Heuristic Search: BestFS and A*

Computer Science cpsc322, Lecture 8

(Textbook Chpt 3.6)

May, 23, 2017

Lecture Overview

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

How to Combine Heuristics

iclicker

If $h_1(n)$ is admissible and $h_2(n)$ is also admissible then

A. **min($h_1(n)$, $h_2(n)$)** is also admissible and dominates its components (doesn't dominate)

B. **sum($h_1(n)$, $h_2(n)$)** is also admissible and dominates its components (may not be admissible)

C. **avg($h_1(n)$, $h_2(n)$)** is also admissible and dominates its components (doesn't dominate)

D. **None** of the above

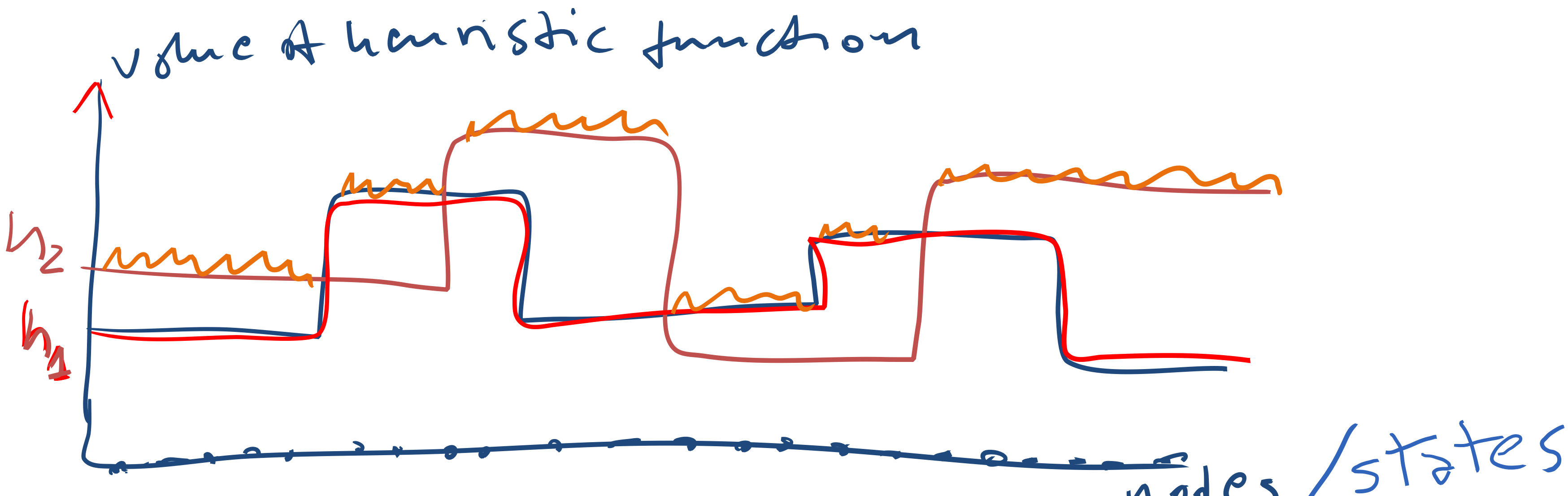
Combining Admissible Heuristics

How to combine heuristics when there is no dominance?

If $h_1(n)$ is admissible and $h_2(n)$ is also admissible then

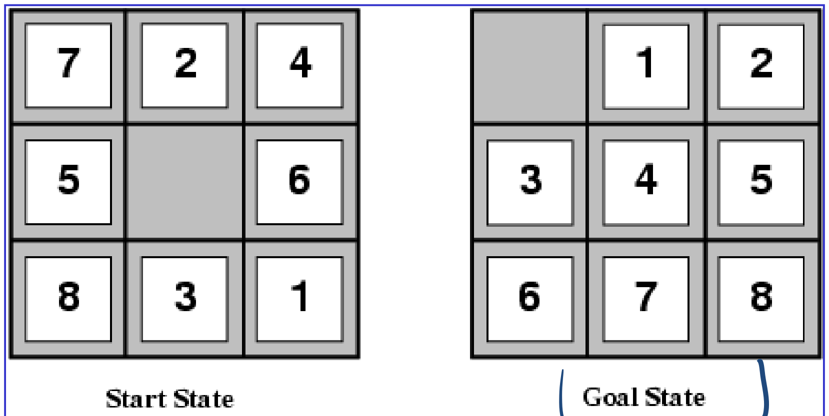
$h(n)$= ___$\max(h_1, h_2)$___ is also admissible

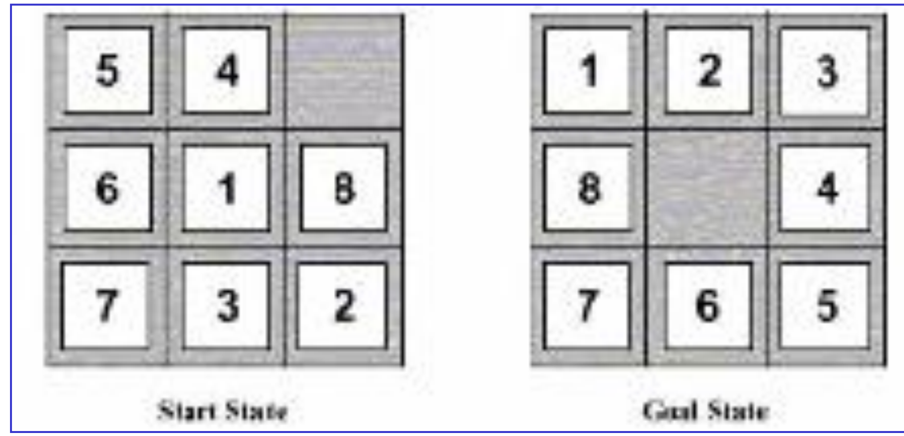
… and dominates all its components



Example Heuristic Functions (2)

- Another one we can use the number of moves between each tile's current position and its position in the solution





tiles

1	2	3	4	5	6	7	8
3	1	2	2	2	3	3	2

$\sum$ = 18

1	2	3	4	5	6	7	8
2	3						

Another approach to construct heuristics

Solution cost for a subproblem

1 2 3 4

Original Problem

	1	3
8	2	5
7	6	4

Current node

simpler

SubProblem

	1	3
@	2	@
@	@	4

1	2	3
8		4
7	6	5

Goal node

1	2	3
@		4
@	@	@

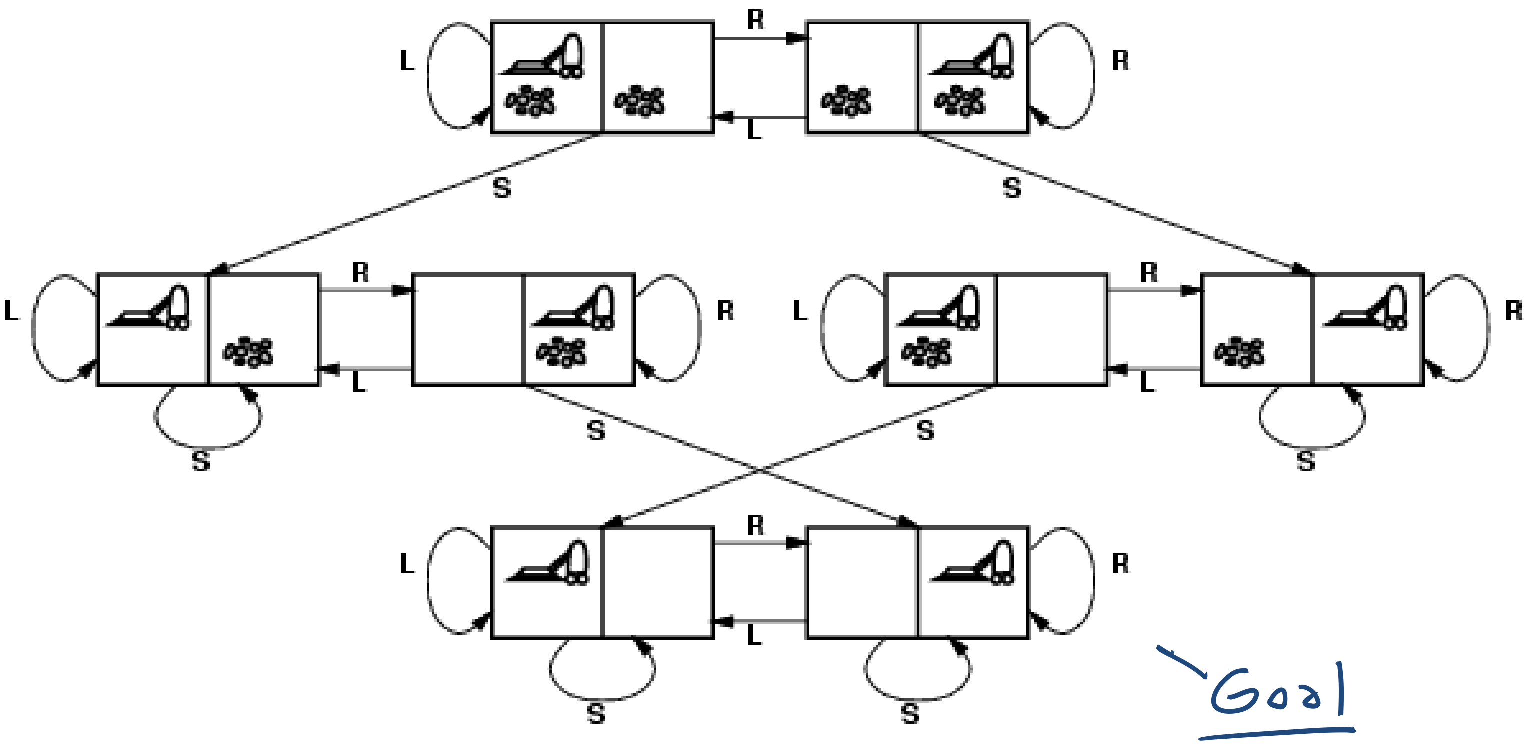
Goal

Combining Heuristics: Example

In 8-puzzle, solution cost for the 1,2,3,4 subproblem is substantially more accurate than sum of Manhattan distance of each tile from its goal position **in some cases**

So…..

Admissible heuristic for Vacuum world?

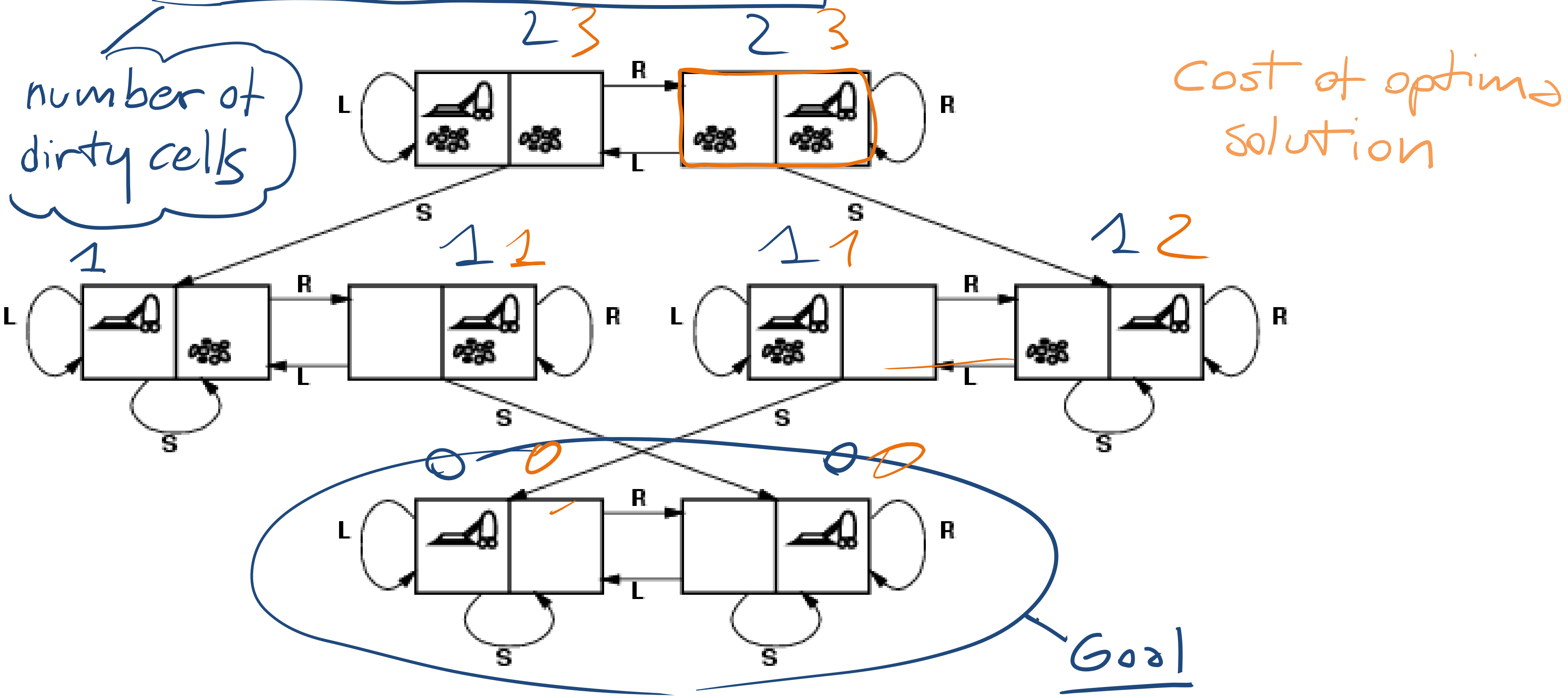


states? Where it is dirty and robot location

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

Possible goal test? no dirt at all locations

Admissible heuristic for Vacuum world?



states? Where it is dirty and robot location

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

Possible goal test? no dirt at all locations

Lecture Overview

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

Best-First Search

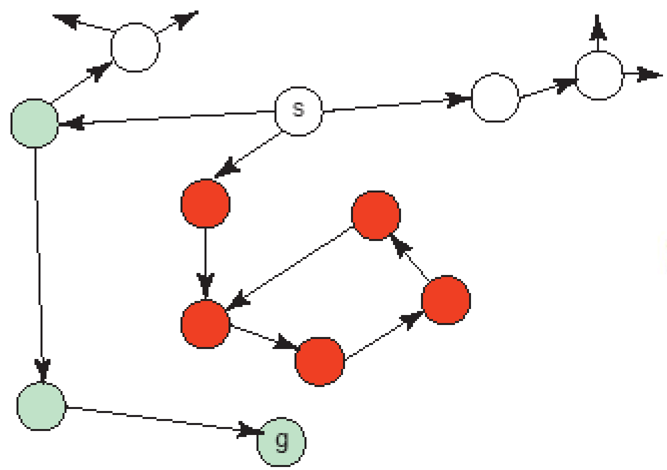
- **Idea:** select the path whose end is closest to a goal according to the heuristic function.

- **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 ?) LCFS ← by cost
- This is a greedy approach: it always takes the path which appears locally best

Analysis of Best-First Search

- Not Complete : a low heuristic value can mean that a cycle gets followed forever.



see course web page for file

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

worst case

Lecture Overview

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

How can we effectively use h(n)

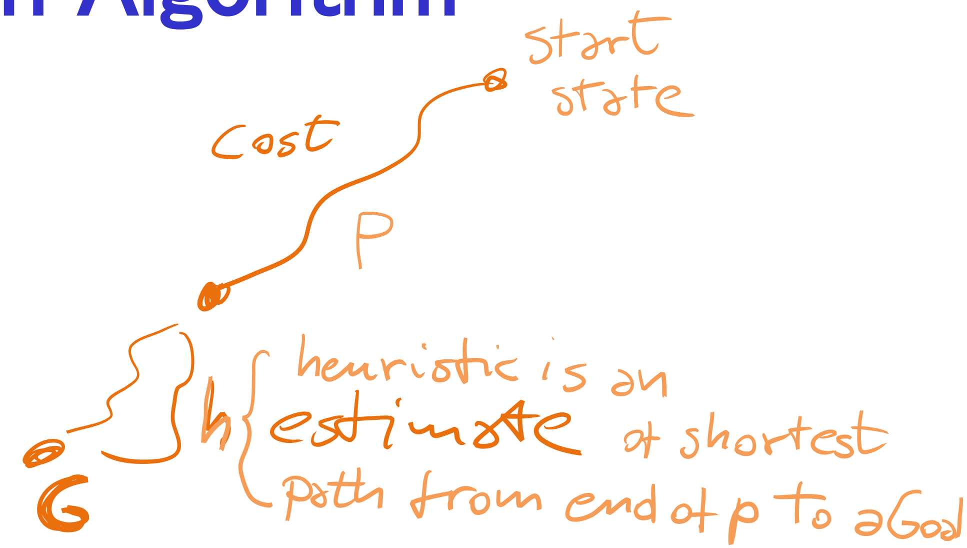
Maybe we should combine it with the cost. How?

Shall we select from the frontier the path *p* with:

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** Search Algorithm

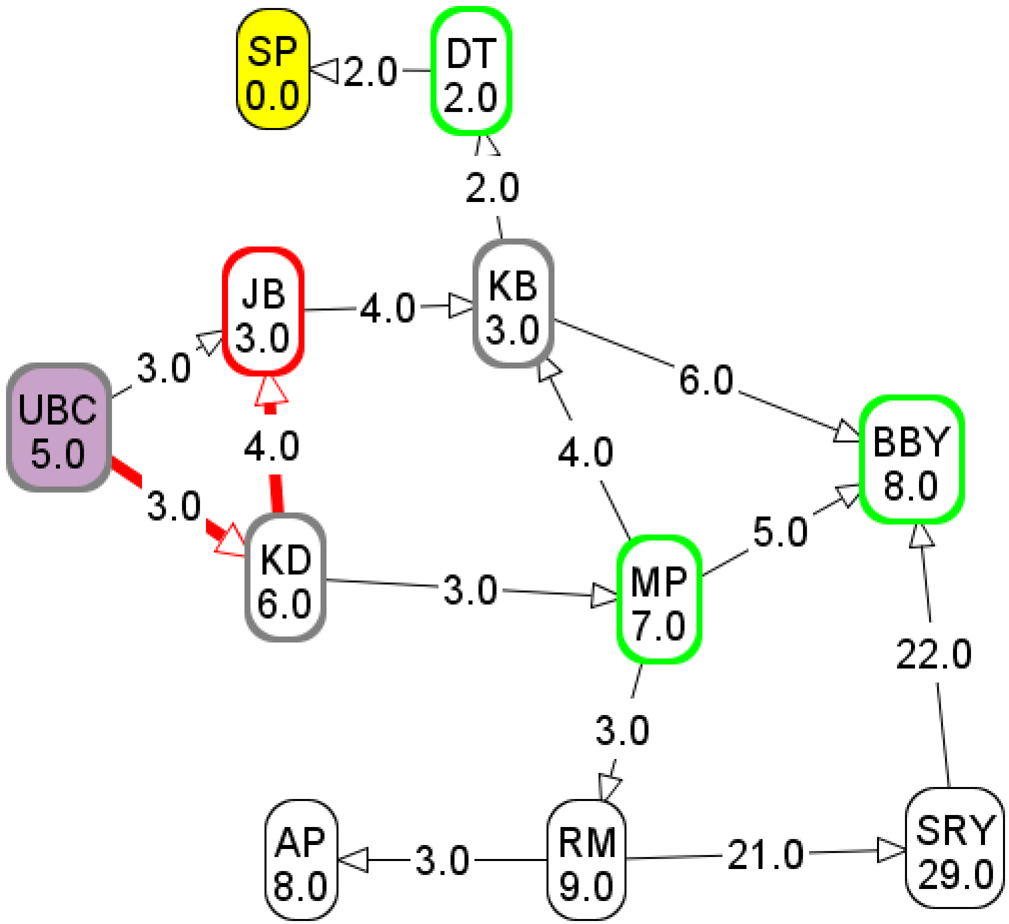
- *A** is a mix of:
  - **lowest–cost–first** and
  - **best–first search**



- *A** treats the frontier as a priority queue ordered by $f(p)=$ cost(p) + h(p) — is an estimate

- It always selects the node on the frontier with the lowest estimated total distance.

Computing f-values



f-value of UBC → KD→ JB?

6 | 9 | 10 | 11

Analysis of A*

iclicker

If the heuristic is completely uninformative and the edge costs are all the same, A* is equivalent to….

A. BFS
B. LCFS
C. DFS
D. None of the Above

Analysis of A^*

for all states heuristic is equal to 0

Let's assume that arc costs are strictly positive.

- Time complexity is $O(b^m)$ $\forall s\ h(s) = 0$
  - the heuristic could be completely uninformative and the edge costs could all be the same, meaning that A^* does the same thing as….

DFS BFS LCFS

- Space complexity is $O(b^m)$ like BFS…., A^* maintains a frontier which grows with the size of the tree

- Completeness: yes.

- Optimality: ??

Optimality of *A**

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

When

- 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, and is non-negative

admissible

Theorem

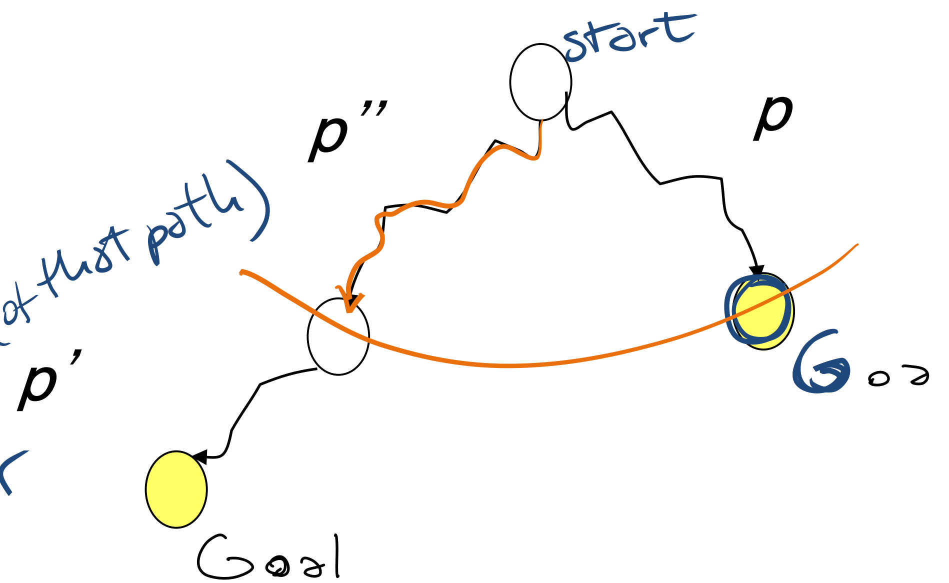
If *A** selects a path *p* as the solution,
p is the shortest (i.e., lowest-cost) path.

Why is *A** optimal?

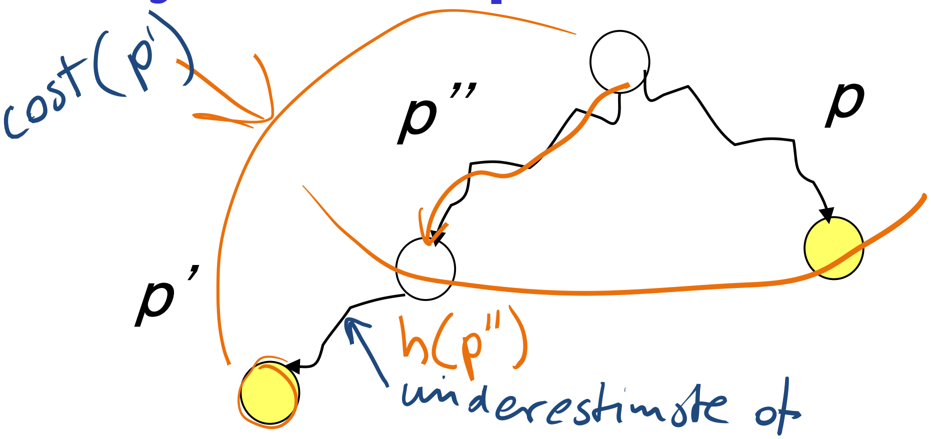
- A* returns *p*
- Assume for contradiction that some other path *p'* is actually the shortest path to a goal
- Consider the moment when *p* is chosen from the frontier Some part of path *p'* will also be on the frontier; let's call this partial path *p''*.

cost(p) > cost(p')

Think! for any path from start to any state there is always a sub path (of that path) on the frontier



Why is *A** optimal? (cont')



$cost(p) + h(p) \le cost(p'') + h(p'')$

- Because *p* was expanded before *p''*, $f(p) \le f(p'')$
- Because *p* is a goal, $h(p) = 0$ Thus $cost(p) \le cost(p'') + h(p'')$ ①
- Because *h* is admissible, $cost(p'') + h(p'') \le cost(p')$ ② for any path *p'* to a goal that extends *p''*

combining ① and ②

- Thus $cost(p) \le cost(p')$ for any other path *p'* to a goal.

$cost(p') < cost(p)$

This contradicts our assumption that *p'* is the shortest 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.

Samples 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

(Natural Language Processing)

Samples 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)

Samples 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 – will be in Vancouver in August!)

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

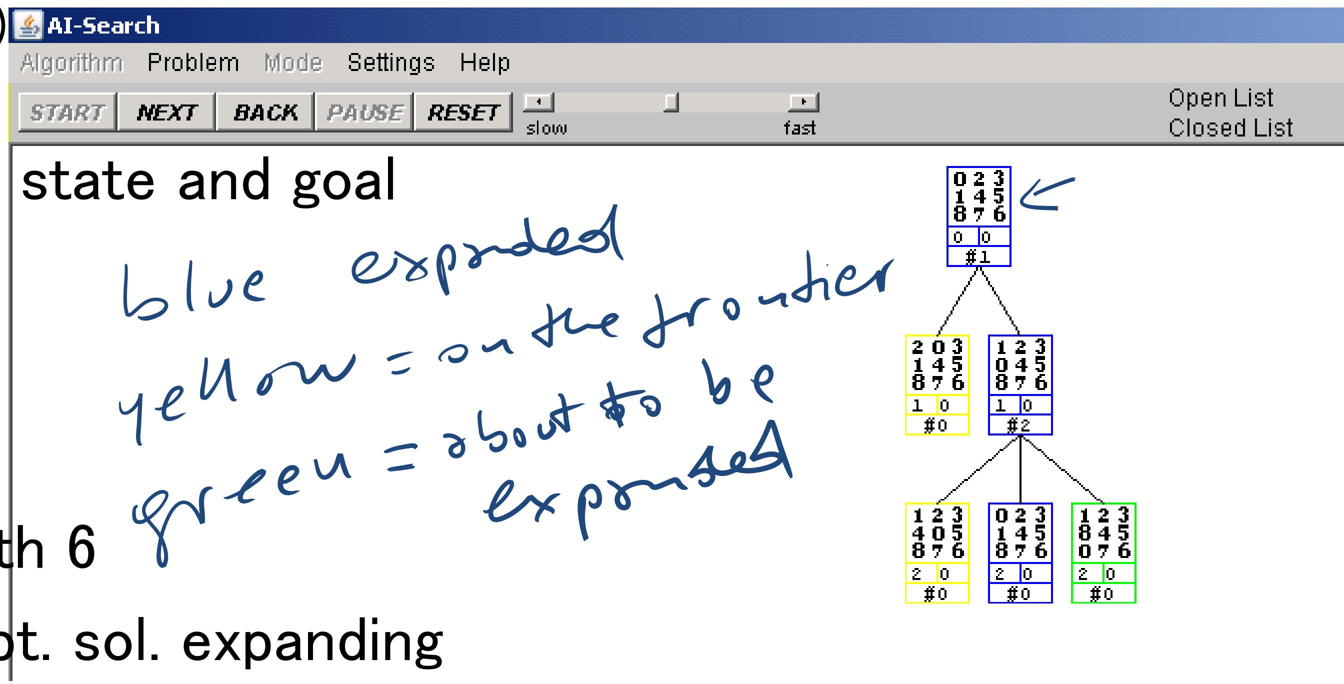
*DFS, BFS, A** Animation Example

- TheAI-Search animation system

http://www.cs.rmit.edu.au/AI-Search/Product/

DEPRECATED ☹

- *To examine Search strategies when they are applied to the 8puzzle*
- Compare only DFS, BFS andA* (with only the two heuristics we saw in class)



- With default start state and goal
- DFS will find

Solution at depth 32

- BFS will find

Optimal solution depth 6

- A* will also find opt. sol. expanding much less nodes

nPuzzles are not always solvable

Half of the starting positions for the *n*-puzzle are impossible to solve (for more info on 8puzzle)

- So experiment with the AI-Search animation system (DEPRECATED) with the default configurations.
- If you want to try new ones keep in mind that you may pick unsolvable problems

Learning Goals for today's class

- **Define/read/write/trace/debug** & **Compare** different search algorithms
  - With / Without cost
  - Informed / Uninformed
- Formally prove A* optimality.

Next class

Finish Search (finish Chpt 3)

- Branch-and-Bound
- A* enhancements
- Non-heuristic Pruning
- Dynamic Programming