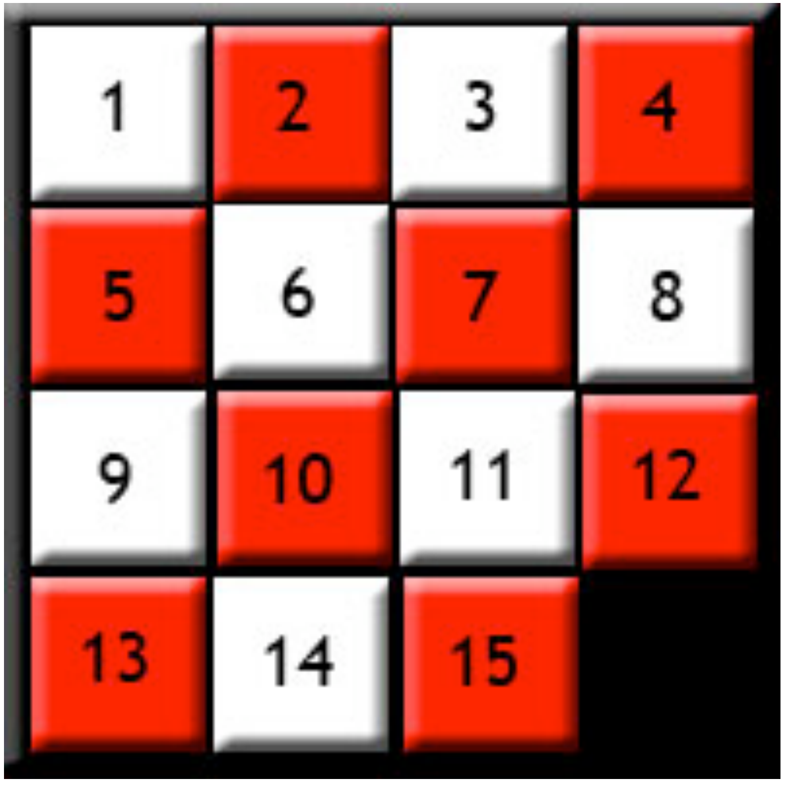
Heuristic Single-Agent Search

Robert Holte

A Heuristic Function Estimates Distance to Goal

Heuristics Speed up Search



10,461,394,944,000 states

heuristic search examines 36,000

Example Heuristic Functions

- $h(n)$ estimates cost of cheapest path from node n to goal node
- Example: 8-puzzle

5		8
4	2	1
7	3	6

n

1	2	3
4	5	6
7	8	

goal

$h_1(n)$ = number of misplaced tiles
= 6

Example Heuristic Functions

- $h(n)$ estimates cost of cheapest path from node n to goal node
- Example: 8-puzzle

5		8
4	2	1
7	3	6

n

1	2	3
4	5	6
7	8	

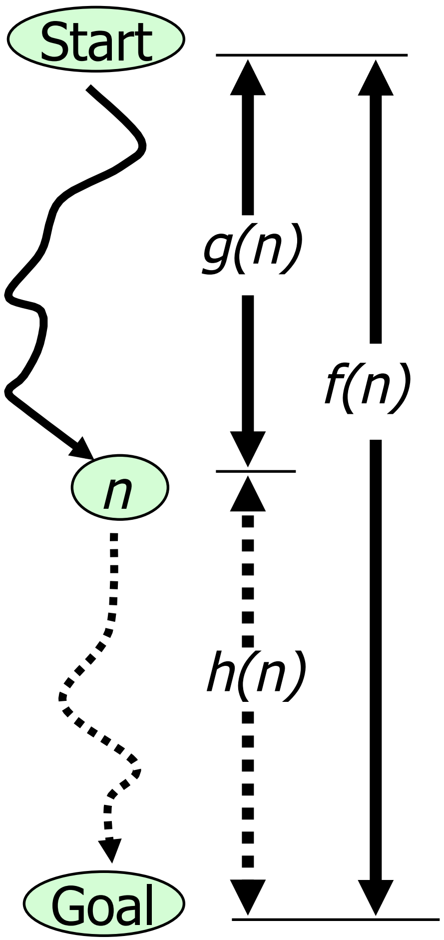
goal

$h_1(n)$ = number of misplaced tiles
= 6

$h_2(n)$ = sum of the distances of every tile to its goal position
= 3 + 1 + 3 + 0 + 2 + 1 + 0 + 3
= 13

How to Use a Heuristic Function to Speed Up Search ?

Notation



- g(n) = distance from start to node n along our current path (not necessarily optimal)
- h(n) = estimated distance from n to goal
- f(n) = g(n)+h(n) = estimated distance from start to goal via n (using our current path to n)

More Notation

- $g^*(n)$ = true distance from start to node n
- $h^*(n)$ = true distance from n to goal
- $f^*(n)$ = true distance from start to goal via n

  $= g^*(n)+h^*(n)$

How to Use a Heuristic Function to Speed Up Search ?

(1) Pruning

- If we have a bound B on total solution cost (e.g. iterative deepening) we can prune n from the search space if $f(n) > B$.
- But this might prune all the optimal paths. One condition that makes pruning safe is …
- $g^*(n)+h(n) \leq f^*(n)$, which is true if h(n)…
- never overestimates, $h(n) \leq h^*(n)$.
- Such a heuristic is called "admissible".

Iterative Deepening Heuristic Search (IDA*)

- Using this pruning idea, when you generate a node n you only explore beneath it (with a recursive call) if $f(n) \leq B$, the current cost bound (not depth bound).

- How do you update the cost bound from one iteration to the next ?

(2) Node Ordering

Use the heuristic to decide which node to expand next. Two main options.

1. Choose the node with minimum h(n). Uniform-cost search using this selection rule is called “Pure Heuristic Search”
2. Choose the node with minimum f(n). Uniform-cost search using this selection rule is called A*.

Terminology

- Expanding a state means generating its successors (children).
- A state is open if it has been generated but not expanded.
- A state is closed if it has been expanded.
- Open list = data structure holding all currently open states.
- Closed list = data structure indicating which states are closed.

Dijkstra's Algorithm*

- Put (start,0) in OPEN
- Repeat
  - If OPEN is empty, exit with failure
  - Select (n,g(n)) on OPEN with minimum g(n)
  - If n is the goal, return path from start to goal
  - Move (n,g(n)) from OPEN to CLOSED
  - For each successor x of n:
    - If (x,g(x)) is on OPEN and g(x) > g(n)+cost(n,x) then replace (x,g(x)) on OPEN with (x,g(n)+cost(n,x))
    - If (x,_) is on neither OPEN nor CLOSED then add (x,g(n) +cost(n,x)) to OPEN

* Uniform-cost Search

A* Algorithm

- Put (start,**f(start)**) in OPEN
- Repeat
  - If OPEN is empty, exit with failure
  - Select (n,**f**(n)) on OPEN with minimum **f**(n)
  - If n is the goal, return path from start to goal
  - Move (n,**f**(n)) from OPEN to CLOSED
  - For each successor x of n:
    - If (x,**f**(x)) is on OPEN and **f**(x) > g(n)+cost(n,x)**+h(x)** then replace (x,**f**(x)) on OPEN with (x,g(n)+cost(n,x)**+h(x)**)
    - If (x,_) is on neither OPEN nor CLOSED then add (x,g(n)+cost(n,x)**+h(x)**) to OPEN
    - If (x,f(x)) is on CLOSED and f(x) > g(n)+cost(n,x)+h(x) then remove (x,f(x)) from CLOSED and add (x,g(n)+cost(n,x)+h(x)) to OPEN

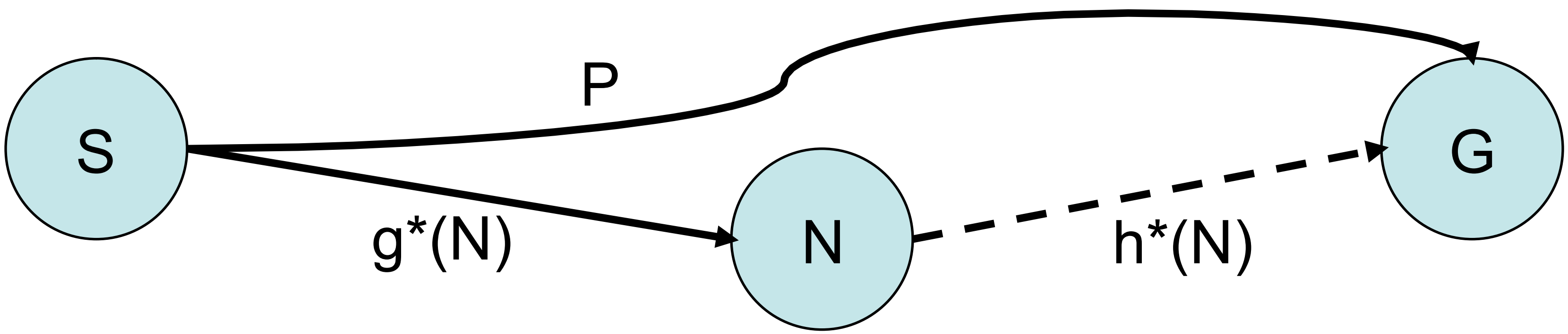
Question

When these algorithms stop (first remove the goal from Open), under what conditions are we guaranteed that this path to the goal is optimal ?

Dijkstra: all edge weights are non-negative

A*: impose conditions on the heuristic

Condition on the Heuristic



Require <N, g*(N)+h(N) > lower cost than <G,P>

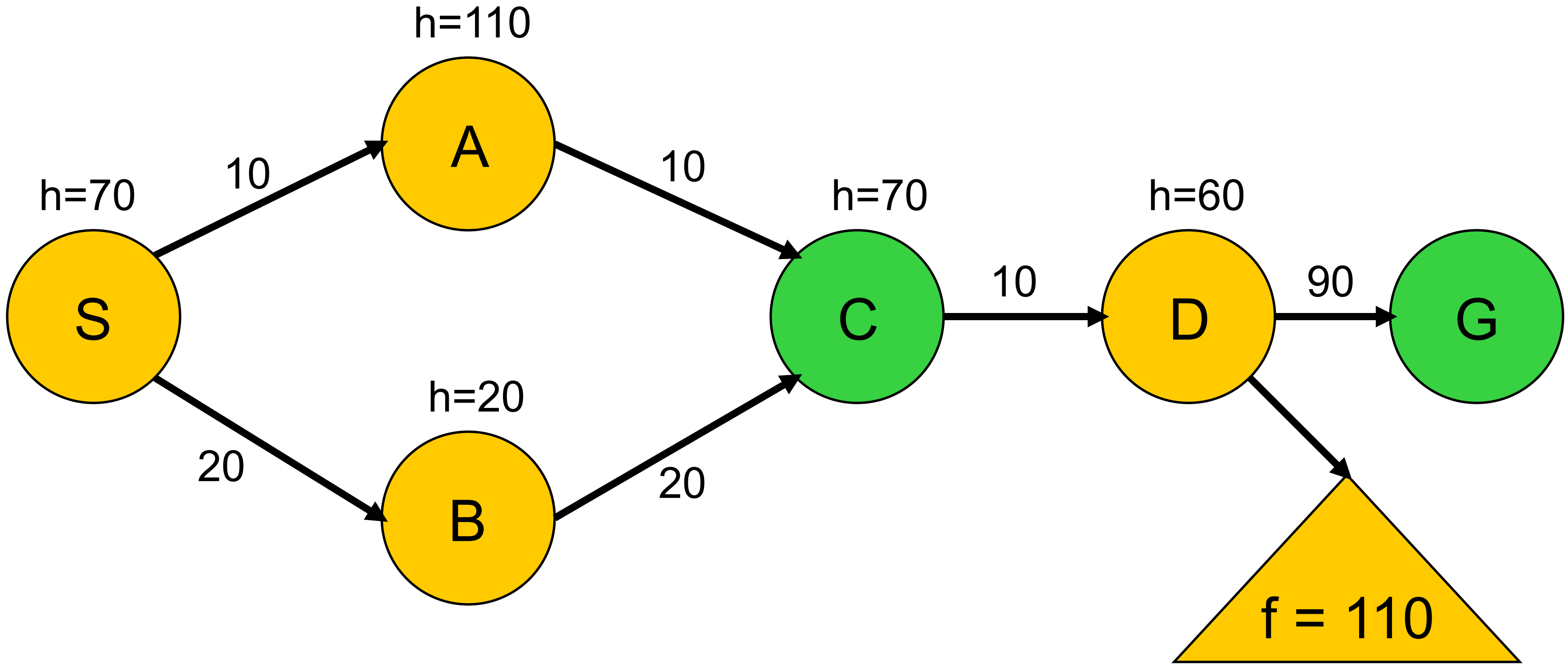
$g^*(N)+h(N) < P$

$\Leftrightarrow h(N) < P - g^*(N)$

$\Leftarrow h(N) \leq h^*(N)$ (because $h^*(N) < P - g^*(N)$)

Admissible Heuristic $\Rightarrow$ A* stops with an optimal path to goal

A* must re-open closed nodes



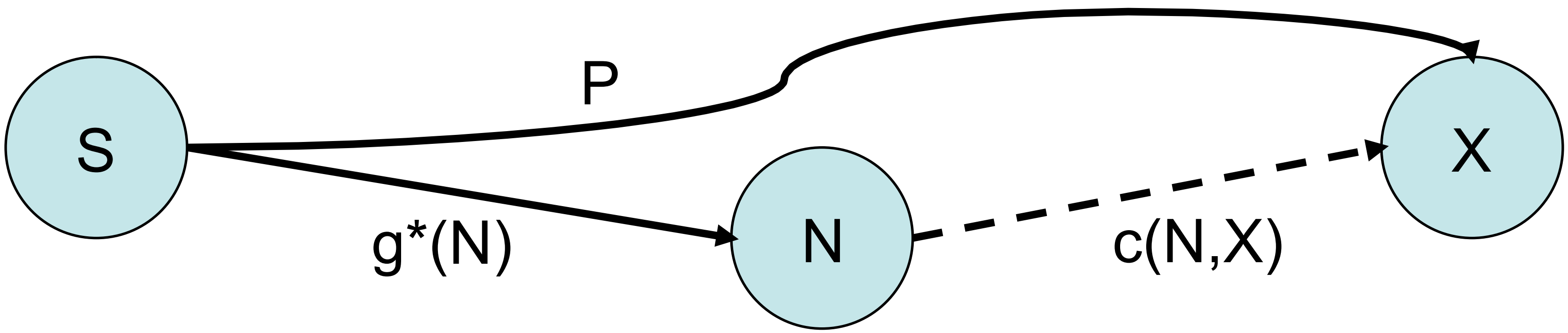
OPEN: (G,140), **(C,90)**

CLOSED: (S,70), (B,40), (C,110) , (D,110), …(A, 120)

How Bad Can This Be ?

- If N is the number of distinct nodes that A* expands, the total number of node expansions could be as bad as $O(2^N)$.
- Some A* variations reduce the worst-case to $O(N^2)$.
- Alternatively, if the heuristic has a special property, basic A* never re-opens a node once it is closed, so is $O(N)$.

Consistent Heuristic



Require <N, g*(N)+h(N) > lower cost than <X,P+h(X)>

$$g^*(N)+h(N) < P+h(X)$$

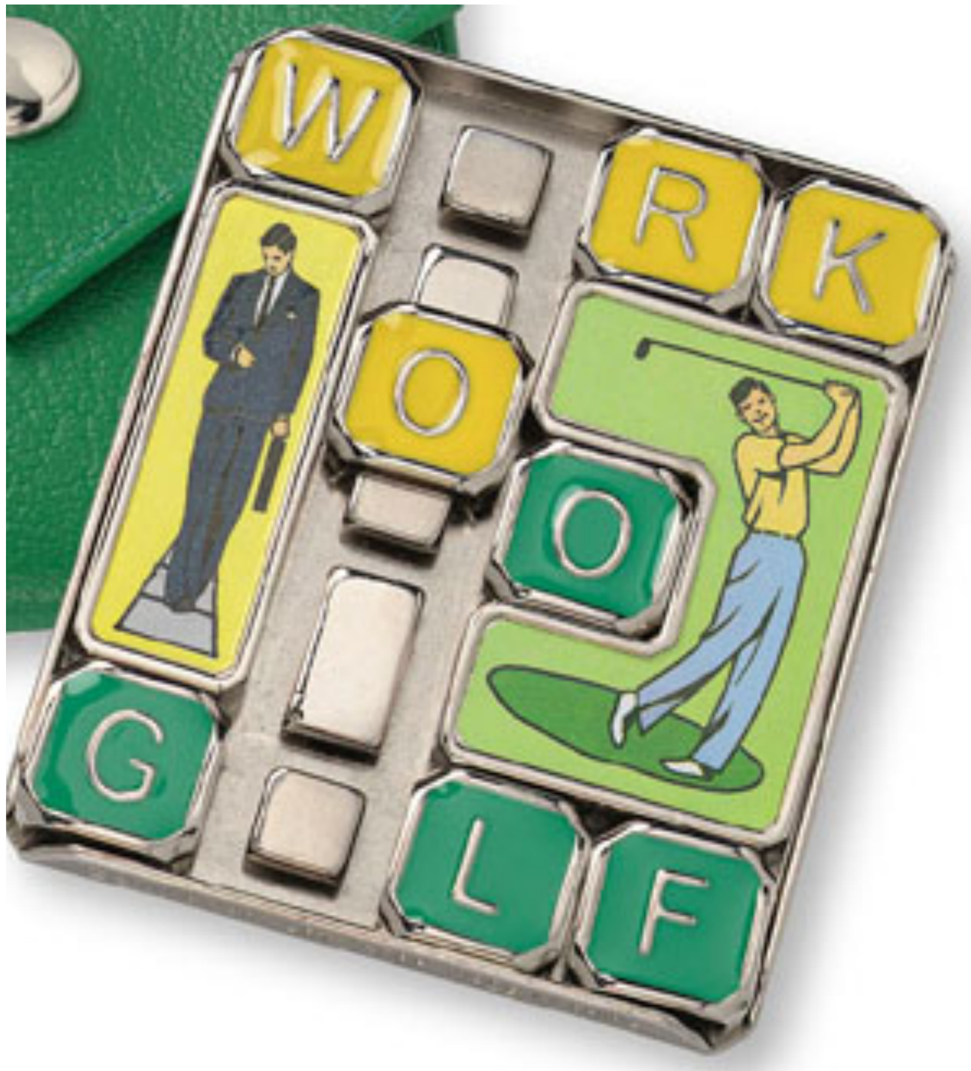
$$\Leftrightarrow h(N) - h(X) < P - g^*(N)$$

$$\Leftarrow h(N) - h(X) \leq c(N,X) \qquad \text{(because } c(N,X) < P - g^*(N)\text{)}$$

A heuristic is <u>consistent</u> if $h(N) \leq c(N,X) + h(X)$ for all N and **all** X.

Consistent $\Rightarrow$ first path to X off Open is optimal for all X

The Need for Automatically Generated Heuristics



Summary

- There are lots of different kinds of search
  - With different optimizations and guarantees
- All involve planning – using your knowledge of the worlds dynamics to anticipate the consequences of your action, and then picking the best
- All search involves computing how good it is to be in each state
  - And benefits from a good initial estimate