CMSC 471
Fall 2012

Class #5

Thu 9/13/12
Informed Search

Kevin Winner, winnerk1@cs.umbc.edu

Informed Search

Sections 3.5 and 3.6

Outline

- Heuristic search
- Best-first search
  - Greedy search
  - Beam search
  - A, A*
  - Examples
- Memory-conserving variations of A*
- Heuristic functions

Heuristic

Merriam-Webster's Online Dictionary

Heuristic (pron. \hy*u*-ˈris-tik\): adj. [from Greek *heuriskein* to discover.] involving or serving as an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods

The Free On-line Dictionary of Computing (15Feb98)

heuristic 1. <programming> A rule of thumb, simplification or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood. Unlike algorithms, heuristics do not guarantee feasible solutions and are often used with no theoretical guarantee. 2. <algorithm> approximation algorithm.

From WordNet (r) 1.6

heuristic adj 1: (computer science) relating to or using a heuristic rule 2: of or relating to a general formulation that serves to guide investigation [ant: algorithmic] n : a commonsense rule (or set of rules) intended to increase the probability of solving some problem [syn: heuristic rule, heuristic program]

Heuristics

- Adding domain knowledge and heuristics gives us **informed search** or **heuristic search**
- All domain knowledge used in the search is encoded in the **heuristic function** ***h*****()**.
- Define a heuristic function ***h*****(*****n*****)** that estimates the "goodness" of a node *n*.
  - Specifically, *h*(*n*) = **estimated cost** (or distance) of minimal cost path from *n* **to a goal state**.
- The heuristic function is an estimate of how close we are to a goal, based on domain-specific information that is computable from the current state description.

Properties of Heuristic Functions

- In general:
  - $h(n) \geq 0$ for all nodes n
  - $h(n) = 0$ implies that n is a goal node
  - $h(n) = \infty$ implies that n is a dead-end that can never lead to a goal
- $h^*(n)$ is the perfect heuristic
  - $h^*(n)$ is the true shortest path value from n to the goal
- iff $h(n) < h^*(n)$ for all n then $h(n)$ is **admissible**
  - in other words $h(n)$ always underestimates the true distance
  - whenever possible, we'd like to develop admissible heuristics

Examples

- Missionaries and Cannibals: Number of people on starting river bank
- 8-puzzle: Number of tiles out of place
- 8-puzzle: Sum of distances each tile is from its goal position

Examples

- Missionaries and Cannibals: Number of people on starting river bank
- 8-puzzle: Number of tiles out of place
- 8-puzzle: Sum of distances each tile is from its goal position

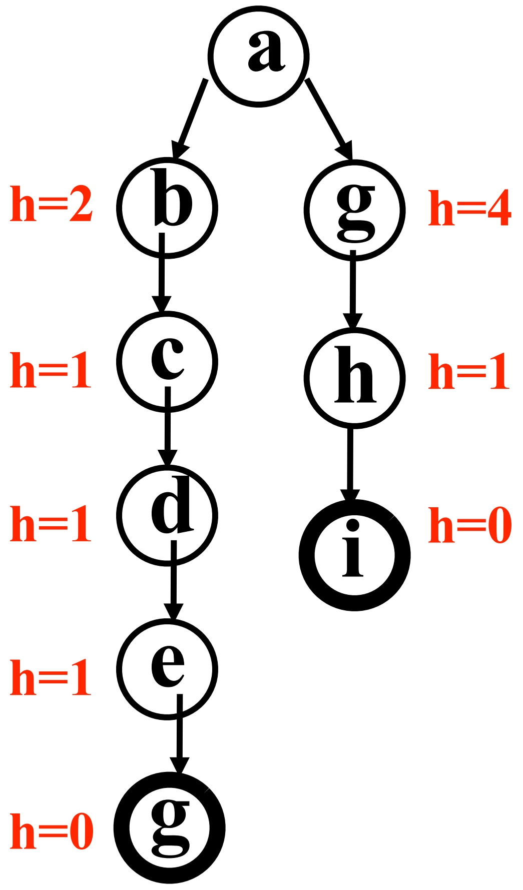
If $h_1(n) < h_2(n) < h^*(n)$ for all n, then $h_2(n)$ is better than $h_1(n)$

Best-First Search

- Order nodes on the nodes list by increasing value of an evaluation function ***f*(*n*)**
  - $f(n)$ incorporates domain-specific information in some way.
- This is a generic way of referring to the class of informed methods.
  - We get different searches depending on the evaluation function $f(n)$

Greedy Search

- Use as an evaluation function $f(n) = h(n)$, sorting nodes by increasing values of f.
- Selects node to expand believed to be **closest** (hence "greedy") to a goal node (i.e., select node with smallest f value)
- Not admissible, as in the example.
  - Assuming all arc costs are 1, then greedy search will find goal g, which has a solution cost of 5.
  - However, the optimal solution is the path to goal I with cost 3.
- Not complete



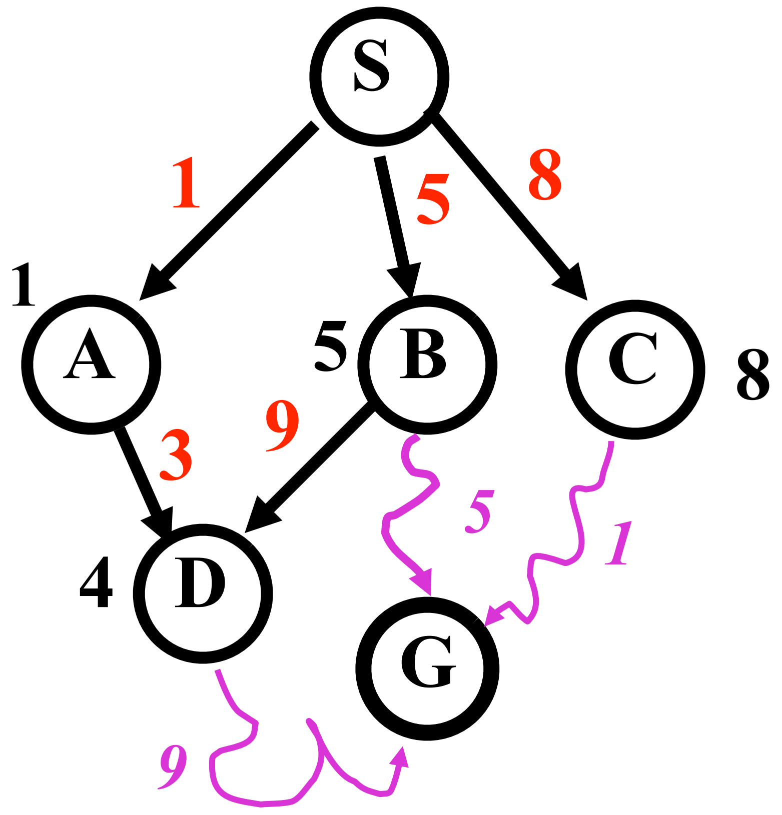
Beam Search

- Use an evaluation function $f(n) = h(n)$, but the maximum size of the nodes list is k, a fixed constant
- Only keeps k best nodes as candidates for expansion, and throws the rest away
- More space efficient than greedy search, but may throw away a node that is on a solution path
- Not complete
- Not admissible

Algorithm A

- Use as an evaluation function

  $\boldsymbol{f(n) = g(n) + h(n)}$
- $g(n)$ = path cost from the start state to state n.
- The $g(n)$ term adds a “breadth-first” component to the evaluation function.
- Ranks nodes on search frontier by estimated cost of solution from start node through the given node to goal.
- Not complete if $h(n)$ can equal infinity.
- Not admissible.



g(d)=4

h(d)=9

C is chosen next to expand

Algorithm A

1. Put the start node S on the nodes list, called OPEN
2. If OPEN is empty, exit with failure
3. Select node in OPEN with minimal $f(n)$ and place on CLOSED
4. If n is a goal node, collect path back to start and stop.
5. Expand n, generating all its successors and attach to them pointers back to n. For each successor n' of n
    1. If n' is not already on OPEN or CLOSED
        - put n' on OPEN
        - compute $h(n')$, $g(n') = g(n) + c(n,n')$, $f(n') = g(n') + h(n')$
    2. If n' is already on OPEN or CLOSED and if $g(n')$ is lower for the new version of n', then:
        - Redirect pointers backward from n' along path yielding lower $g(n')$.
        - Put n' on OPEN.

Algorithm A*

- Algorithm A with constraint that $\boldsymbol{h(n) \leq h^*(n)}$
  - $h(n)$ is **admissible** when $h(n) \leq h^*(n)$ holds.
- Using an admissible heuristic guarantees that the first solution found will be an optimal one.
- A* is **complete** whenever the branching factor is finite, and every operator has a fixed positive cost
- A* is **admissible**

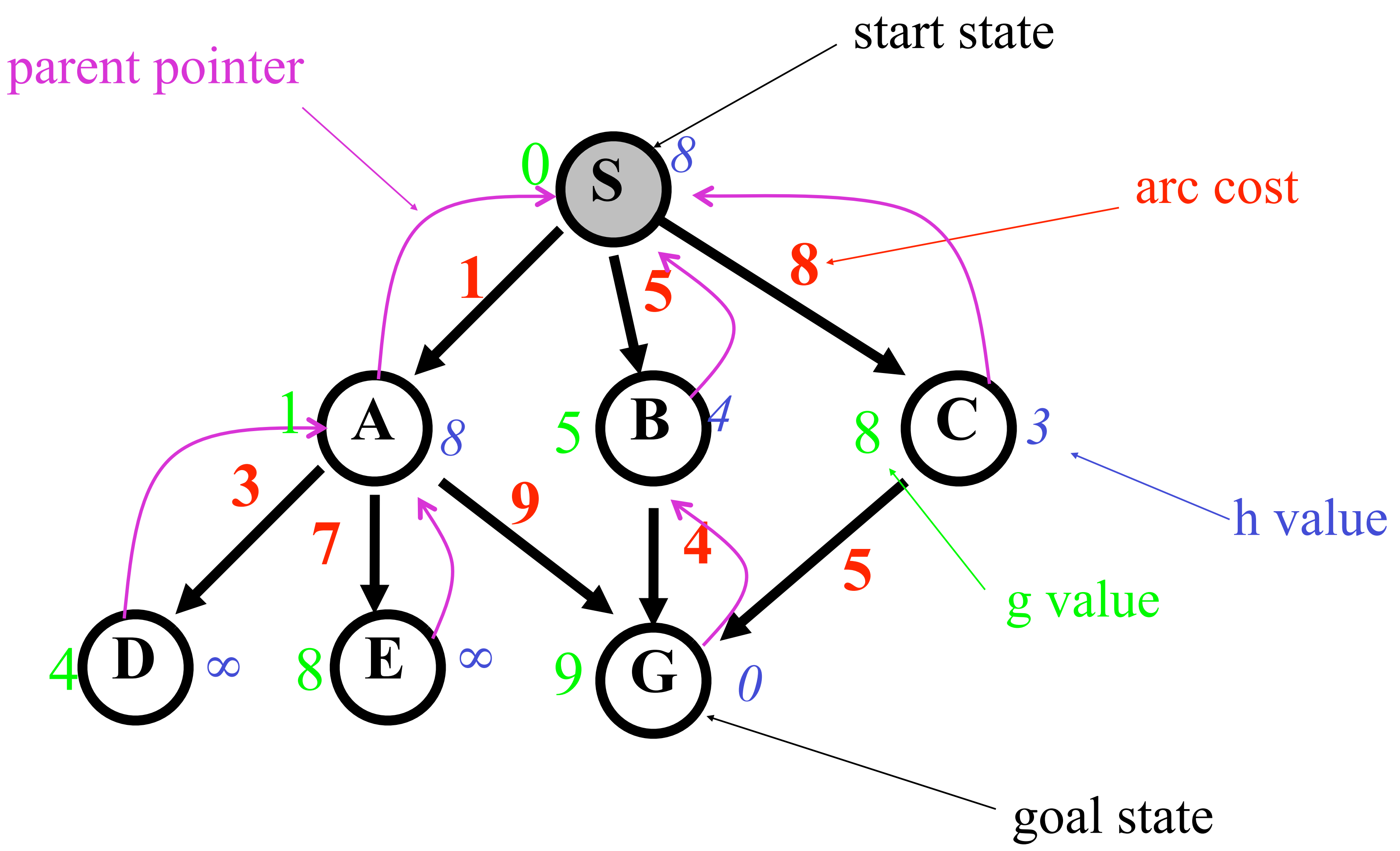
Some Observations on A/A*

- **Perfect heuristic:** If $h(n) = h^*(n)$ for all n, then only the nodes on the optimal solution path will be expanded. So, no extra work will be performed.
- **Null heuristic:** If $h(n) = 0$ for all n, then this is an admissible heuristic and A* acts like Uniform-Cost Search.
- **Better heuristic:** If $h_1(n) < h_2(n) \leq h^*(n)$ for all non-goal nodes, then h_2 is a better heuristic than h_1
  - The closer h is to h^*, the fewer extra nodes that will be expanded

Effect of a Better Heuristic

	Search Cost (nodes generated)		
d	**IDS**	**A*(h_1)**	**A*(h_2)**
2	10	6	6
4	112	13	12
6	680	20	18
8	6384	39	25
10	47127	93	39
12	3644035	227	73
14		539	113
16		1301	211
18		3056	363
20		7276	676
22		18094	1219
24		39135	1641

Example Search Space Revisited



In-Class Example

n	***g*(*n*)**	***h*(*n*)**	***f*(*n*)**	***h*(n)***
S	0	8	8	9
A	1	8	9	9
B	5	4	9	4
C	8	3	11	5
D	4	inf	inf	inf
E	8	inf	inf	inf
G	9	0	9	0

- $h^*(n)$ is the (hypothetical) perfect heuristic.
- Since $h(n) \leq h^*(n)$ for all n, h is admissible
- Optimal path = S B G with cost 9.

Greedy Search

$f(n) = h(n)$

```
node expanded     nodes list
                { S(8) }
     S          { C(3) B(4) A(8) }
     C          { G(0) B(4) A(8) }
     G          { B(4) A(8) }
```

- Solution path found is S C G, 3 nodes expanded.

A* Search

$f(n) = g(n) + h(n)$

```
node exp.     nodes list
          { S(8) }
 S        { A(9) B(9) C(11) }
 A        { B(9) G(10) C(11) D(inf) E(inf) }
 B        { G(9) G(10) C(11) D(inf) E(inf) }
 G        { C(11) D(inf) E(inf) }
```

- Solution path found is S B G, 4 nodes expanded..
- Still pretty fast. And optimal, too.

Dealing with Hard Problems

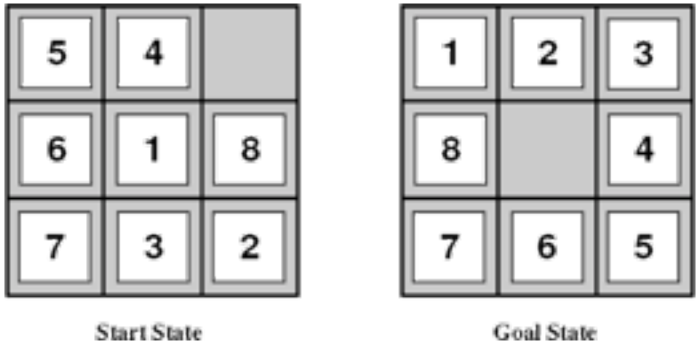
- For large problems, A* often requires too much space.
- Three variations conserve memory: IDA*, SMA*, RBFS
- IDA* -- iterative deepening A*
  - uses successive iteration with growing limits on *f*. For example,
    - A* but don't consider any node *n* where *f* (*n*) > 10
    - A* but don't consider any node *n* where *f* (*n*) > 20
    - A* but don't consider any node *n* where *f* (*n*) > 30, ...
- SMA* -- Simplified Memory-Bounded A*
  - uses a queue of restricted size to limit memory use.
  - throws away the "oldest" worst solution.
- RBFS -- Recursive Best First Search
  - Tracks the best *alternative* path
  - When the currently expanding path is worse than the alternative, the current path is thrown away and the knowledge is backed up to the alternative

What’s a Good Heuristic?

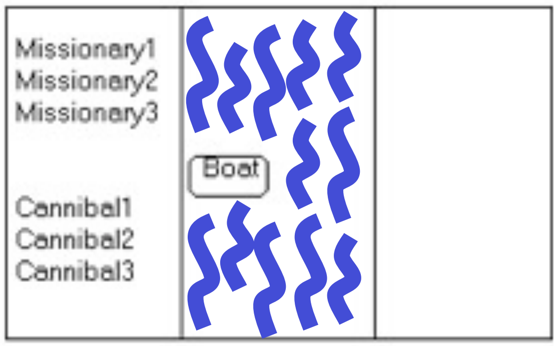
- If $h_1(n) < h_2(n) \leq h^*(n)$ for all n, h_2 is better than (**dominates**) h_1.
- **Relaxing the problem:** remove constraints to create a (much) easier problem; use the solution cost for this problem as the heuristic function
- **Combining heuristics:** take the max of several admissible heuristics: still have an admissible heuristic, and it’s better!
- **Subgoals:** restructure the problem in abstract subgoals and use search in this easier domain as a heuristic
- **Learning:** very similar to reinforcement learning (chapter 21)
  - based on experience solving similar problems, compare states in the current problem to ones seen historically and estimate based on prior performance

In-class Exercise: Creating Heuristics

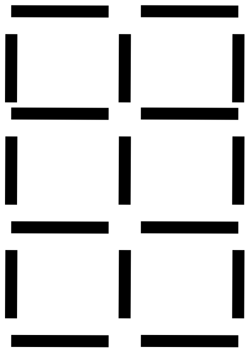
8-Puzzle



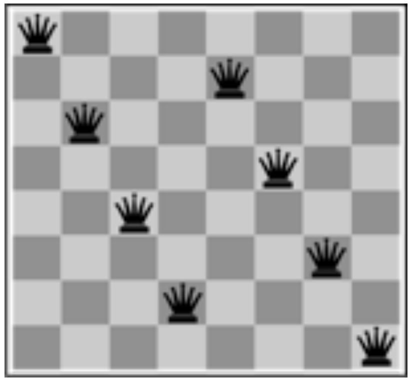
Missionaries and Cannibals



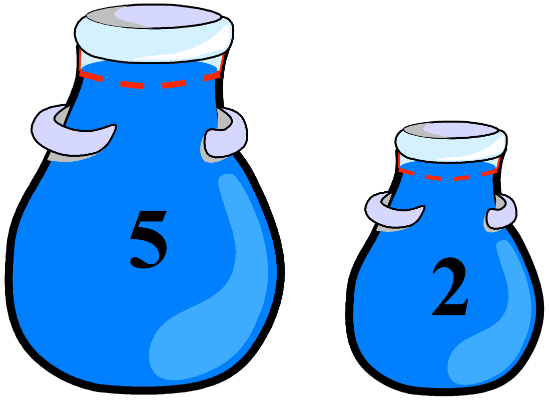
Remove 5 Sticks



N-Queens



Water Jug Problem



Route Planning

North Sea
NETHERLANDS
Zeebrugge
Oostende
Brugge
Antwerp
Gent
Mechelen
Kortrijk
Aalst
Leuven
BRUSSELS
GER.
Liège
Mons
Charleroi
Namur
Bastogne
LUX.
FRANCE