minimal *h*-value (for the end node).

- It treats the frontier as a priority queue ordered by *h*. (similar to _____)
- This is a greedy approach: it always takes the path which appears locally best

Analysis of Best-First Search

- Not Complete : low heuristic values in a cycle can mean that the cycle gets followed forever.



(sample: misleading heuristic)

- Optimal: no (why not?)
- Time complexity is $O(b^m)$
- Space complexity is $O(b^m)$

(ex4 from course website)

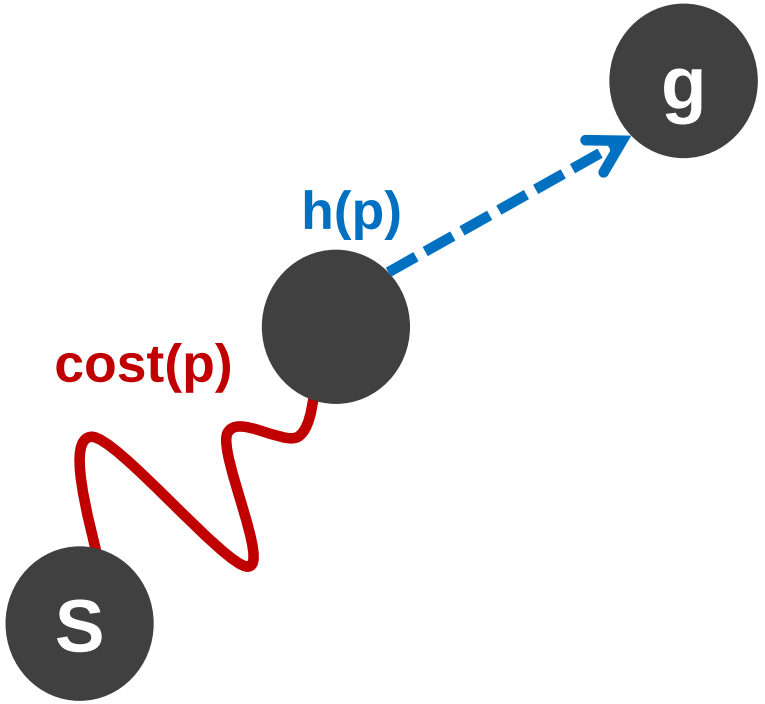
Lecture Overview

- Recap Heuristic Function
- Best First Search
- A* Search Strategy

A*: "Mixing" LCFS and BestFS

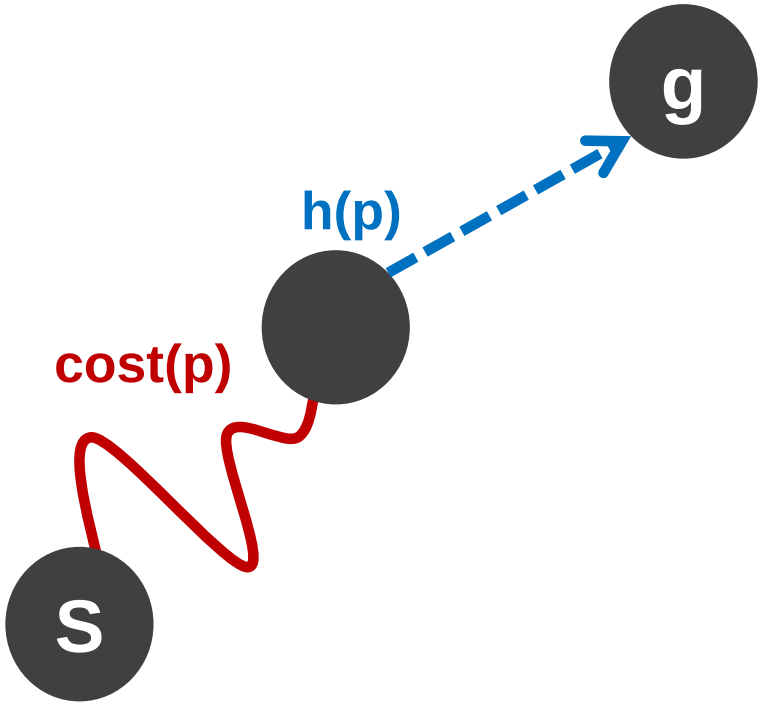
- **LCFS** uses the **cost** of a path ***p***
- **BestFS** uses ***h(p)*** from the end of a path ***p***
- Could we use **both**? If so, how?

A. Lowest cost(p) – h(p)
B. Highest cost(p) – h(p)
C. Highest cost(p) + h(p)
D. Lowest cost(p) + h(p)



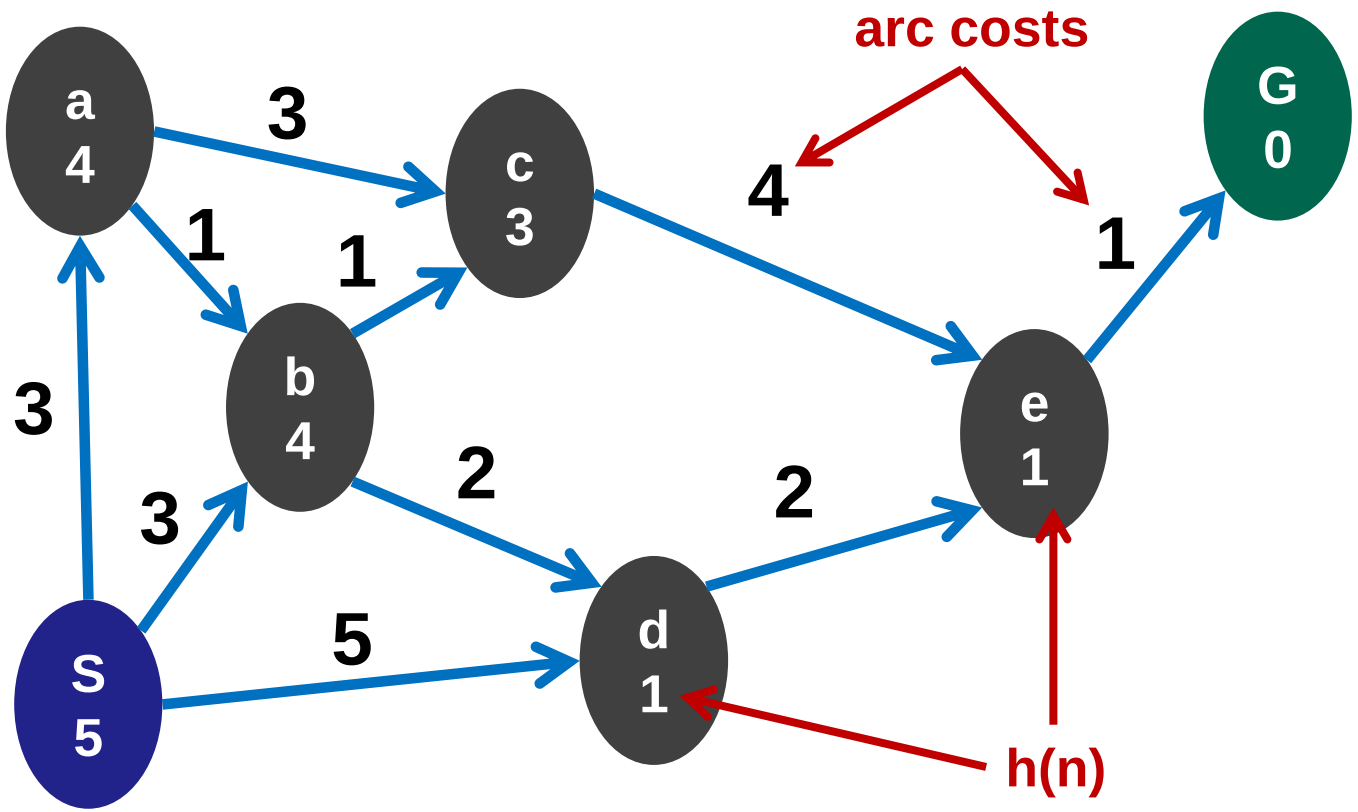
A*: “Mixing” LCFS and BestFS

- A^* is a mix of:
  - **lowest-cost-first** and
  - **best-first search**



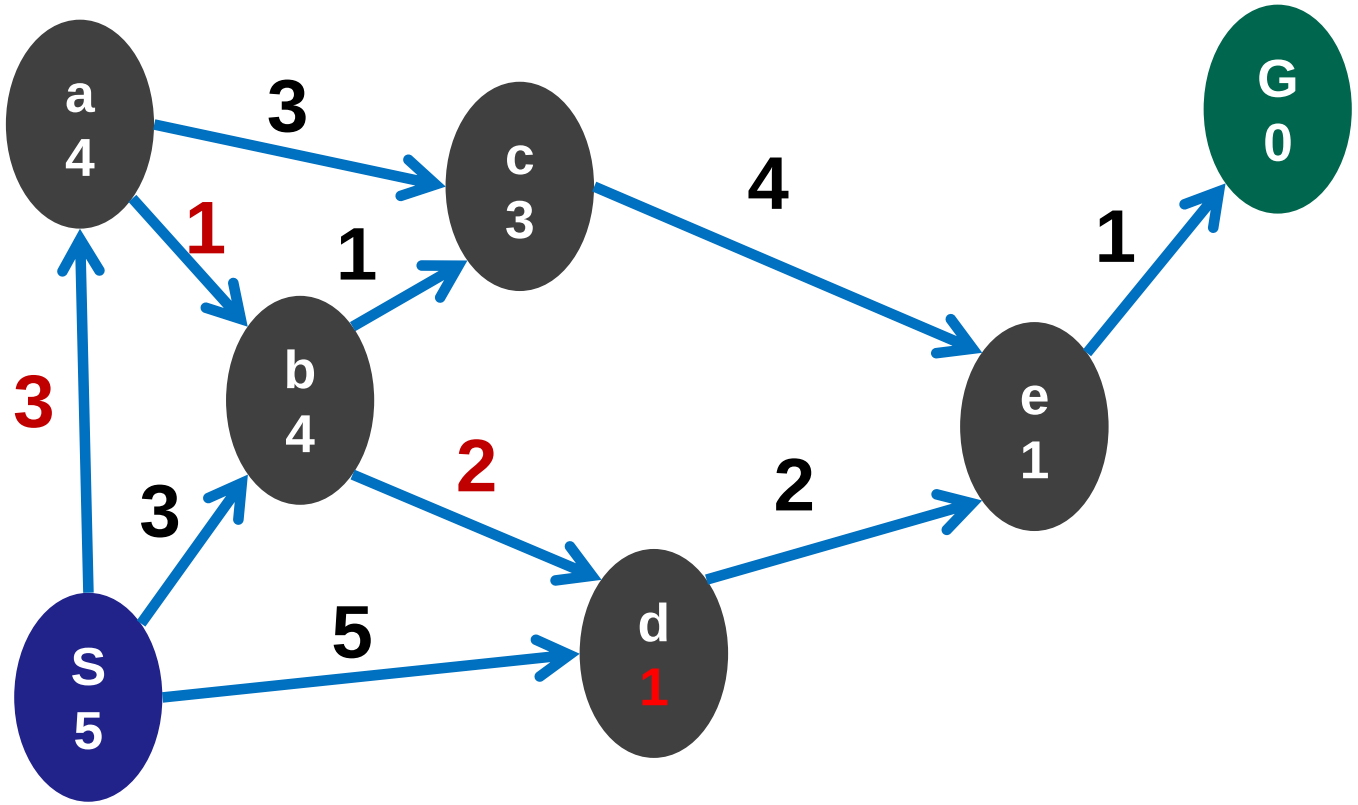
- A^* treats the frontier as a **priority queue** ordered by ***f(p)***=
- It always selects the path on the frontier with the **lowest** estimated **total** distance to a goal.

A*: Computing f-values



What is the f-value of S → a → b → d?

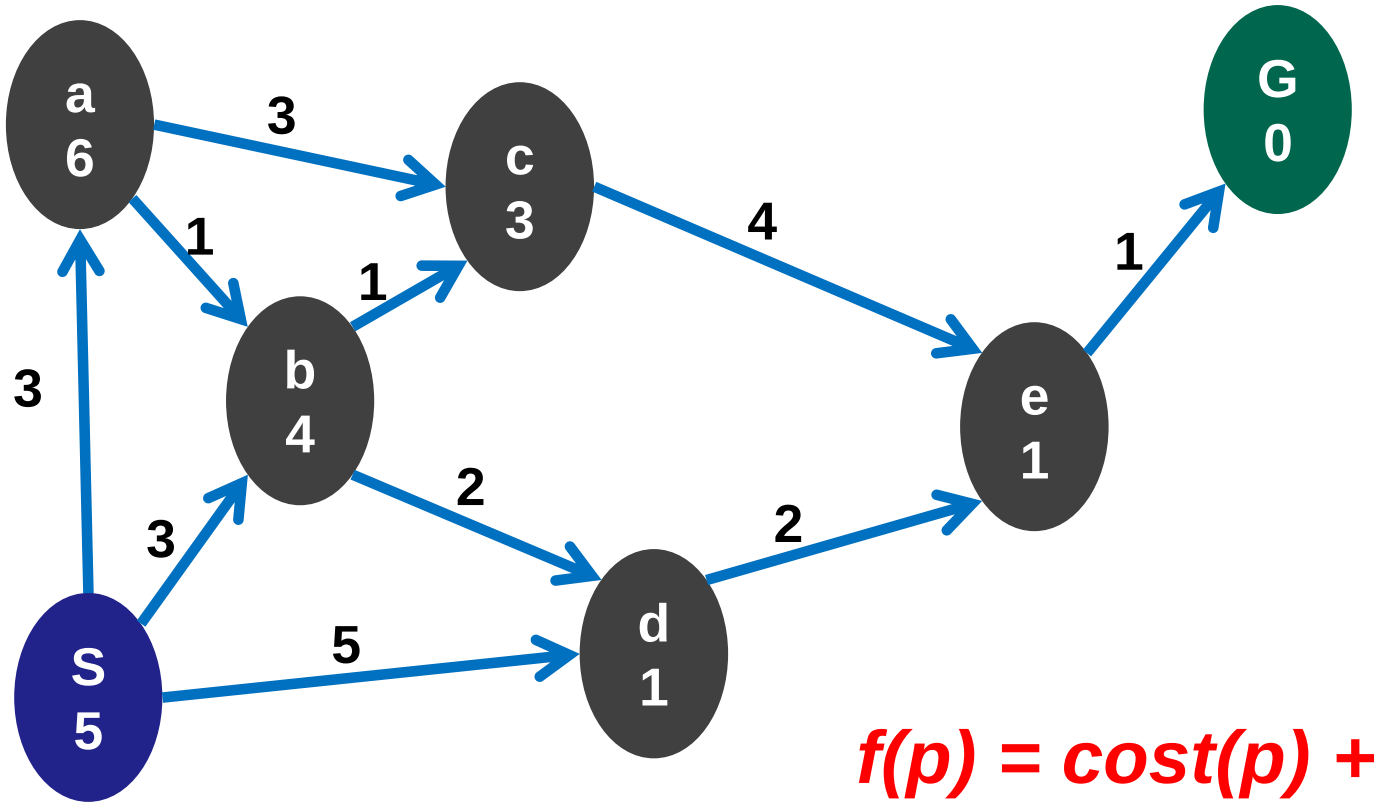
A*: Computing f-values



cost(<S,a,b,d>) h(d)

What is the f-value of S → a → b → d? (3+1+2) + 1 = **7**

A*: Sample problem (Group activity)



f(p) = cost(p) + h(p)

Trace through A* on this graph. Break ties alphabetically.

- What is the order of visited nodes?
- What is the solution path, and what is its cost?

Analysis of A*

If the heuristic is completely uninformative *(eg. h=0 everywhere)* and the edge costs are all the same, A* is equivalent to….

A. BFS
B. LCFS
C. DFS
D. A and B
E. B and C

Analysis of A^*

Let's assume that arc costs are strictly positive. The heuristic could be completely uninformative, and the edge costs could all be the same, meaning that A* would do the same thing as _____. SO:

- Time complexity: $O(b^m)$
- Space complexity: $O(b^m)$
- Completeness: YES
- Optimality: **??**

Optimality of A^*

If A^* returns a solution, that solution is guaranteed to be optimal, as long as

- the branching factor is finite
- arc costs are strictly positive
- *h(n)* is an underestimate of the length of the shortest path from *n* to a goal node (i.e. is **admissible**), and is non-negative

Theorem

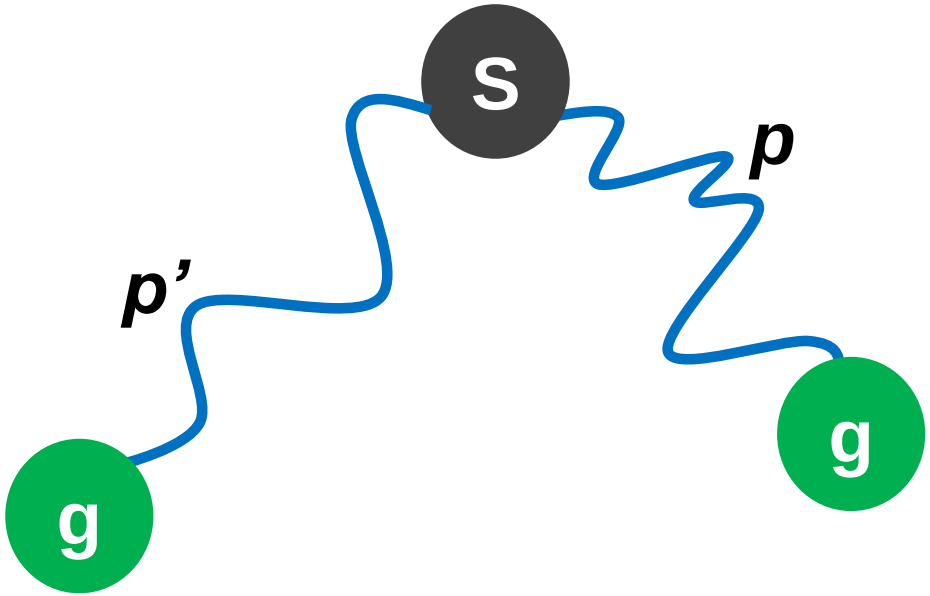
If A^* selects a path *p* as the solution, then *p* is an optimal (i.e., lowest-cost) path.

Proof of Optimality of A*

Suppose A* returns path ***p***

Proof by contradiction:

Assume that there exists some other path ***p'*** that is a "better" path to a goal

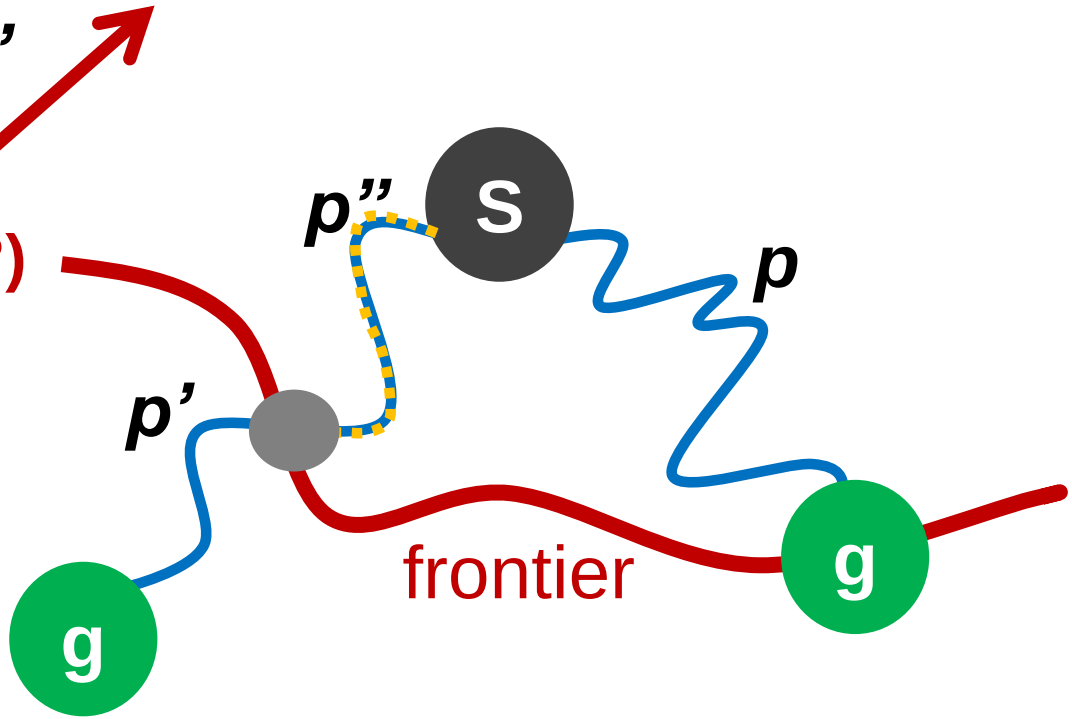


Proof of Optimality of A*

Consider the moment when ***p*** is chosen from the frontier.

Some part of path ***p'*** will also be on the frontier; let's call this partial path ***p''***

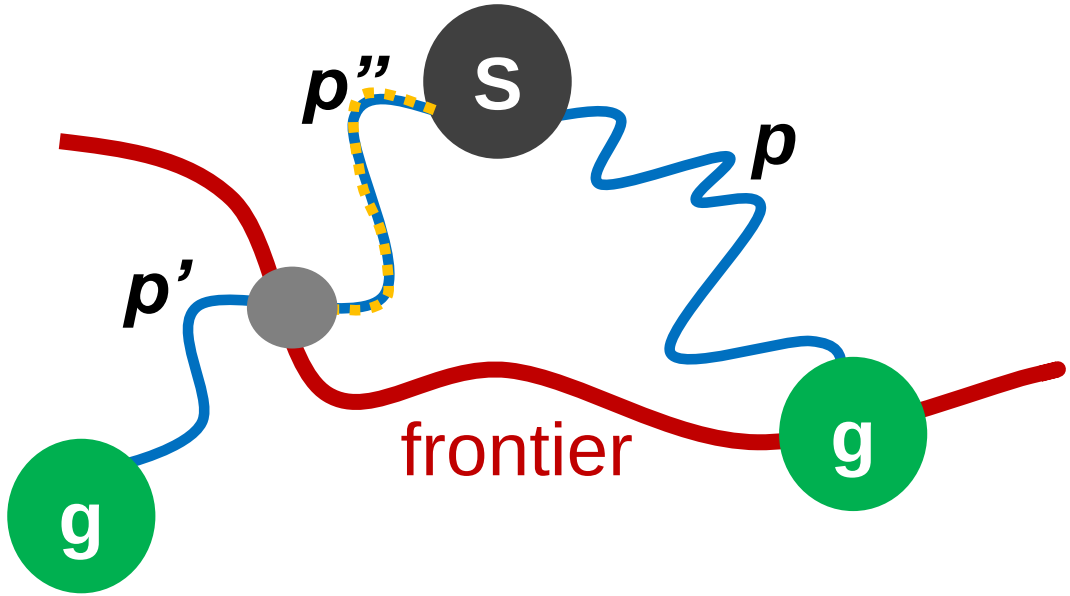
(Why can we claim this?)



Proof of Optimality of A*

Because ***p*** was expanded before ***p”***, **f(p) ≤ f(p”)**; therefore,

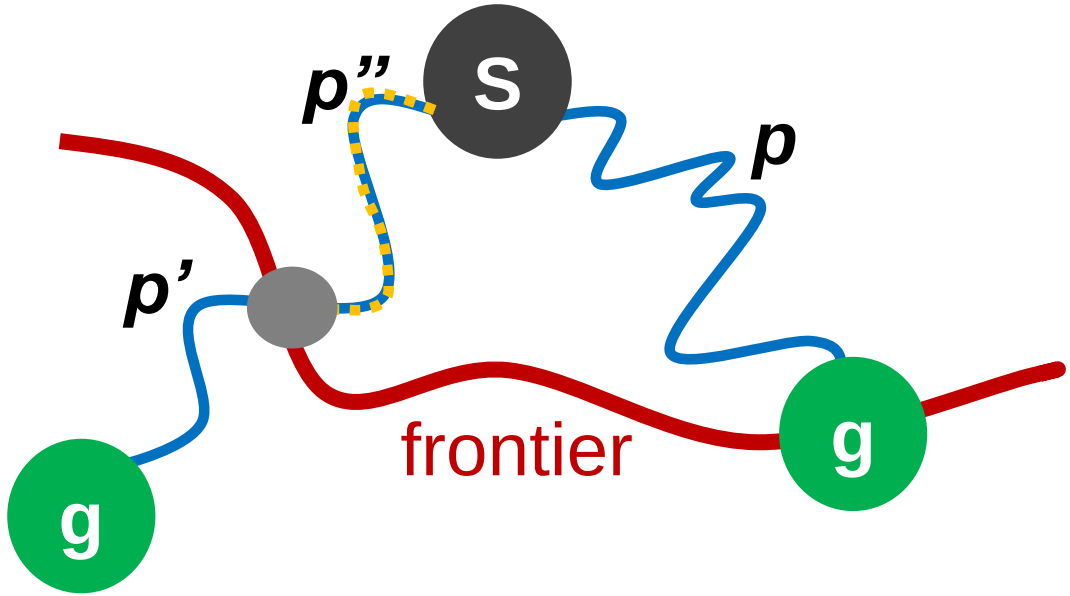
cost(p) + h(p) ≤ cost(p”) + h(p”)



Proof of Optimality of A*

Because ***p*** ends at a goal, **h(p)=**

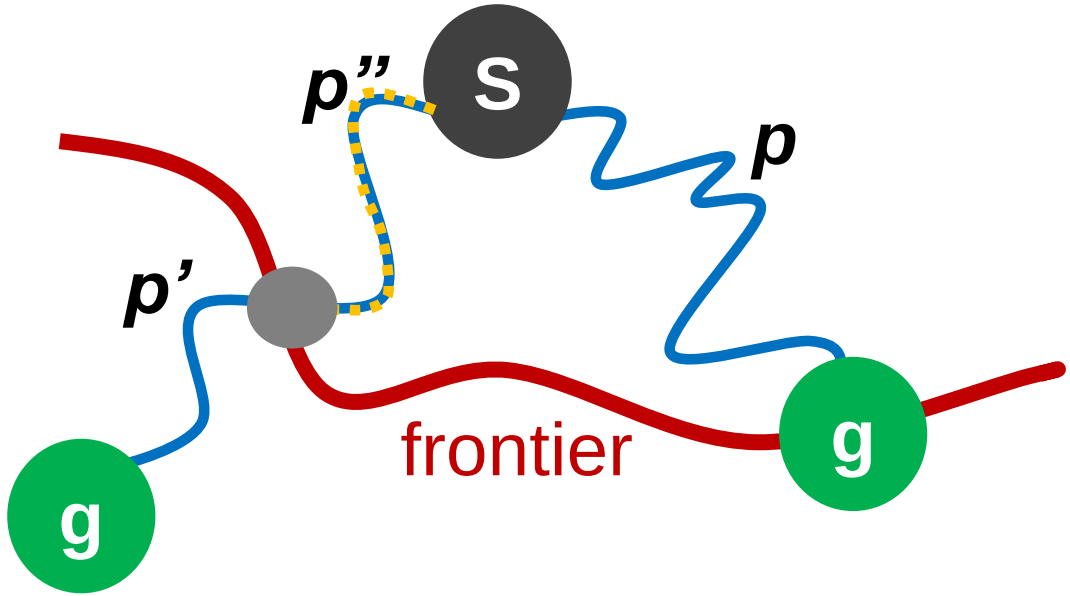
cost(p) + h(p) ≤ cost(p”) + h(p”)



Proof of Optimality of A*

Because ***p*** ends at a goal, **h(p)=0**

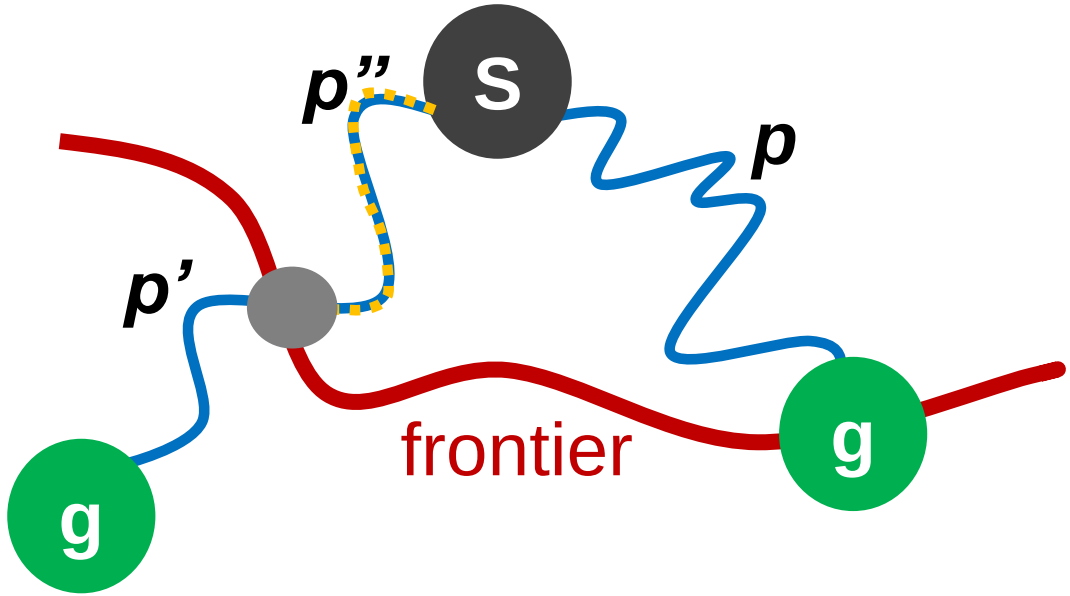
cost(p) + h(p) ≤ cost(p") + h(p")



Proof of Optimality of A*

Because ***p*** ends at a goal, **h(p)=0**

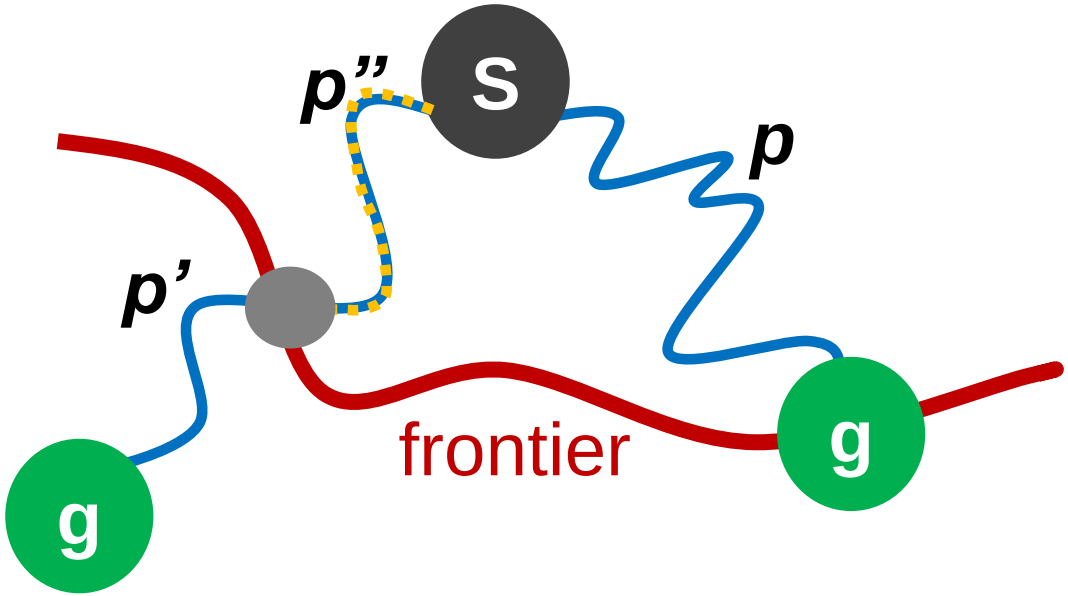
cost(p) ≤ cost(p”) + h(p”)



Proof of Optimality of A*

Because **h** is admissible, **cost(p”) + h(p”) ≤ cost(p’)**

cost(p) ≤ cost(p”) + h(p”)

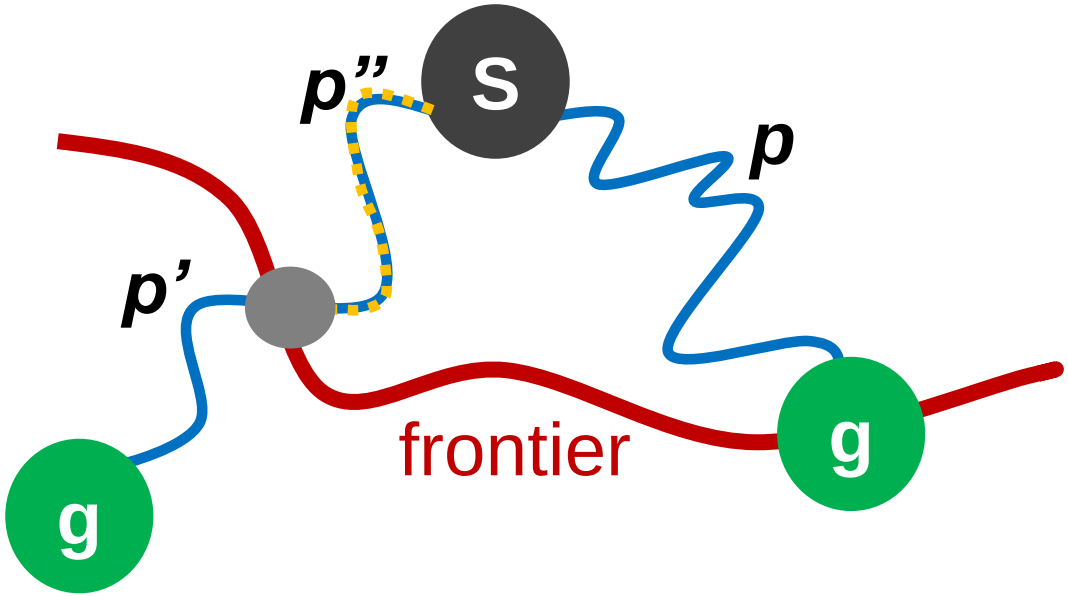


Proof of Optimality of A*

Therefore, we see that

$$\text{cost}(p) \leq \text{cost}(p')$$

Which **contradicts** our assumption that p' was a better path!



Optimal efficiency of A^*

- In fact, we can prove something even stronger about A^*: in a sense (given the particular heuristic that is available) **no search algorithm could do better!**

- **Optimal Efficiency:** Among **all optimal algorithms** that **start from the same start node** and **use the same heuristic** *h*, A^* expands the minimal number of paths.
  - *Note: we're ignoring possible issues with tie-breaking*

Sample A* applications

- **An Efficient A* Search Algorithm For Statistical** Machine Translation. 2001
- **The Generalized A* Architecture**. Journal of Artificial Intelligence Research (2007)
  - Machine Vision … Here we consider a new compositional model for finding salient curves.
- **Factored A*search for models over sequences and trees** International Conference on AI. 2003…. It starts saying… *The primary challenge when using A* search is to find heuristic functions that simultaneously are admissible, close to actual completion costs, and efficient to calculate…* applied to NLP and BioInformatics

Sample A* applications (cont')

Aker, A., Cohn, T., Gaizauskas, R.: **Multi-document summarization using A* search and discriminative training.** Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing.. ACL (2010)

Sample A* applications (cont')

EMNLP 2014 **A* CCG Parsing with a Supertag-factored Model** M. Lewis, M. Steedman

We introduce a new CCG parsing model which is factored on lexical category assignments. Parsing is then simply a deterministic search for the most probable category sequence that supports a CCG derivation. The parser is extremely simple, with a tiny feature set, no POS tagger, and no statistical model of the derivation or dependencies. Formulating the model in this way allows a highly effective heuristic for A∗ parsing, which makes parsing extremely fast. Compared to the standard C&C CCG parser, our model is more accurate out-of-domain, is four times faster, has higher coverage, and is greatly simplified. We also show that using our parser improves the performance of a state-of-the-art question answering system

Follow up ACL 2017 (main NLP conference – in Vancouver!)

A* CCG Parsing with a Supertag and Dependency Factored Model Masashi Yoshikawa, Hiroshi Noji, Yuji Matsumoto

Search Summary Table

	complete?	optimal?	time O( )	space O( )
DFS	No	No	b^m	mb
BFS	Yes	Yes*	b^m	b^m
IDS	Yes	Yes*	b^m	mb
LCFS	Yes	Yes^	b^m	b^m
BestFS	No	No	b^m	b^m
A*	Yes	Yes^{+}	b^m	b^m

* Assuming arc costs are equal
^ Assuming arc costs are positive
$^{+}$ Assuming h(n) is admissible and non-negative

Next class

Finish Search (finish Ch. 3)

- Branch-and-Bound
- A* enhancements
- Pruning