15-780: Graduate AI

Lecture 1. Intro & Search

Geoff Gordon (this lecture)
Ziv Bar-Joseph
TAs Michael Benisch, Yang Gu

Admin

www.cs.cmu.edu/~ggordon/780/

Image from USA Today

15-780: Graduate Artificial Intelligence, Fall 2006

Home
Schedule
Handouts
News
Links

Course Overview

Lectures | Mon. & Wed. 10:30 AM - 11:50 AM in **Wean Hall 5409** (starting on 9/11)

This course is targeted at graduate students who need to learn about current-day research, and about how to perform current-day research, in Artificial Intelligence---the discipline of designing intelligent decision-making machines.

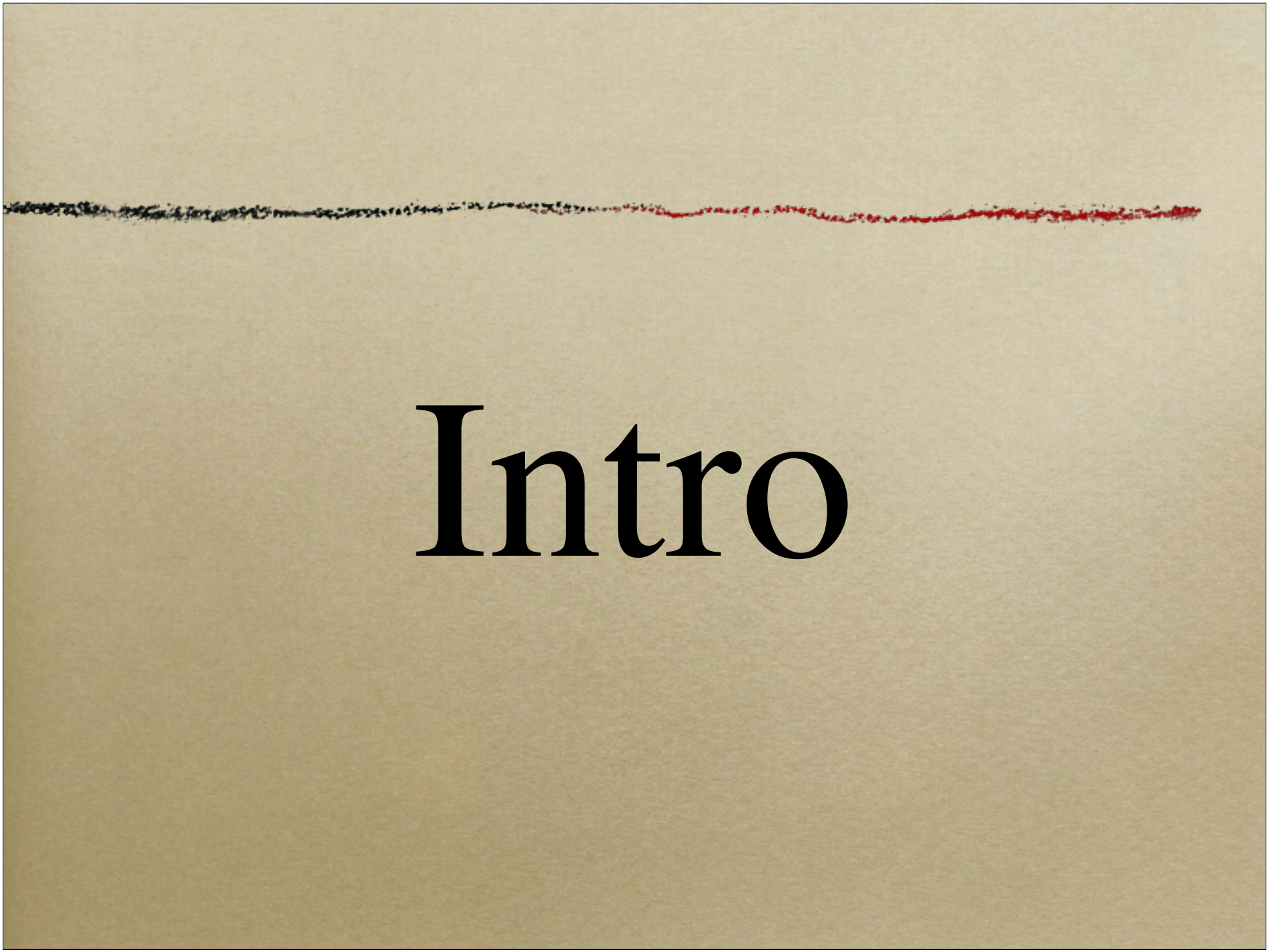
Techniques from Probability, Statistics, Economics, Algorithms, Operations Research and Optimal Control are increasingly important tools for improving the intelligence and autonomy of machines, whether those machines are robots surveying Antarctica, schedulers moving billions of dollars of inventory, spacecraft deciding which experiments to perform, or vehicles negotiating for lanes on the freeway. This AI course is a review of a selected set of these tools. The course will cover the ideas underlying these tools, their implementation, and how to use them or extend them in your research.

A PDF version of the information on this page can be found here.

Instructors

Website highlights

- Book: Russell and Norvig. *Artificial Intelligence: A Modern Approach*, 2nd ed.
- Grading
- Final project
- Office hours


Intro

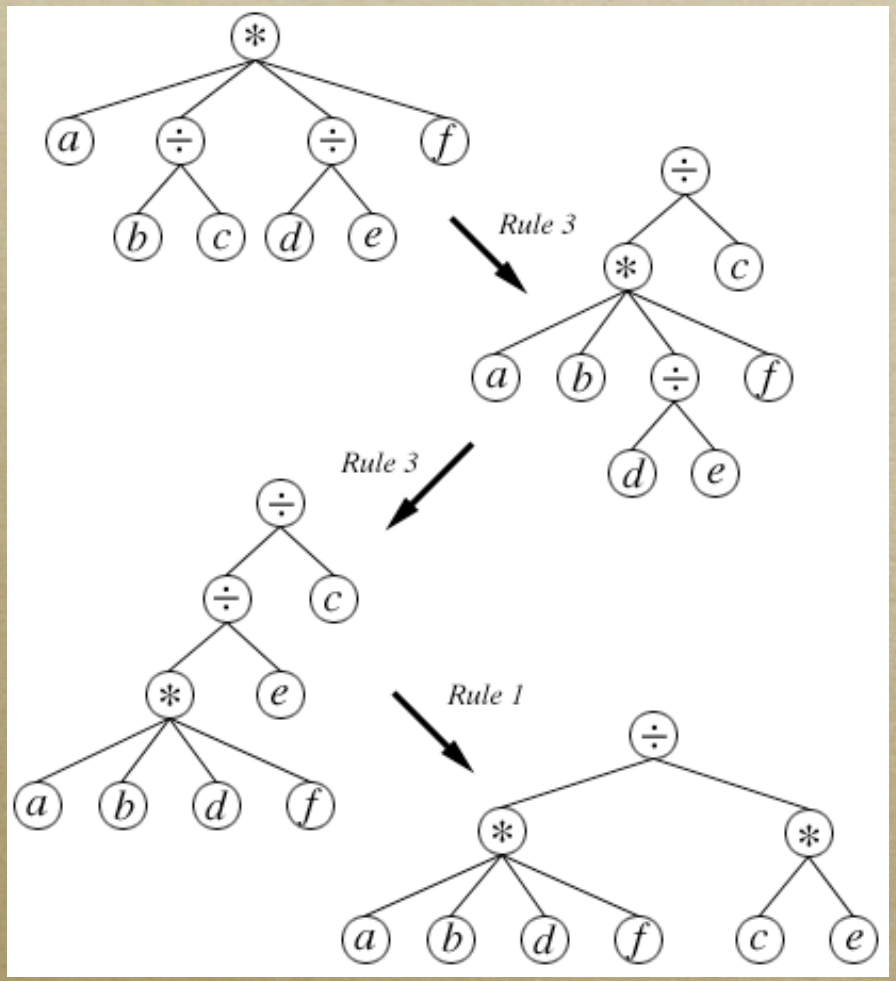
What is AI?

- *Easy part: A*
- *Hard part: I*
  - *Anything we don't know how to make a computer do yet*
  - *Corollary: once we do it, it isn't AI anymore :-)*

Definition by examples

- *Deep Blue*
- *TD-Gammon*
- *Samuels's checkers player*
- *Mathematica*

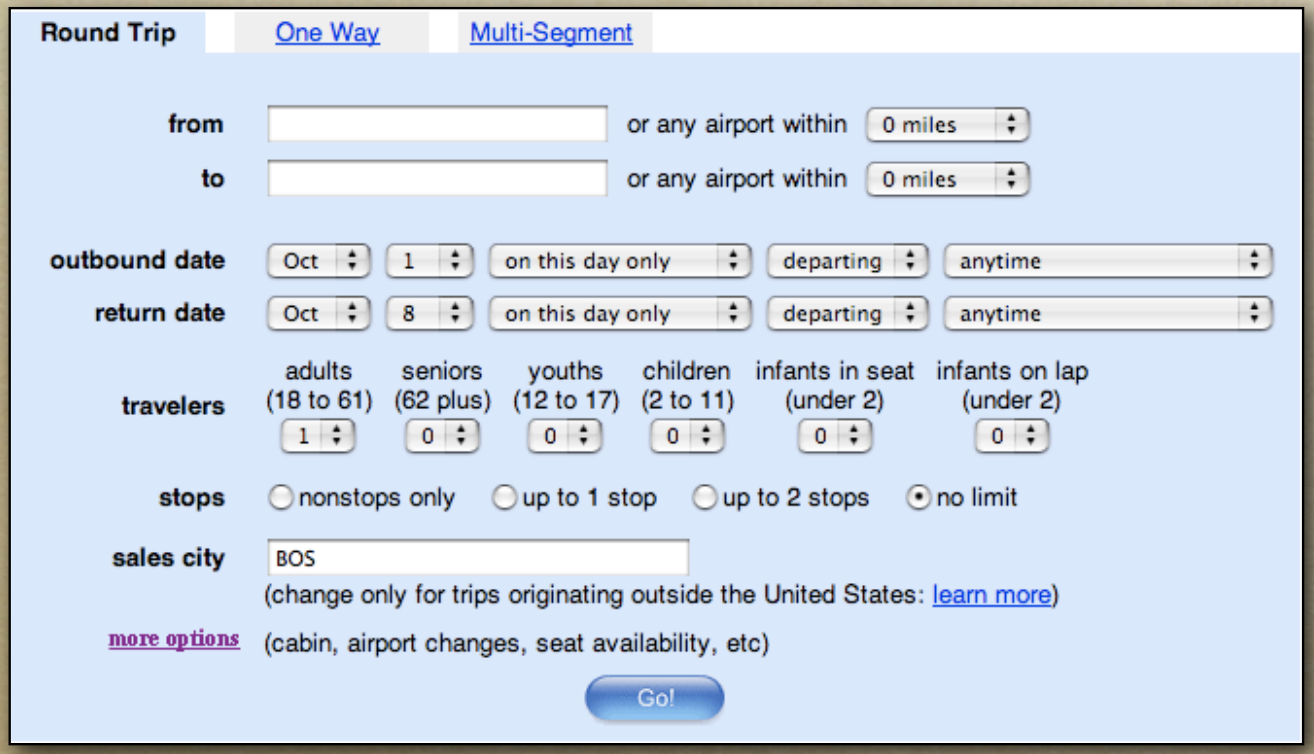
from http://www.math.wpi.edu/IQP/BVCalcHist/calc5.html



More examples

- *Grand Challenge road race*

More examples



- *ITA software (http://beta.itasoftware.com)*

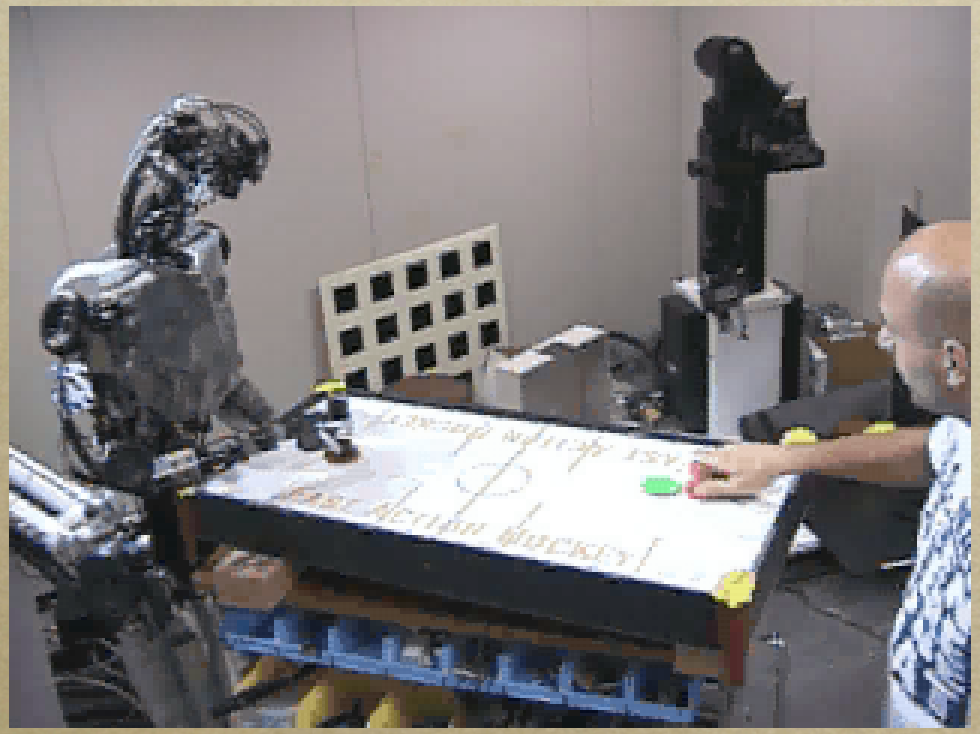
More examples

- *Robocup*

More examples

- *Marble-maze game*

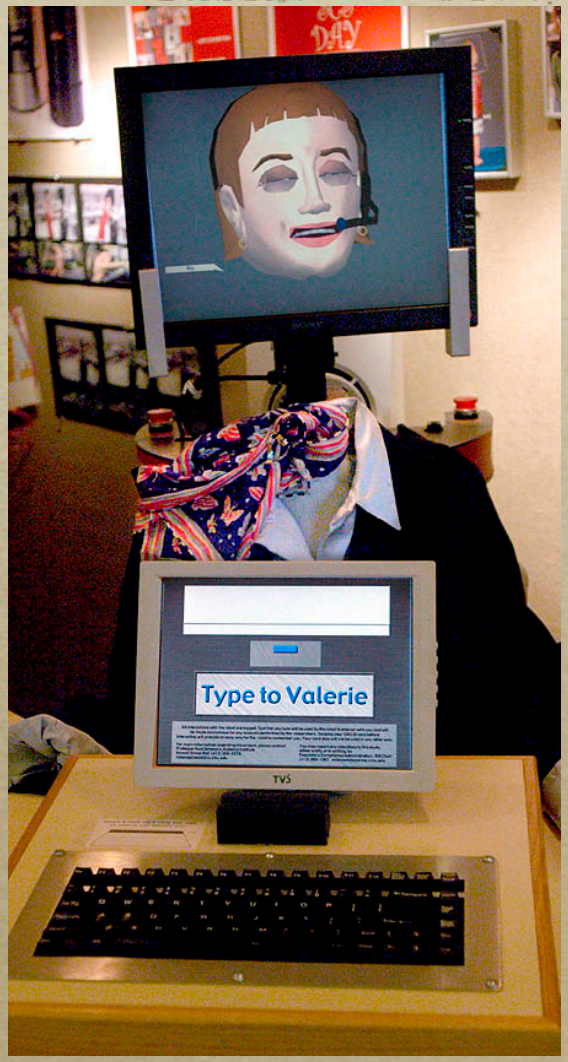
More examples



- *Air hockey*

More examples

- *Valerie and Tank, the Roboceptionists*



More examples

http://www.cs.cmu.edu/~ggordon/poker/

- *Poker playing*

More examples

- *Airline crew scheduling*
- *Diagnosing F-18 engine faults*

More examples

- *Riding a bicycle, learning to walk, playing pool, …*
- *AAAI competitions: rescue, standing in line, …*

More examples

- *The IMP*
- *Robot exploring a building*
- *Robot team exploration w/ markets*

More examples

- *McCallum's driving simulator*
- *Kanfer-Ackerman air-traffic control task*

More examples

- *Keeping track of other agents moving*
- *Searching for wandering grad students*
- *Getting rid of phone trees using POMDPs*

Common threads

- *Search and optimization*
  - *Set the problem up well (so that we can apply a standard algorithm)*
- *Managing uncertainty*
  - *The more different types of uncertainty, the harder the problem (and the slower the solution)*

Sources of uncertainty

- *Classic AI: no uncertainty, pure search*
  - *Mathematica*
  - *deterministic planning*
- *This is the topic of Part I of the course*

Opponents cause uncertainty

- *In chess, must guess what opponent will do; cannot directly control him/her*
- *Alternating moves: game trees (Part I)*
- *Simultaneous or hidden moves: game theory (Part III; computationally harder, especially if a sequence of moves)*

Outcome uncertainty

- *In backgammon, we don't know ahead of time what the dice will show*
- *When driving a robot down a corridor, wheel slippage will cause unexpected deviations from commanded course*
- *MDPs (Part II) or alternating-move stochastic games (Part I)*

Sensor uncertainty

- *Image of a handwritten digit → 0, 1, …, 9*
- *Measurements of length, width of petals of an iris → one of three species*
- *Interpreting a camera image of a corridor*
- *For a given set of measurements, multiple answers may be possible*
- *More on learning problems in Part II*

Combining sensor and outcome uncertainty

- *Build a robotic mouse*
- *Lives in a cage with levers, lights, etc.*
- *Pressing levers in the right sequence dispenses a snack of robotic cheese*
- *Move around, experiment w/ levers to turn on lights, get robo-cheese*
- *This is a POMDP (more in Part II)*

Other agents cause uncertainty

- *In many AI problems, there are other agents who aren't (necessarily) opponents*
  - *Ignore them & pretend part of Nature*
  - *Assume they're opponents (pessimistic)*
  - *Learn to cope with what they do*
  - *Try to cooperate (paradoxically, this is the hardest case)*
- *Part III*


Search

How to build a robotic grad student

- *Grad students need to:*
  - *take classes*
  - *do research*
  - *get free food from the CS lounge*

Available classes

- *Grad AI: progress for graduation 4, time 4*
- *Wine tasting: progress 1, time 2*
- *Nonlinear Frobnitz Dynamics: progress 5, time 11*

Research

- *Student may work on research:*
  - *lots (L, time 9)*
  - *a moderate amount (M, time 4)*
  - *some (S, time 3)*

Food

- *Pizza: 500 calories/slice*
- *Donuts: 300 cal/each*
- *Mountain Dew: 200 cal/each, $1/each*

Notation

- *ClassGAI, ClassWT, ClassNFD (boolean)*
- *R ∈ {S,M,L}*
- *NumPizza, NumDonut, NumDew (int ≥ 0)*

Constraints

- *Must take courses w/ ttl progress ≥ 5*
- *Must eat 1000 ≤ calories ≤ 1500*
- *Must spend ≤ $2*
- *Total time ≤ 10*

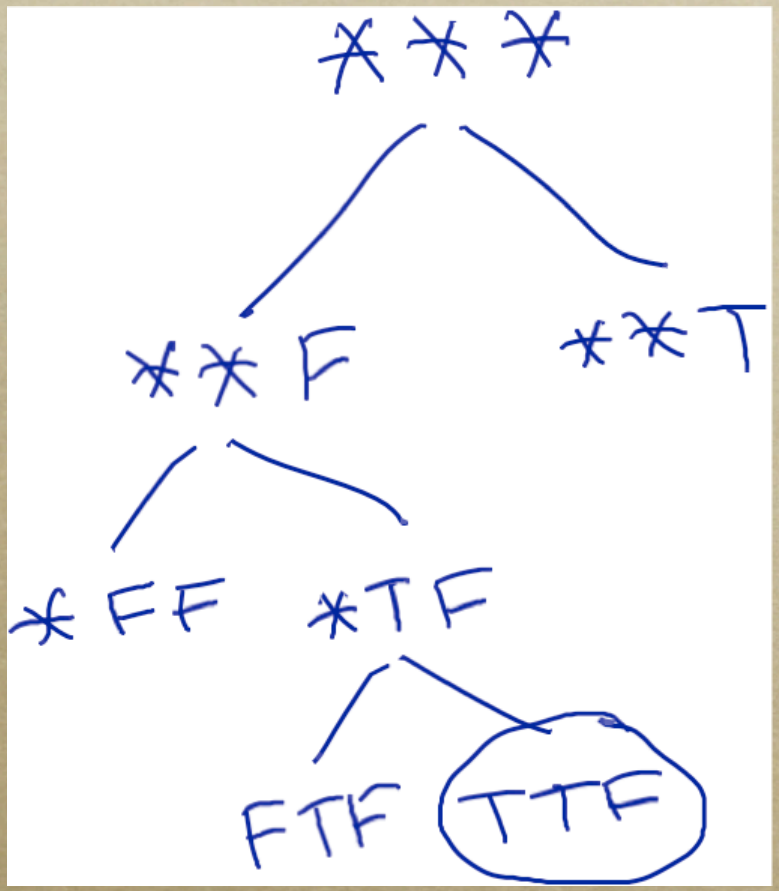
Solution by enumeration

- *Find feasible solutions for subproblem of setting ClassGAI, ClassWT, ClassNFD*

Can we do better?

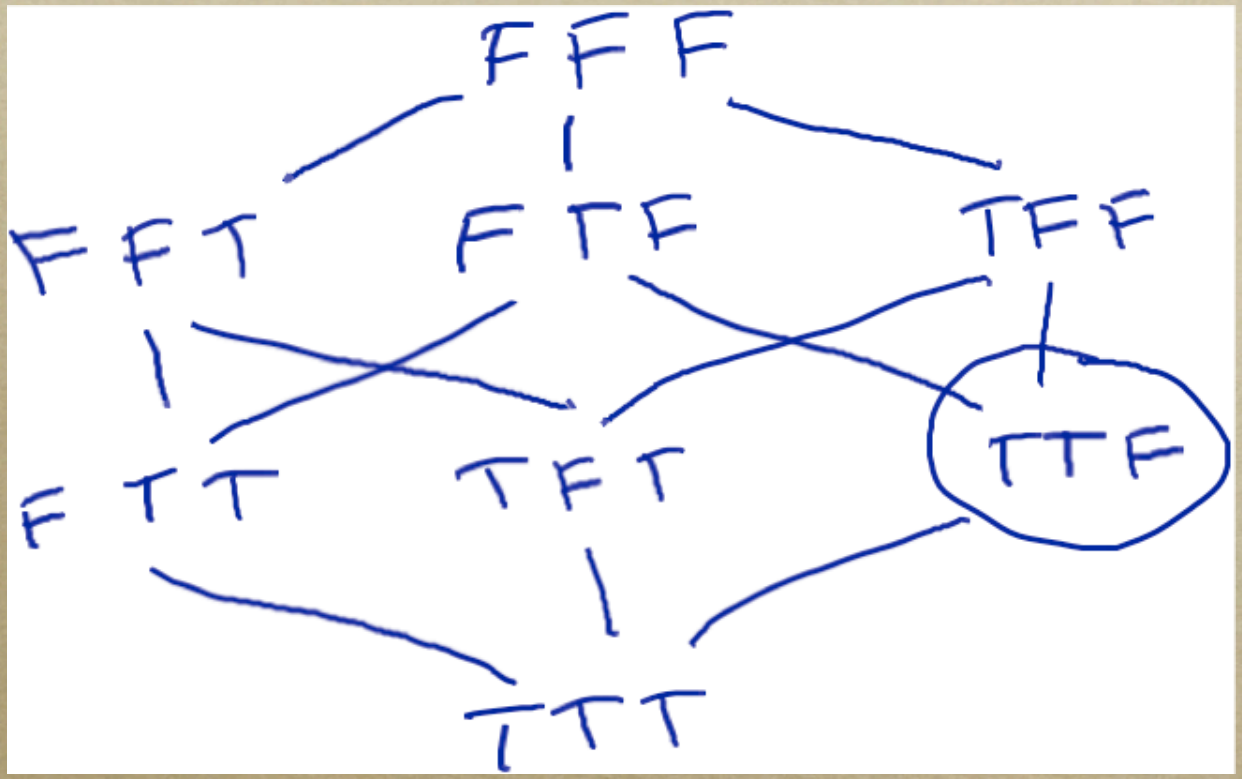
- *What about partial state ClassNFD=T (other vars unspecified)*

Search graph



- *Node: ***, **F, **T, *F*, *FF, *FT, …*

Alternate search graph



- *Nodes: FFF, FFT, FTF, FTT, …*

Search graph

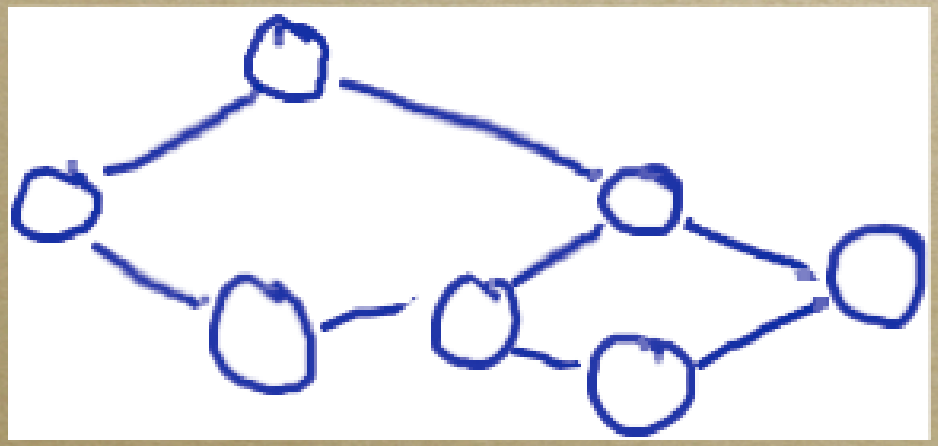
- *Node: solution or partial solution*
- *Neighbor generating function*
- *Solution test = yes, no, maybe*

Nodes can be anything

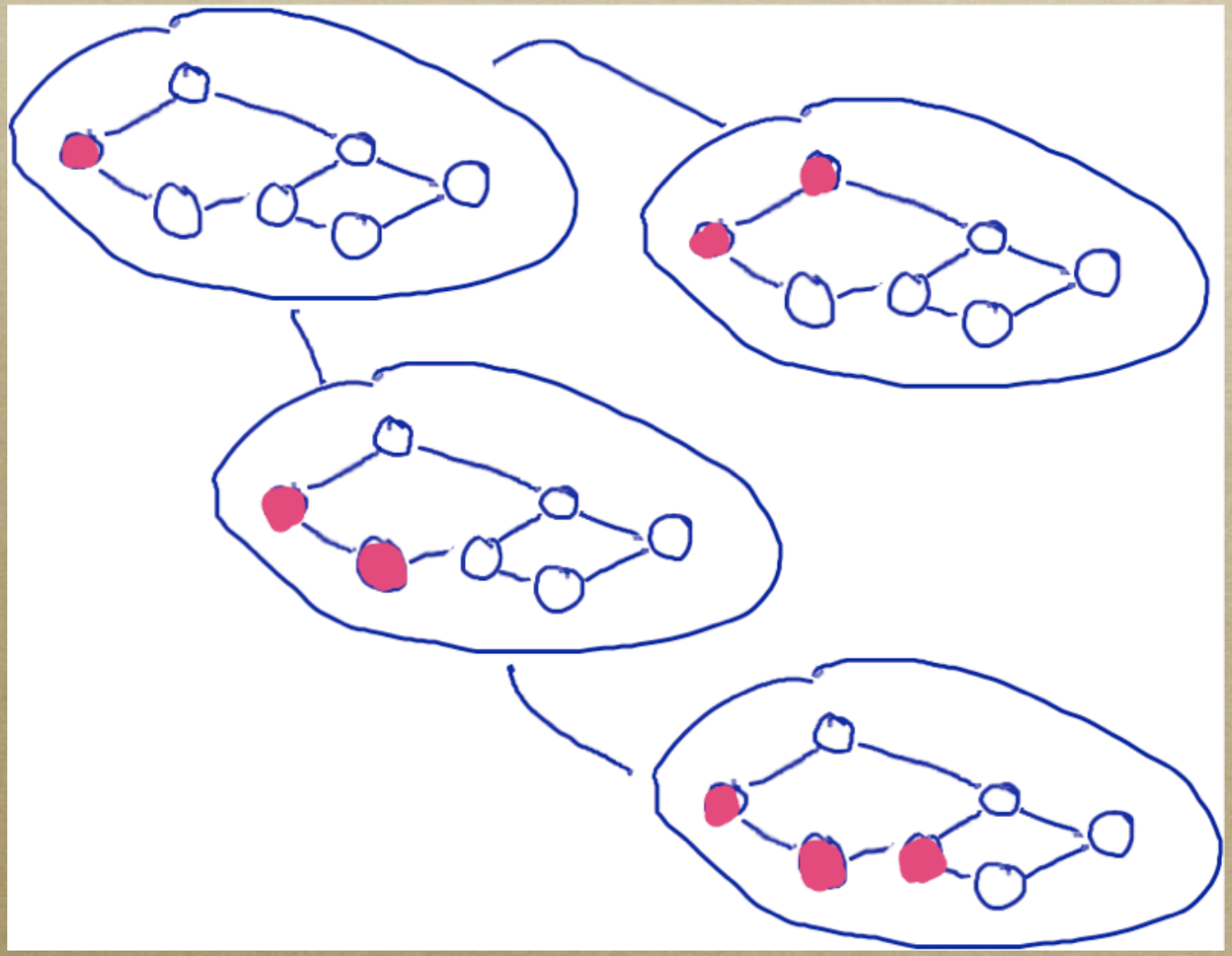
- *List of variable settings*
- *Mathematical formula*
- *A set of flights that go from PIT to LAX*
- *A graph*

When a node is a graph

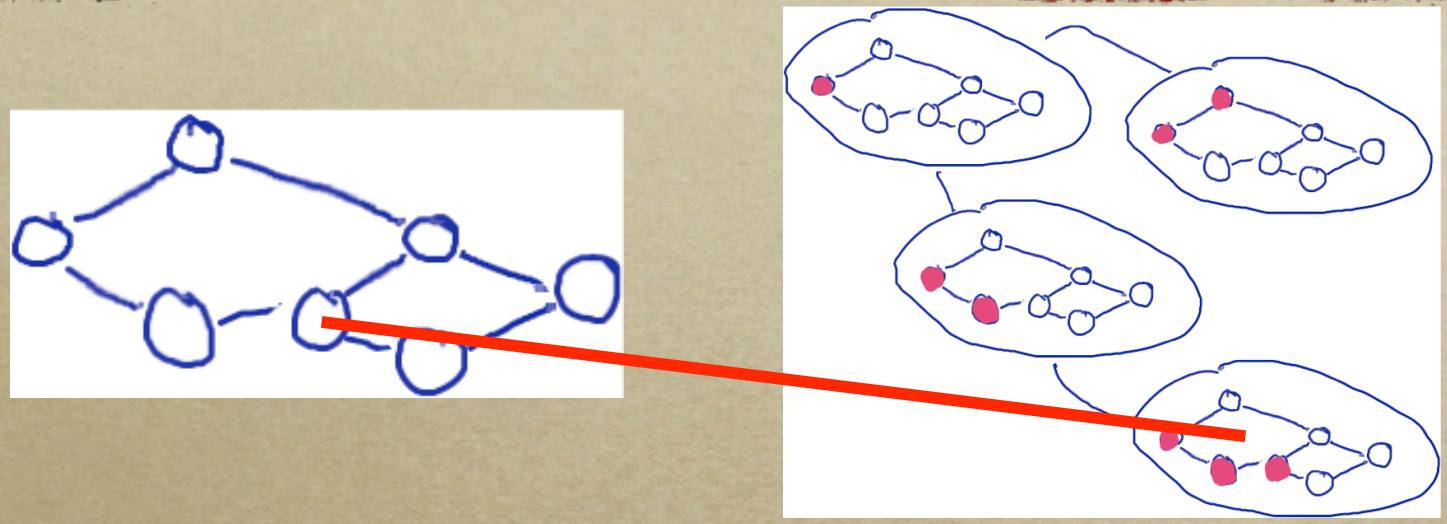
- *Not to be confused with search graph*
- *E.g., a (partial) matching, a (partial) spanning tree, or a (partial) path*



Search graph for shortest path



Isomorphism



- *For path planning, if we prune non-shortest paths, the search graph is isomorphic to the original graph*
- *Node X in original graph = shortest path from start to X*

Generic search

$S = \{$ *some set of nodes* $\}$ $\quad M = \emptyset$

While $(S \neq \emptyset)$

$x \leftarrow$ *some element of* S, $\quad S \leftarrow S \setminus x$

optional: $M = M \cup \{x\}$

if $(solution(x) = Y)$ *return* x

if $(solution(x) = N)$ *continue*

$S = S \cup (neighbors(x) \setminus M)$

Choices

- *Where to start?*
- *Which element of S to visit next?*
- *How much effort do we spend to avoid previously visited nodes?*
  - *Just don't return to parent*
  - *Keep nodes in path from start to X*
  - *Keep all nodes*

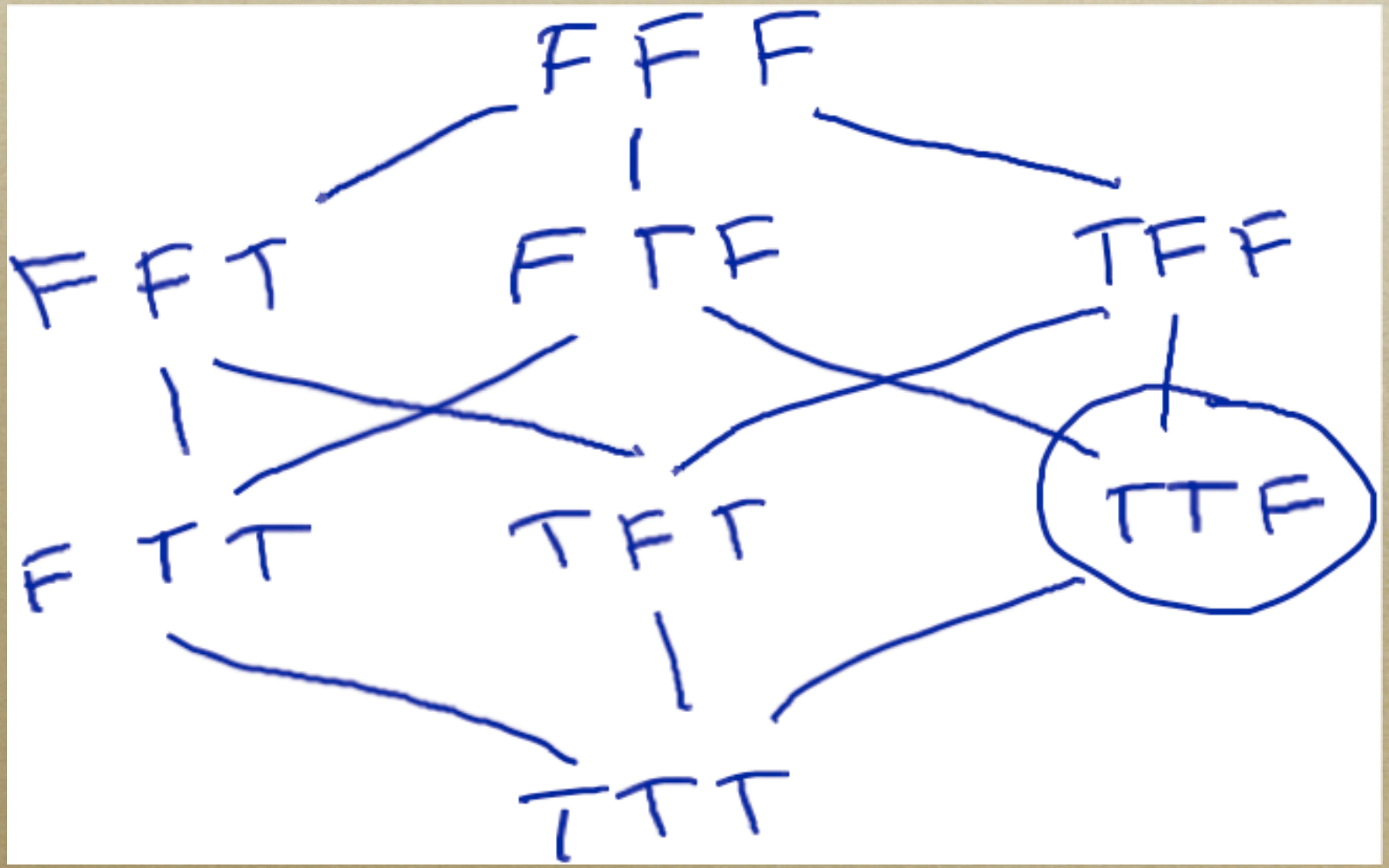
Data structures: M

- *Need insert, membership test*
  - *hash table (expense of equality test?)*
  - *avoid M altogether using node ordering*
    - *only insert neighbors $y > x$ into S*
    - *or just modify neighbors(x)*

Data structures

- *For S: need insert, pop*
  - *LIFO (stack)*
  - *FIFO (queue)*
  - *priority queue (choice of priority)*

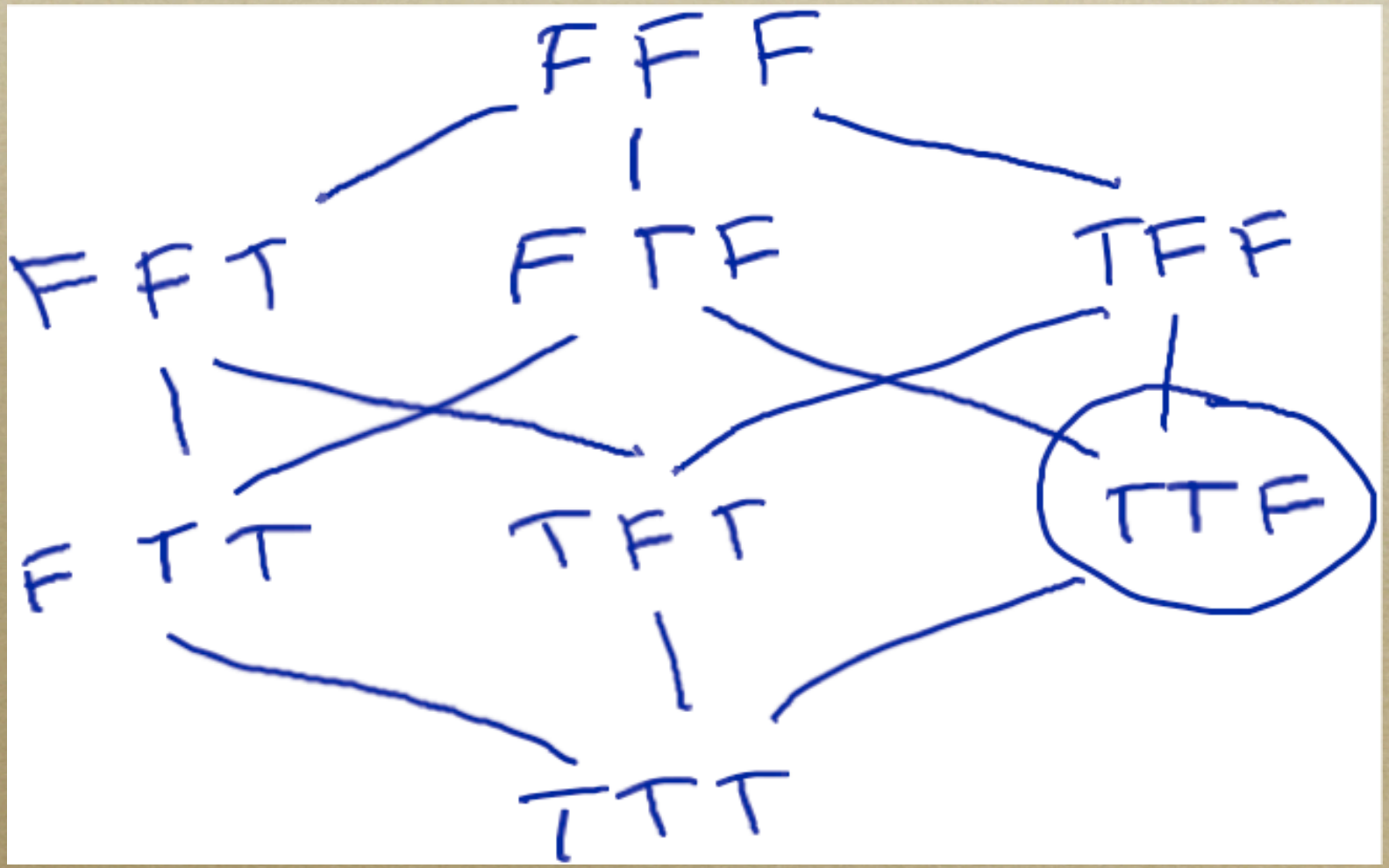
Stack: DFS



DFS discussion

- *Advantages*
  - *low memory if search graph is shallow*
- *Disadvantages*
  - *fails to terminate if graph has infinite depth*
  - *May not find shallowest solution*

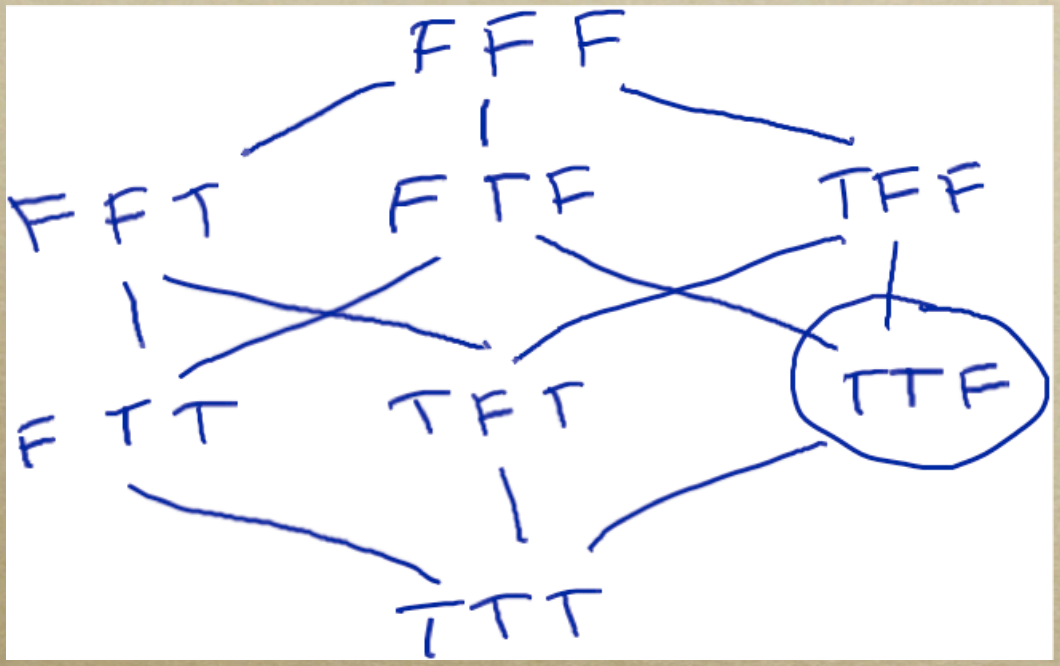
Queue: BFS



BFS discussion

- *Advantages*
  - *low memory if graph is narrow (rare)*
  - *always finds shallowest solution*
- *Disadvantages*
  - *common case: memory grows exponentially with search depth*

DFID



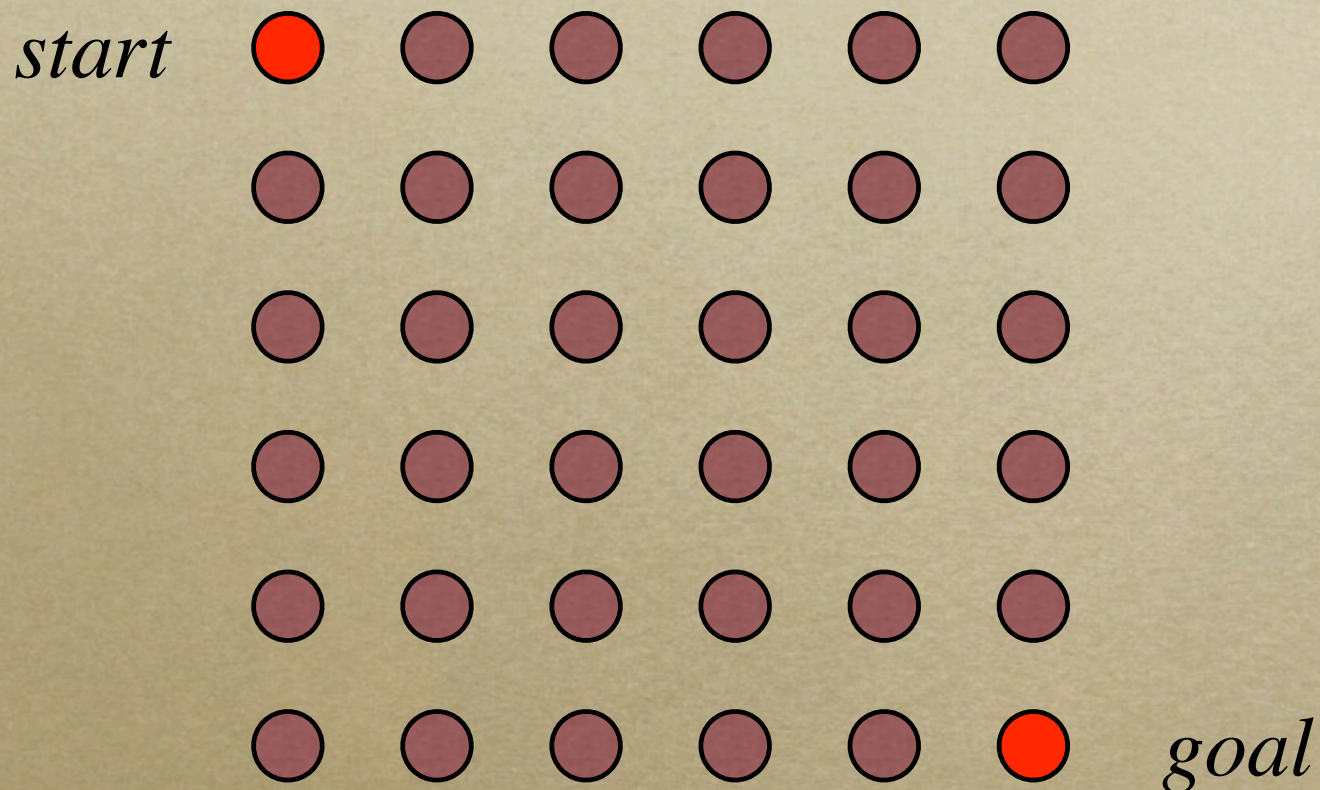
- *Run a DFS but limit search depth to k*
- *If we fail to find a solution, increase k and try again*

DFID discussion

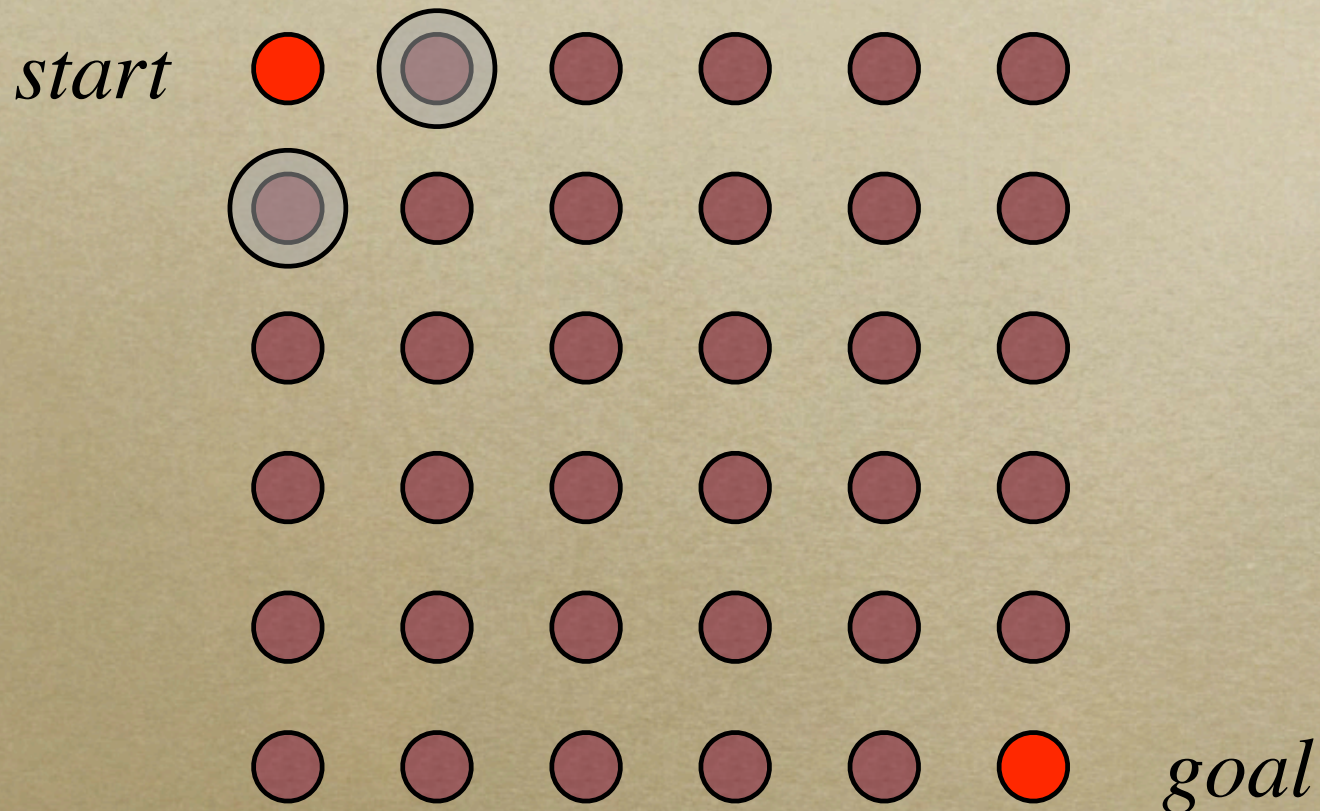
- *Combines advantages of BFS and DFS*
- *Always low memory*
- *Finds shallowest (or nearly shallowest) solution*
- *Also works for A* (described below)*

Heuristic Search

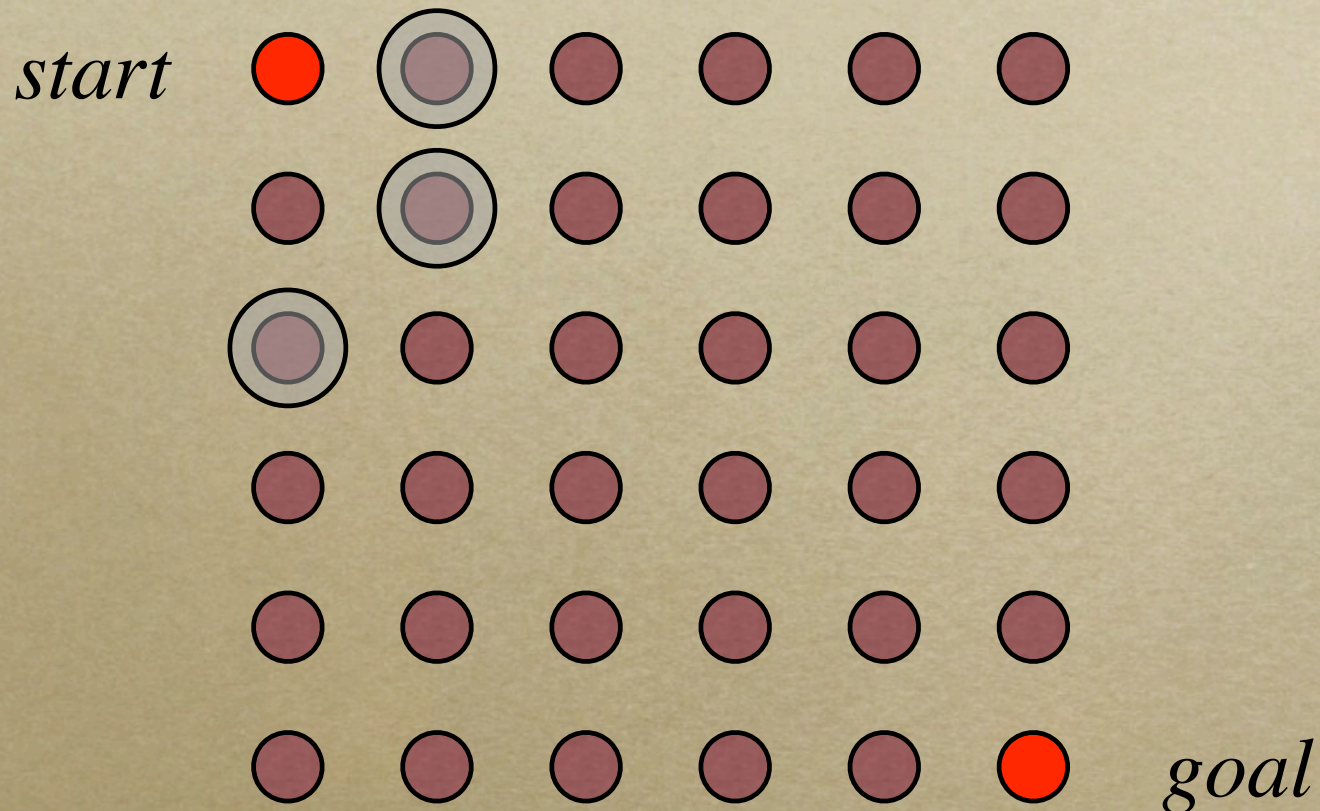
DFS looking stupid



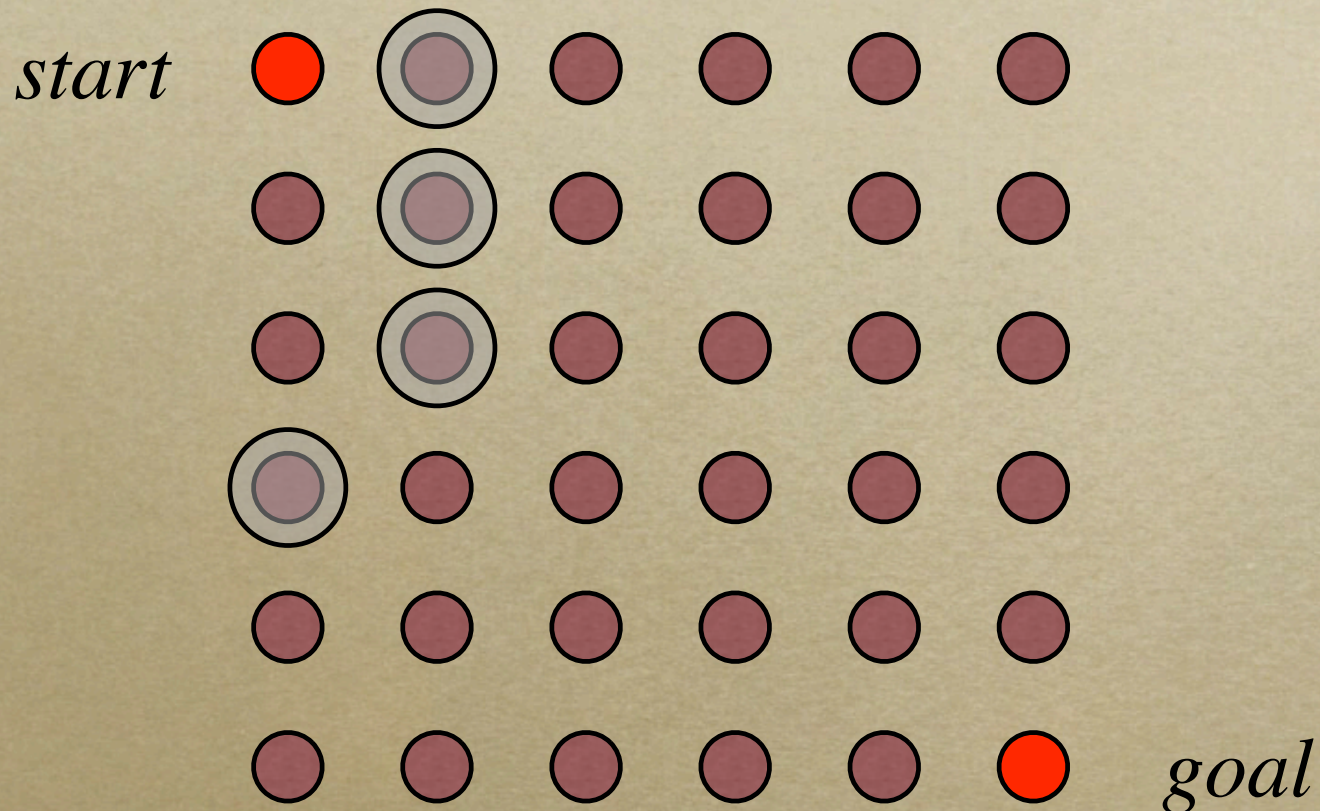
DFS looking stupid



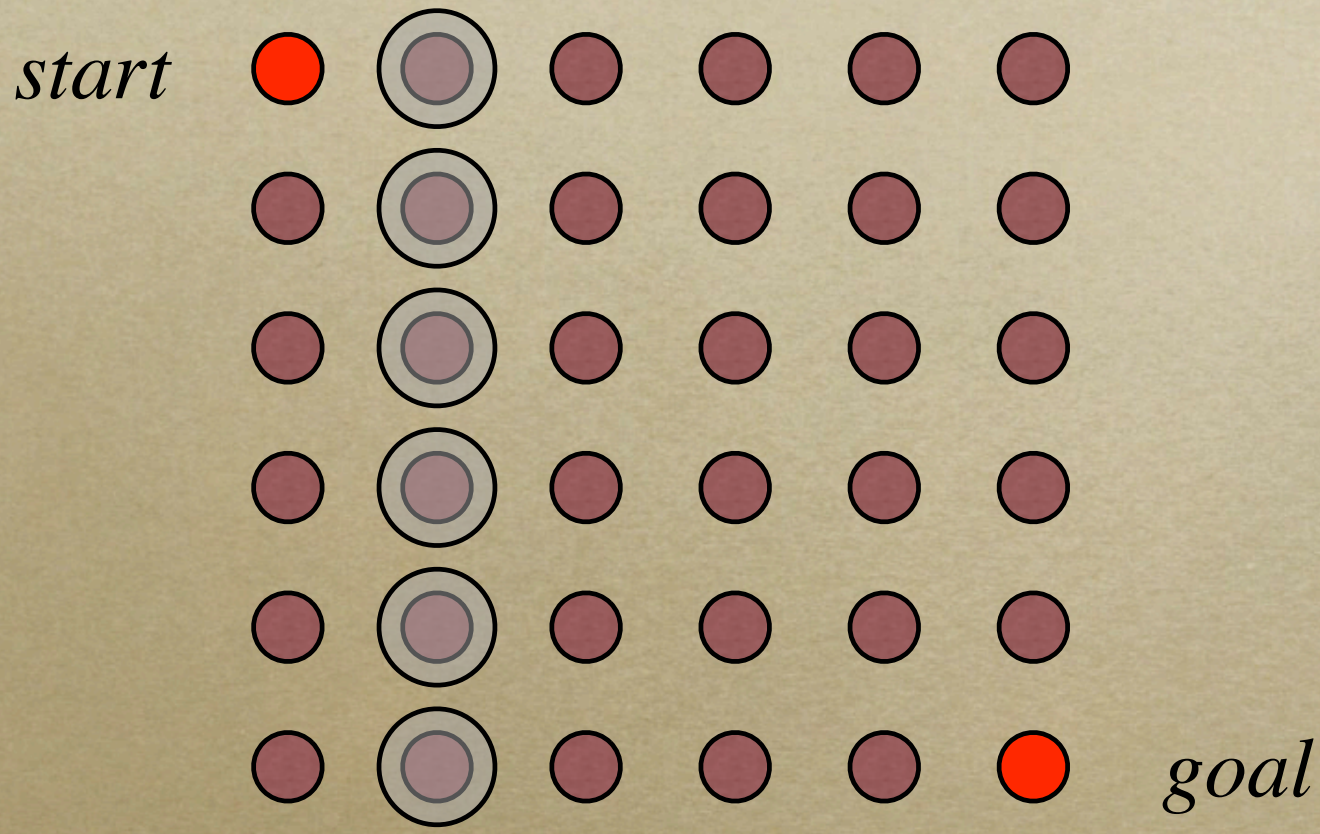
DFS looking stupid



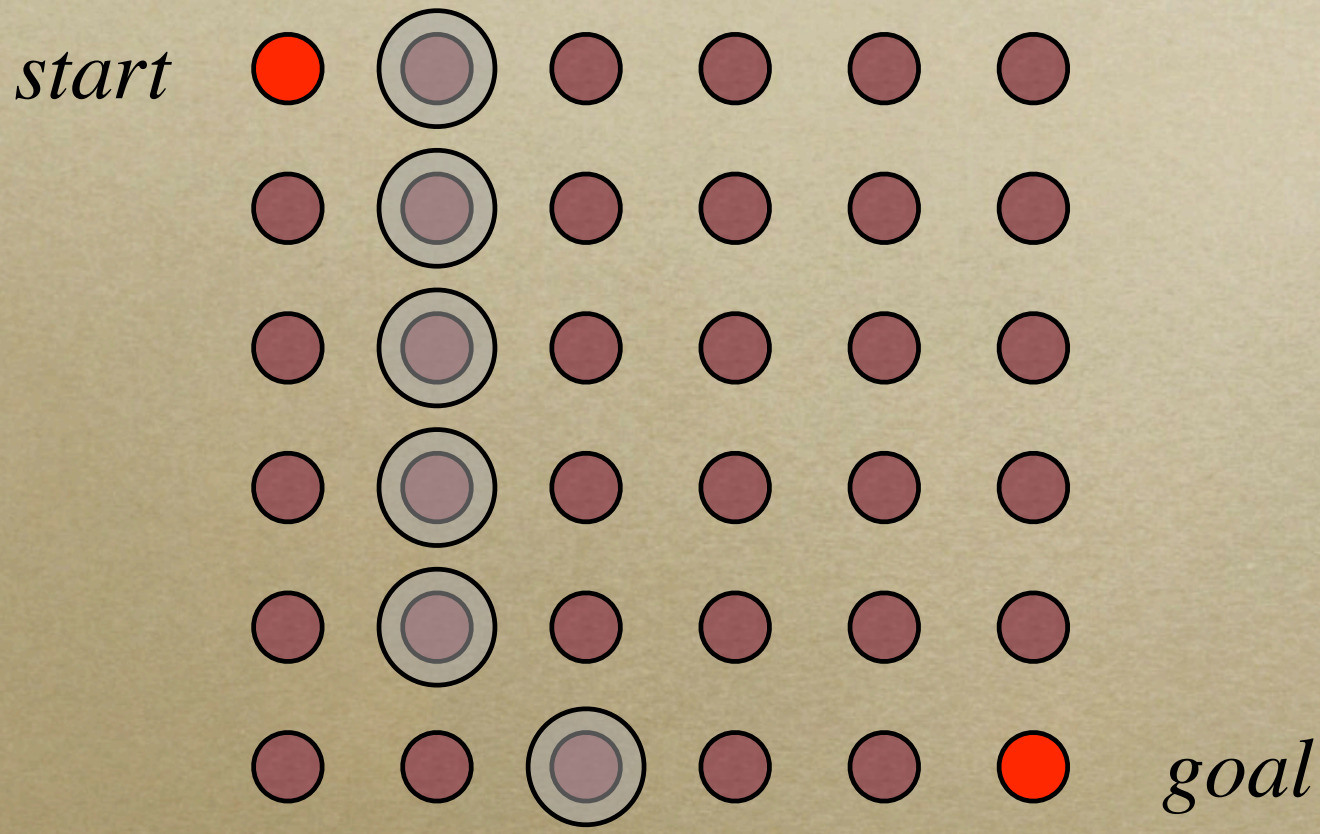
DFS looking stupid



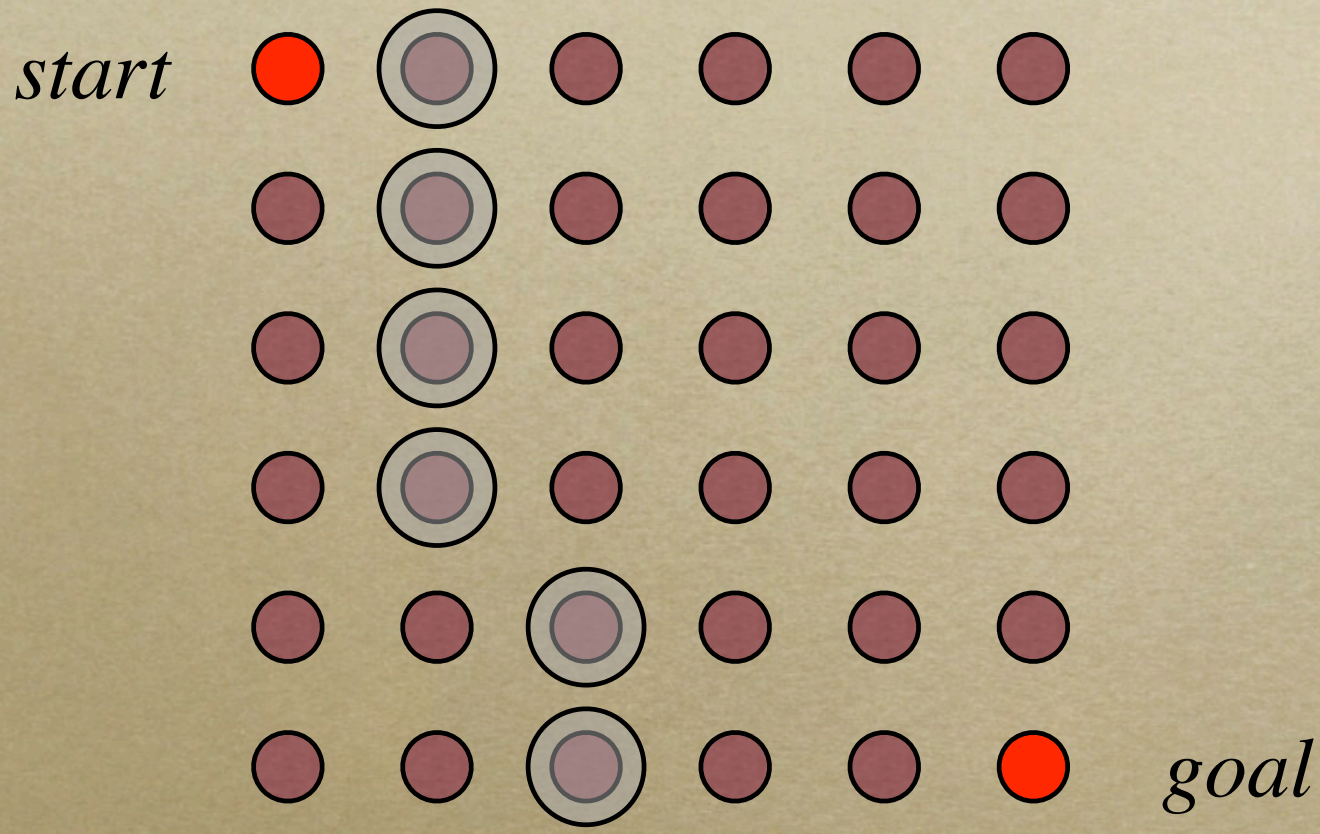
…skipping a few steps



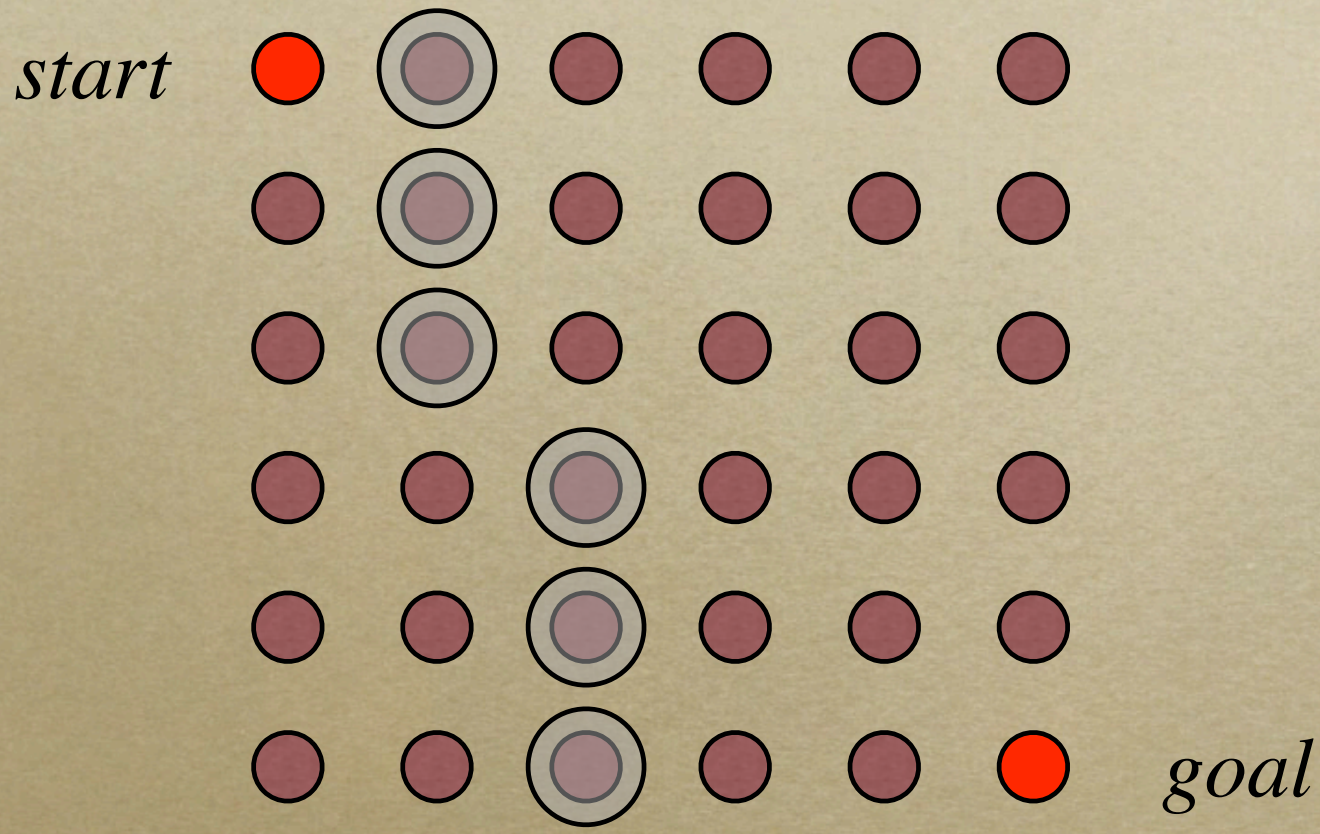
DFS looking stupid



DFS looking stupid



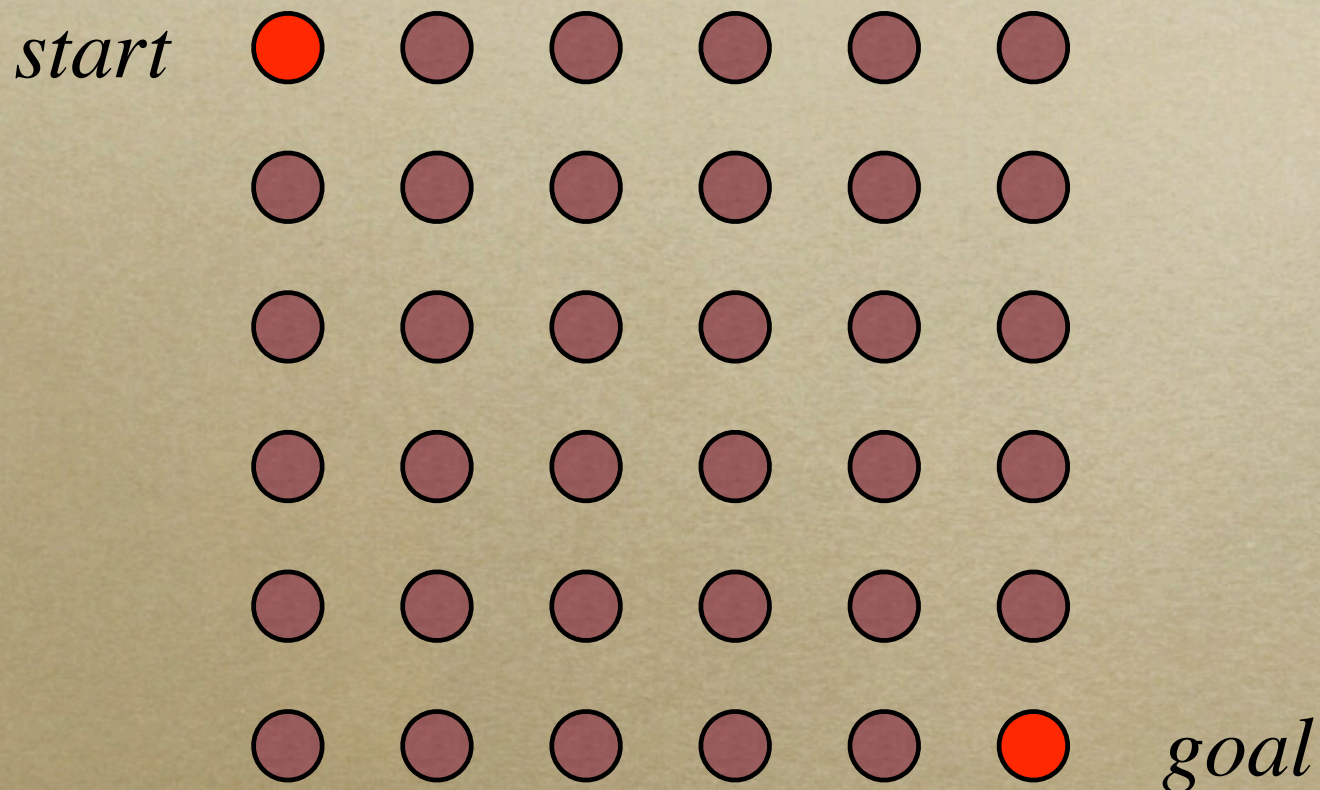
DFS looking stupid



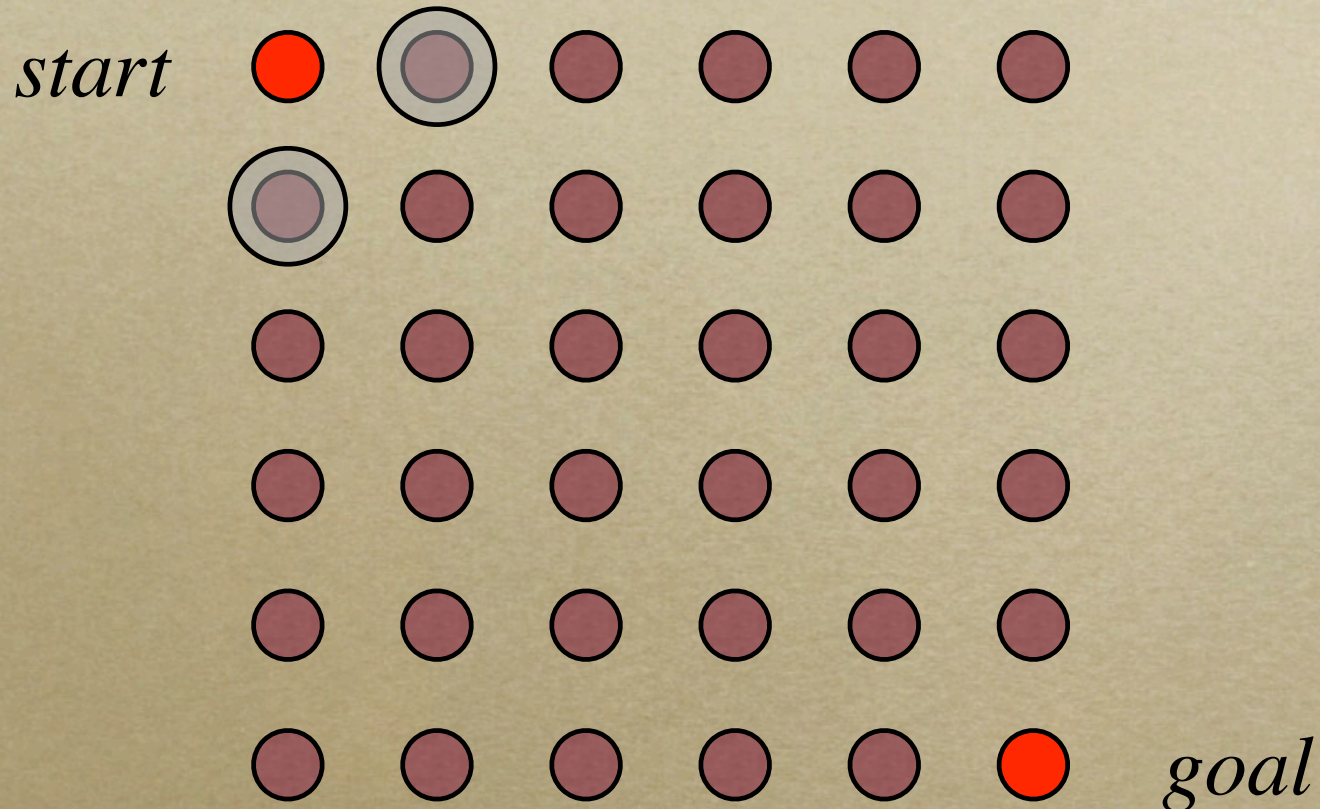
Heuristic search

- *Implement open set S with a priority queue*
  - *Ops: insert, update_priority, pop*
- *Pop always gives node w/ best priority*
- *Priority function = place to give the search algorithm additional info*
- *E.g., priority(x, y) =* $|gx-x| + |gy-y|$

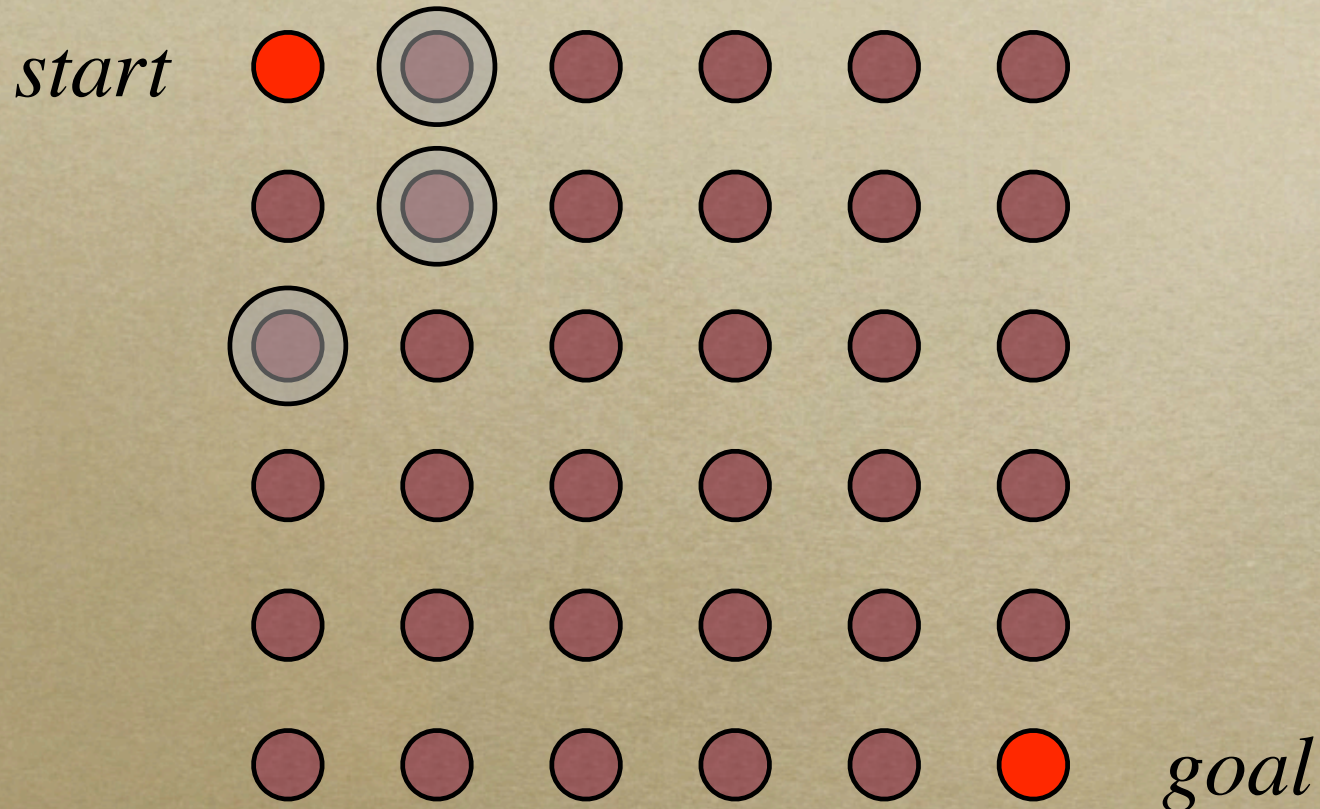
Heuristic search looking smart



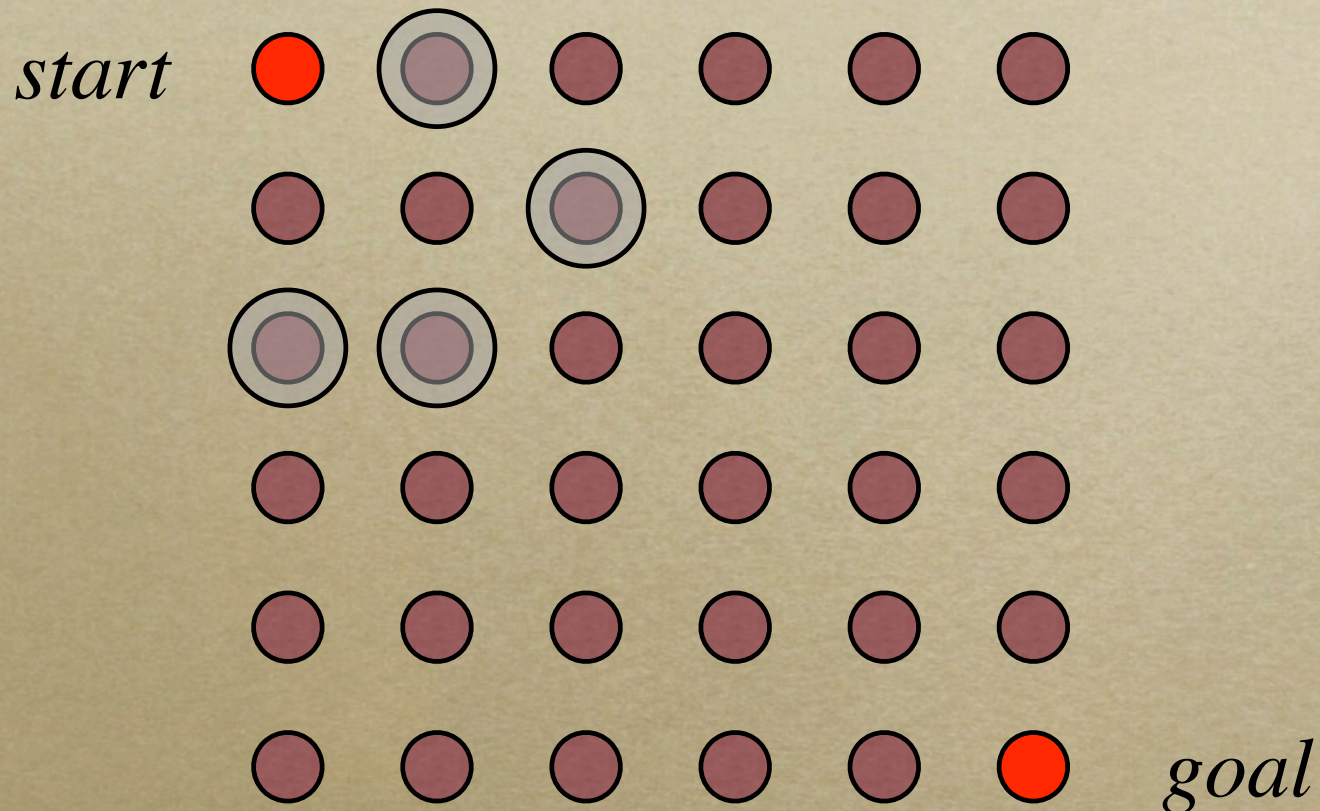
Heuristic search looking smart



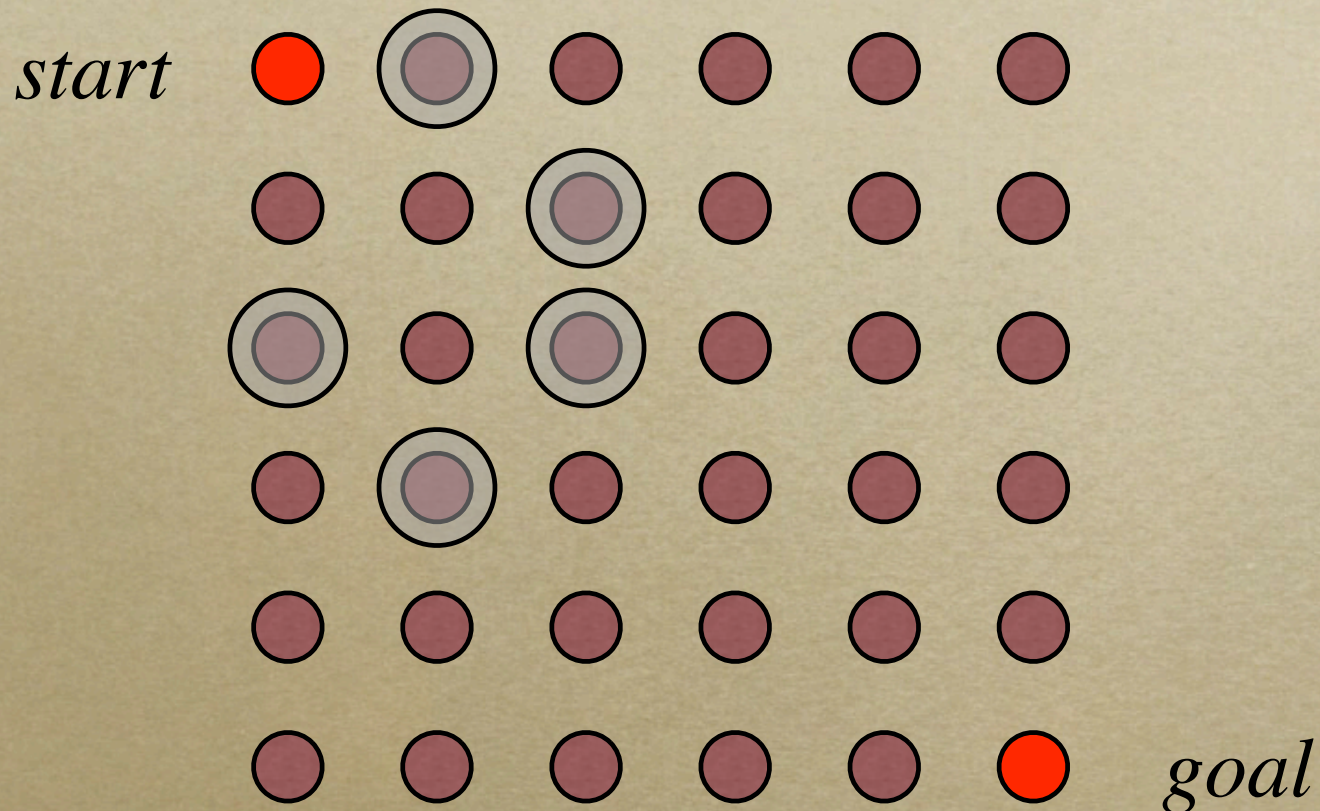
Heuristic search looking smart



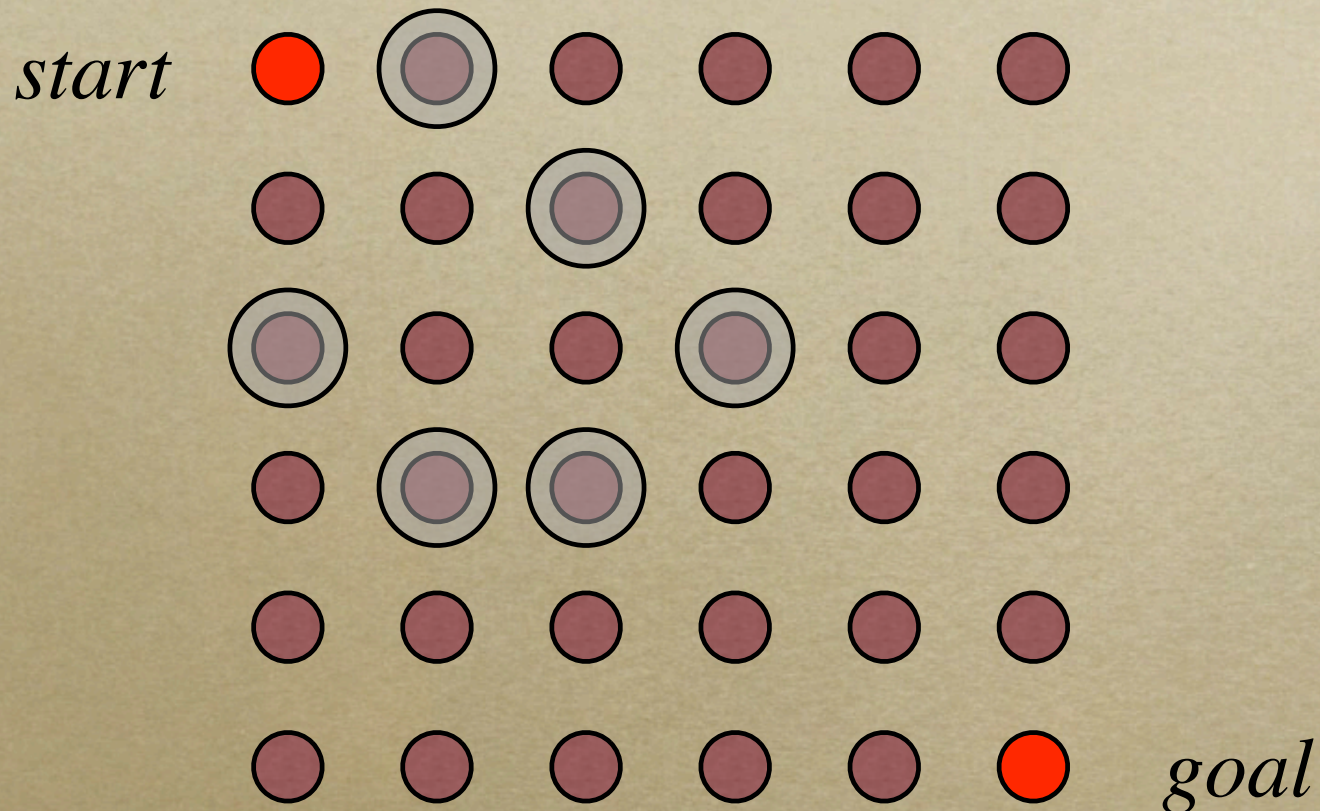
Heuristic search looking smart



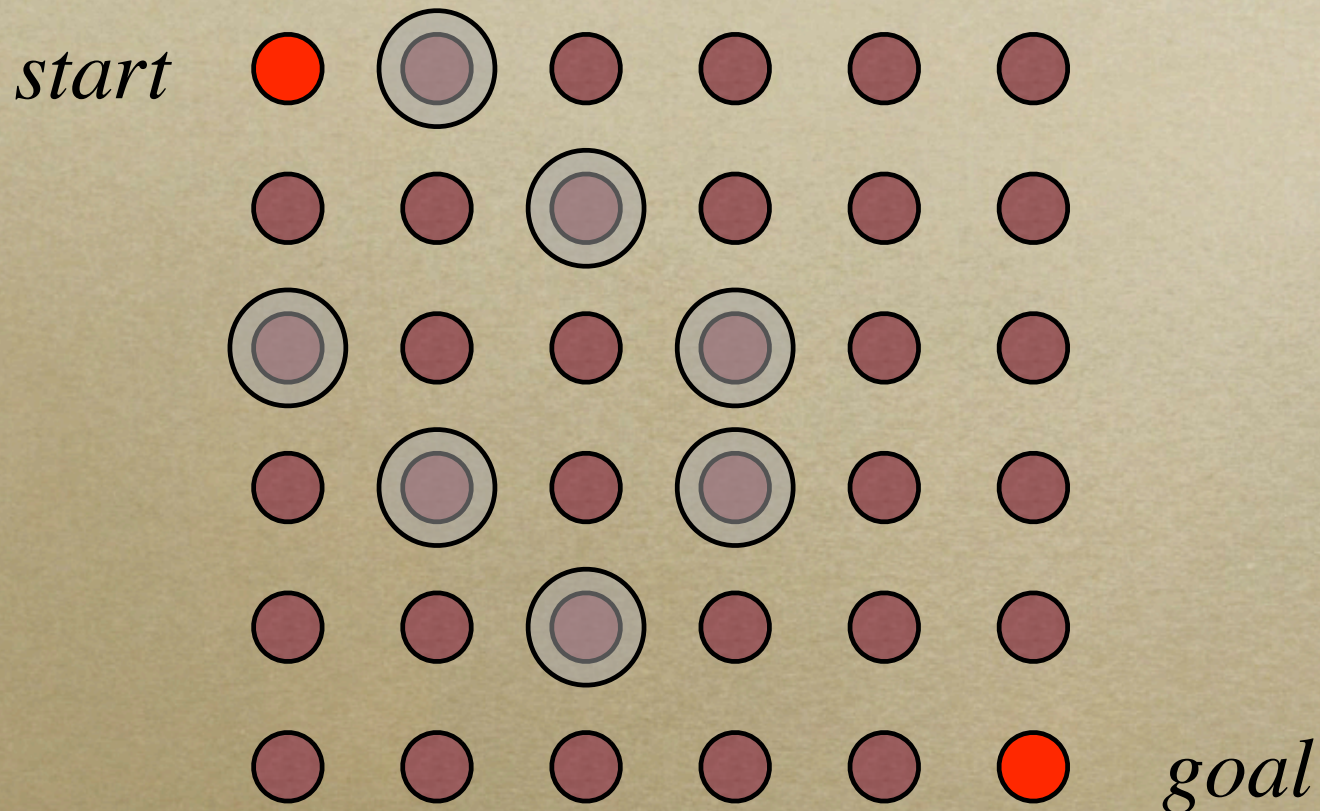
Heuristic search looking smart



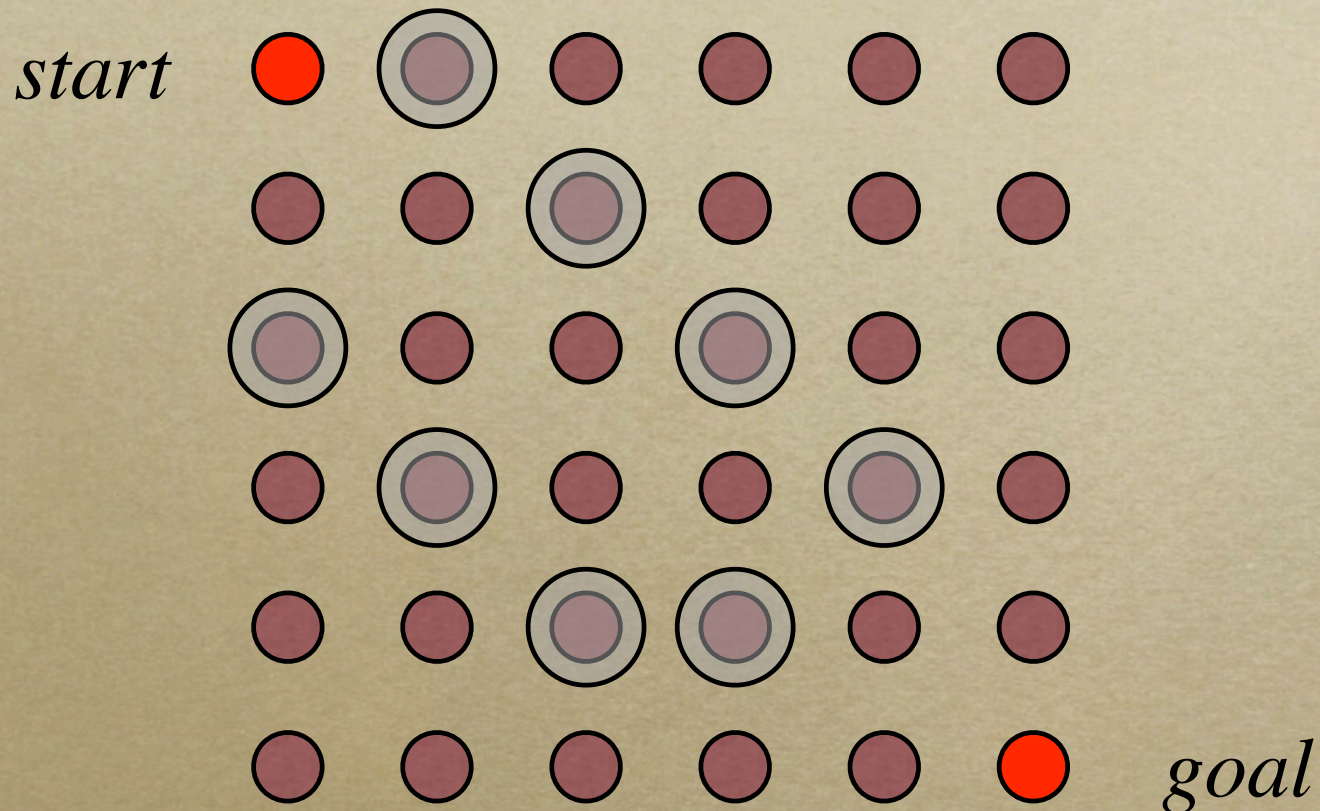
Heuristic search looking smart



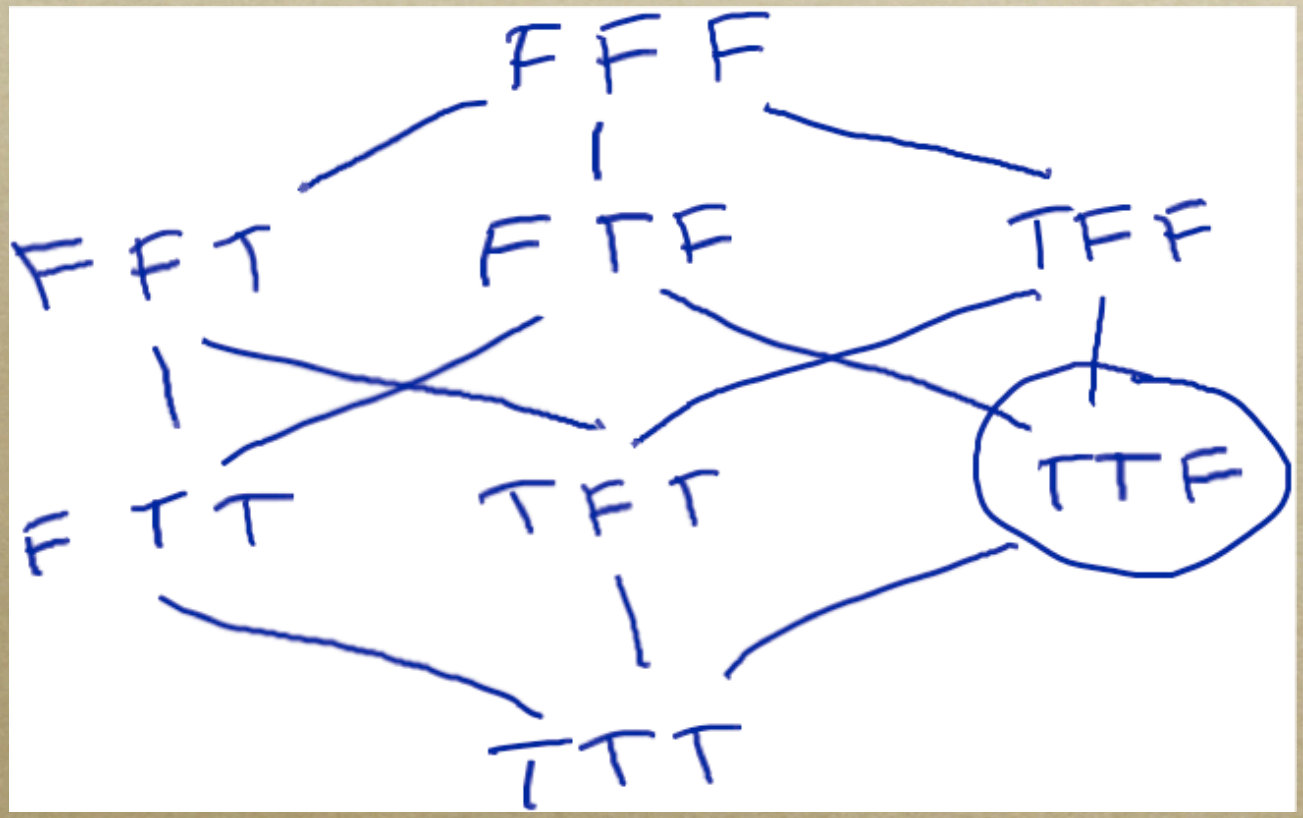
Heuristic search looking smart



Heuristic search looking smart



Question



- *What is optimal heuristic?*

Question

- *When could heuristic search look dumb?*
- *Can we find conditions we can satisfy that guarantee that it won't look dumb?*

A* Search

A* search

- *Set priority to be: f(node) =*

  (Effort so far) + (admissible heuristic)

  = g(node) + h(node)

- *Effort so far = # nodes expanded on direct path from start to node*
- *Admissible heuristic: underestimates distance to closest solution*

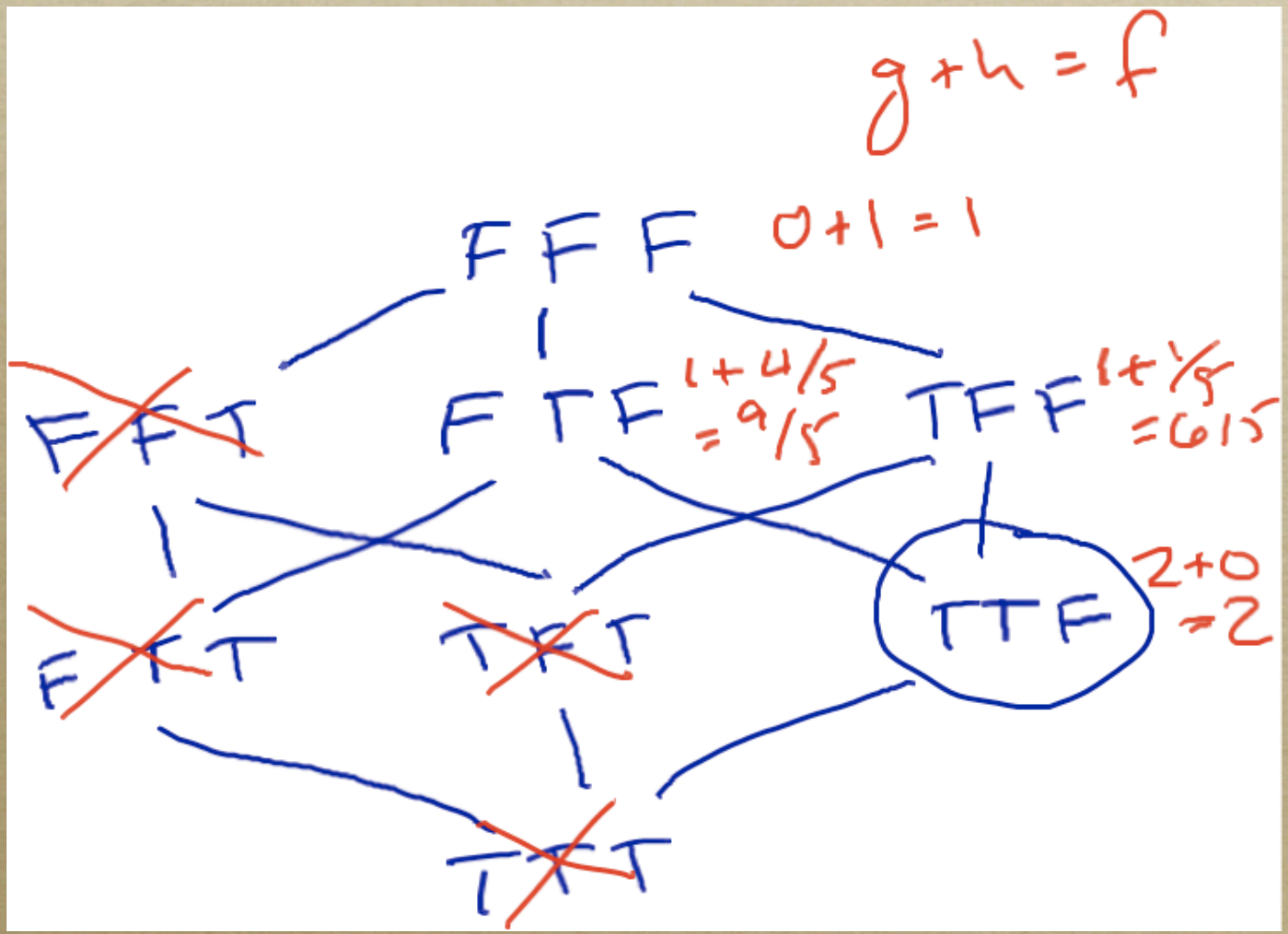
A* details

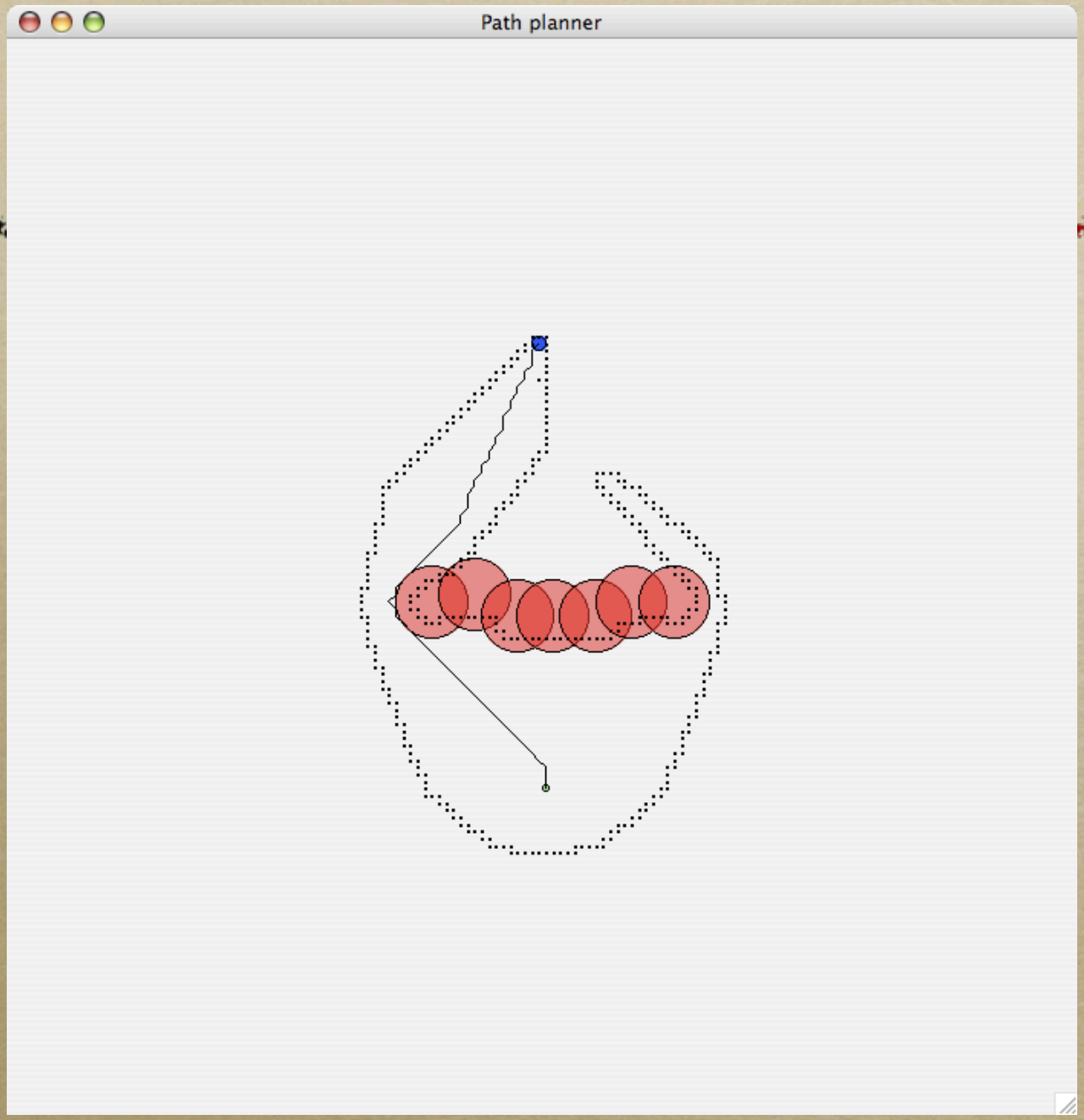
- *What if we revisit a node already on the queue?*
- *Can we terminate when we generate a goal node instead of when we expand it?*

Admissible heuristic

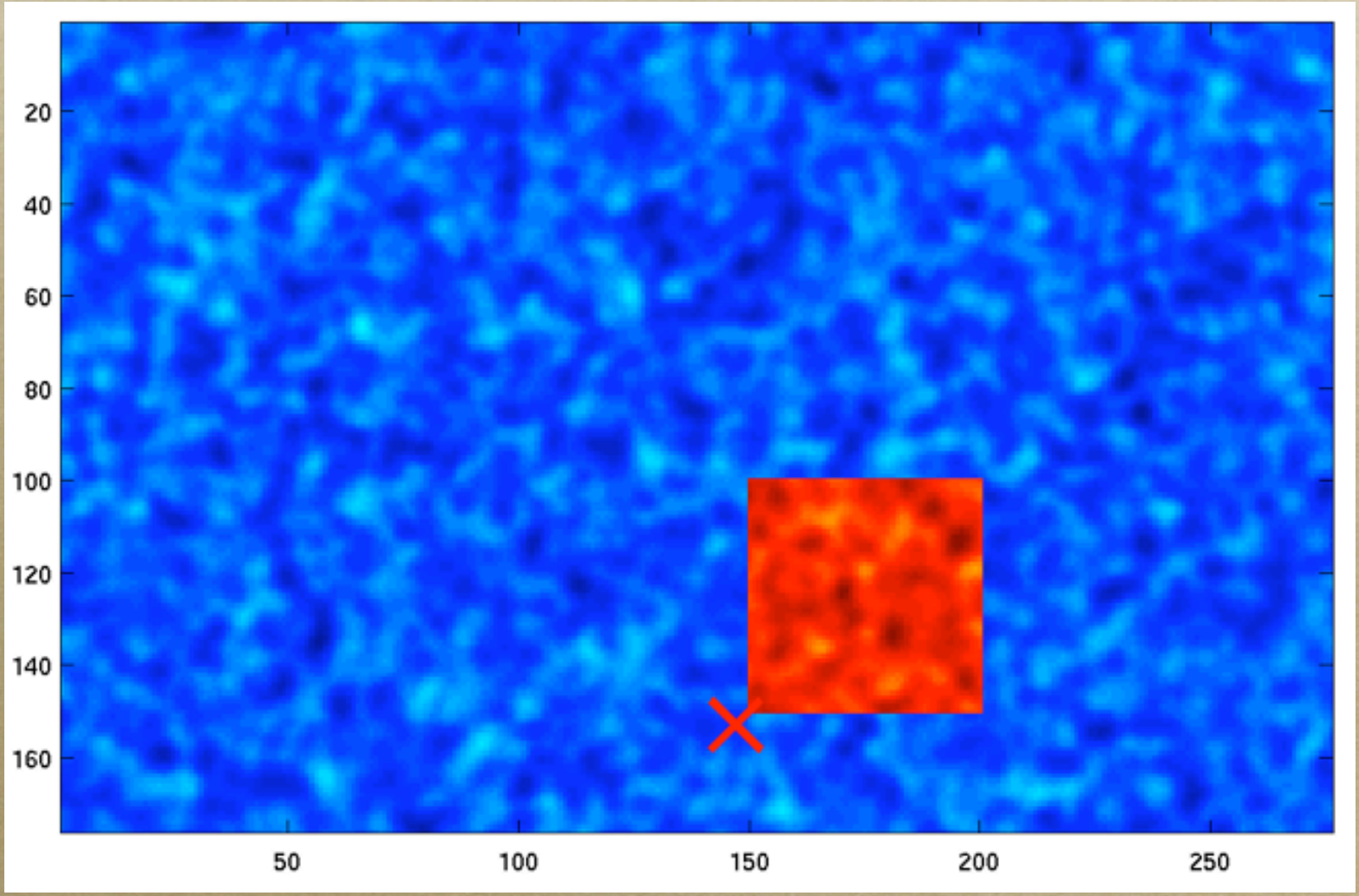
- *Do admissible heuristics exist in practical examples?*
- *Yes: e.g.,*
  - *(5 - total_progress) / 5*
  - *Crow-flies distance in path planning*

A* for robotic grad student

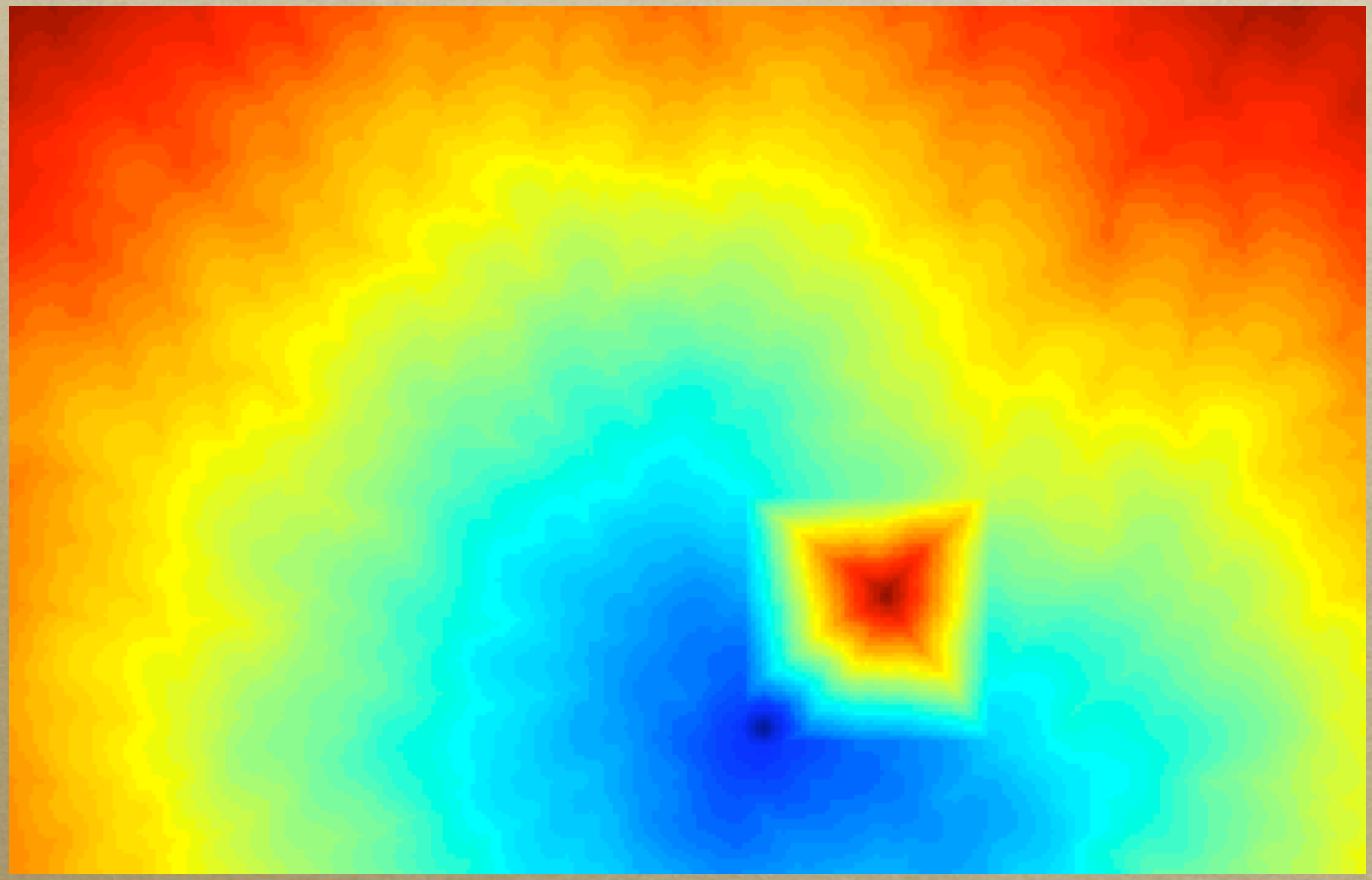



Path planner

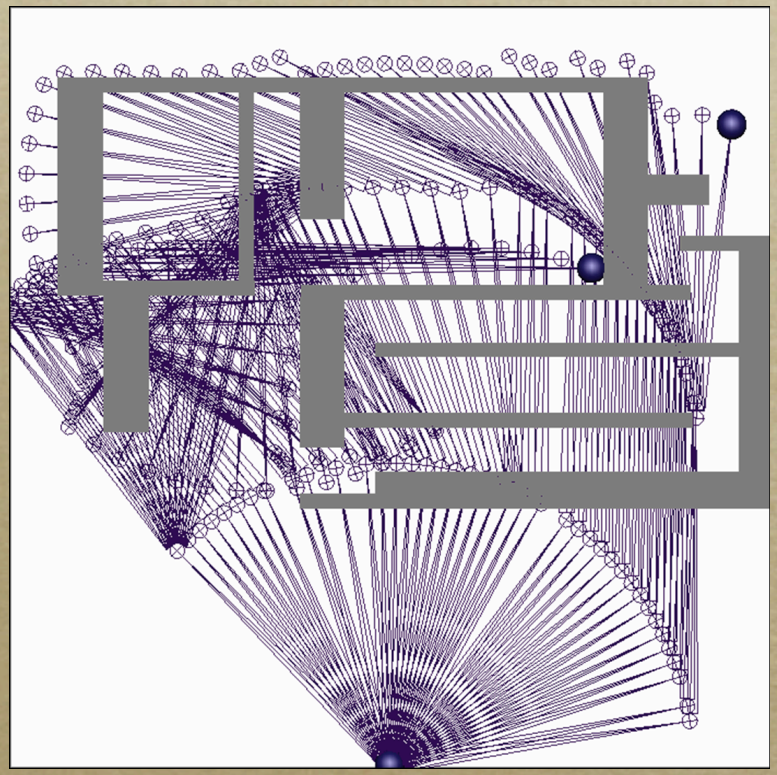
Node costs



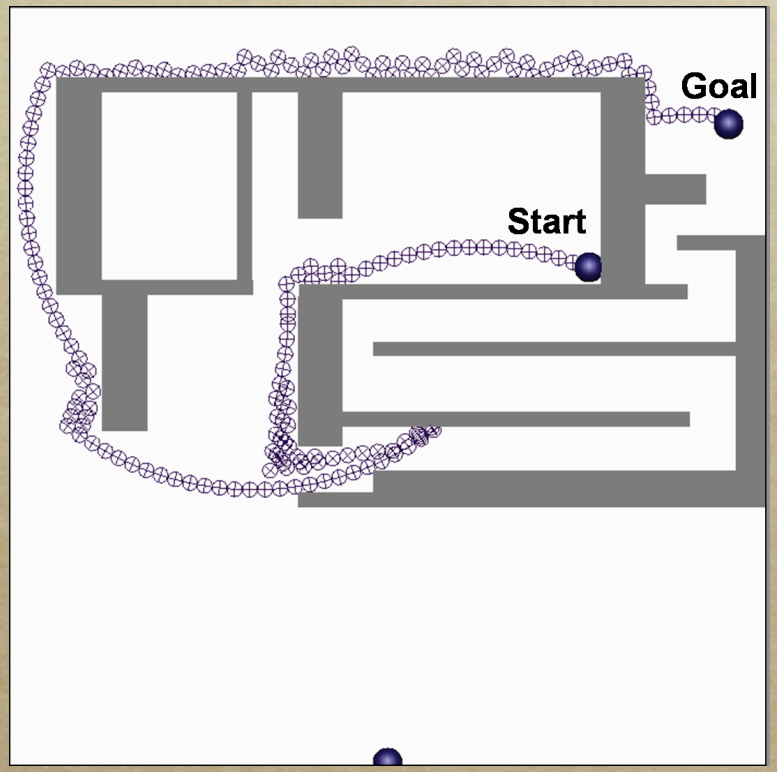
Path costs



More complicated A* example



Optimal Solution



End-effector Trajectory

A* guarantees

- *Write g* for depth of shallowest solution*
- ***(optimality)** A* finds a solution of depth g**
- ***(efficiency)** A* expands no nodes that have f(node) > g**

A* proof

- *Will do a simple case: heuristic satisfies "triangle inequality"*
- *For all neighboring pairs (x, y)*

$$h(x) \leq h(y) + c(x, y)$$

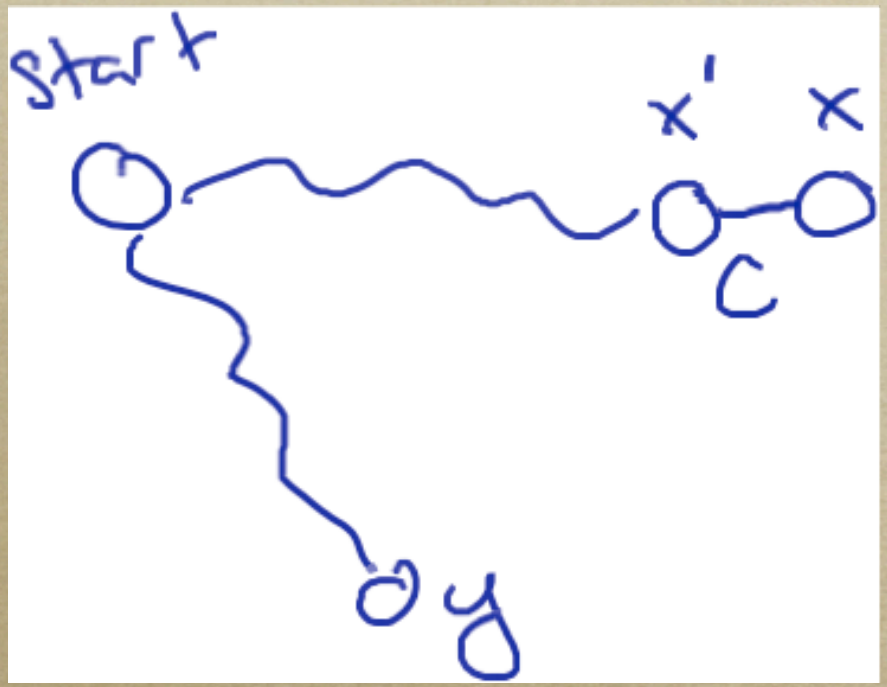
A* proof

- *Both optimality and efficiency follow from:*

  Lemma. For any two nodes x and y which have f(x) < f(y), A* expands x before y

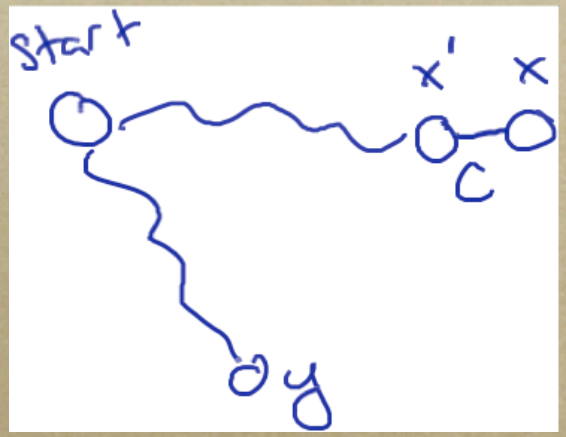
- *To see why optimality and efficiency follow, note goals have f(x) = g(x)*
  - *h(x) must be 0*

Proof of lemma



- *Suppose f(y) > f(x) but we expand y first*
- *Consider shortest path from start to x*

Proof cont'd



$$h(x') \leq h(x) + C$$

$$\begin{aligned} f(x') &= g(x') + h(x') \\ &\leq g(x') + h(x) + C \\ &= g(x) + h(x) \\ &= f(x) \end{aligned}$$

Proof cont'd

- *So, all nodes w on path to x have*

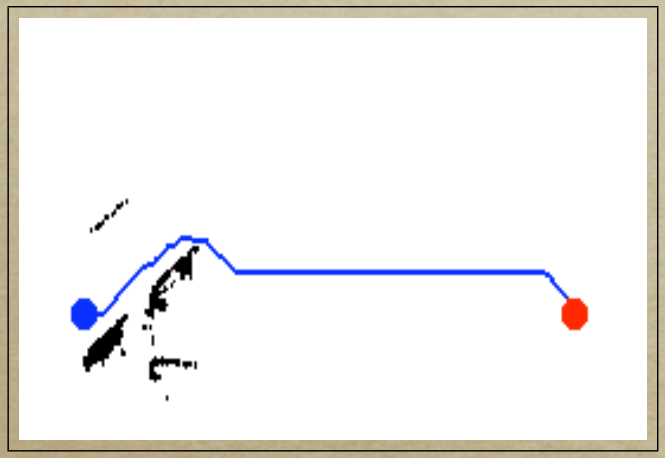
$$f(w) \leq f(x) < f(y)$$

- *At least one such w is always on queue while x has not been expanded (possibly we have w = x)*
- *So if x has not yet been expanded, we must pick w before we expand y — QED*

A* extensions

- *Suboptimal: use non-admissible heuristic, lose guarantees but maybe increase speed*
- *Anytime: start with suboptimal solution, gradually improve it*
- *Dynamic: fast replan if map changes*

Anytime, dynamic planning

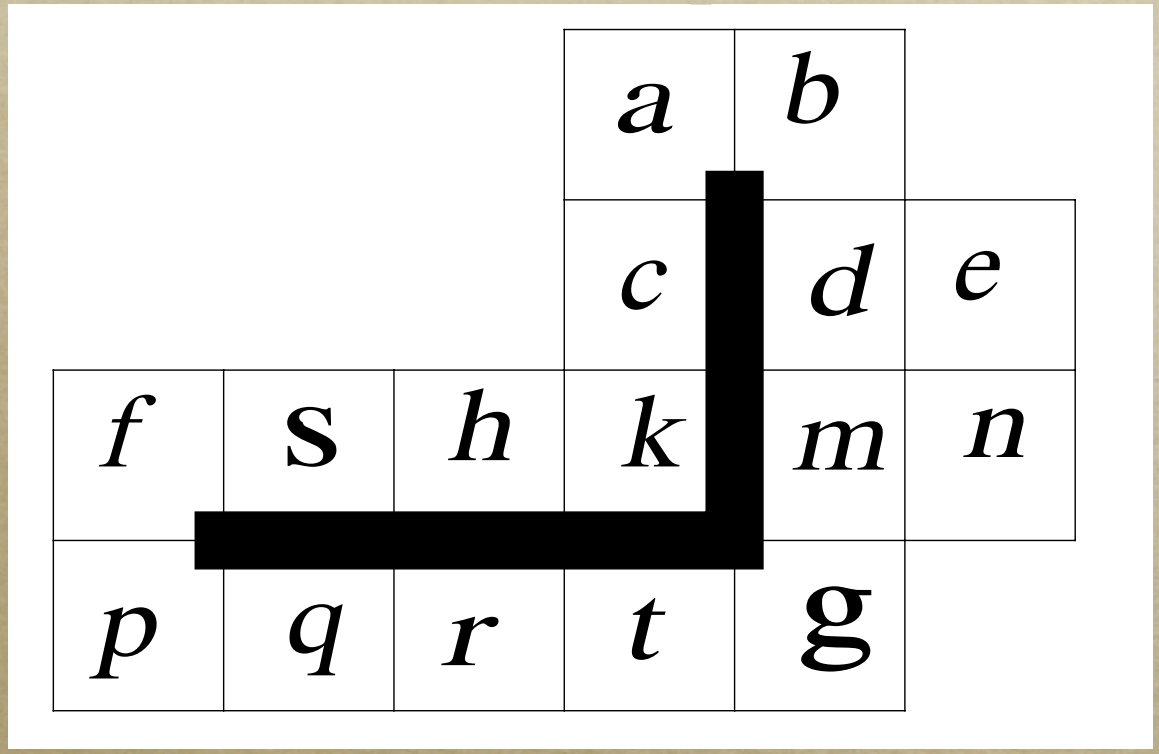


http://www.cs.cmu.edu/~ggordon/likhachev-etal.anytime-dstar.pdf

Sample exercise

- *In graph on next page, to find a path from s to g, what is the expansion order for*
  - *DFS, BFS*
  - *Heuristic search using h = Manhattan*
  - *A* using f = g + h*
- *Assume we can detect when we reach a node via two different paths, and avoid duplicating it on the queue*

Graph for exercise



credit: Andrew Moore

Nodes are connected in 4 cardinal directions, except across dark line