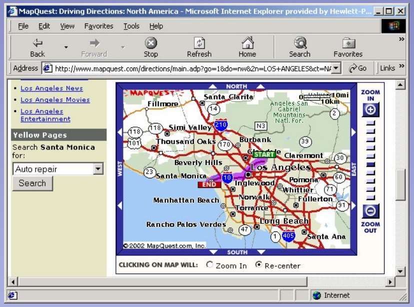
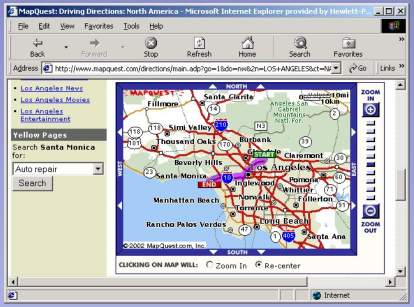

MapQuest: Driving Directions: North America - Microsoft Internet Explorer provided by Hewlett-P...
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://www.mapquest.com/directions/main.adp?go=1&do=nw&2n=LOS+ANGELES&ct=NA
Los Angeles News
Los Angeles Movies
Los Angeles Entertainment
Yellow Pages
Search Santa Monica for:
Auto repair
Search
START
END
CLICKING ON MAP WILL: Zoom In Re-center
Internet

Introduction to Artificial Intelligence
A* Search

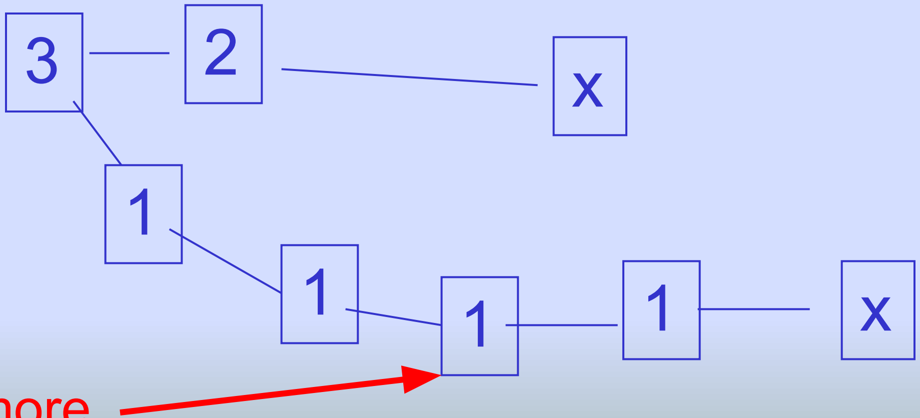
Ruth Bergman

Fall 2004

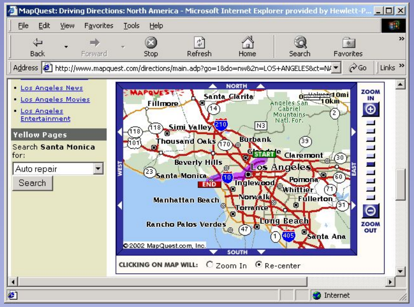


Best-First Search Review

- Advantages
  - Takes advantage of domain information to guide search
  - Greedy advance to the goal
- Disadvantages
  - Considers cost to the goal from the current state
  - Some path can continue to look good according to the heuristic function



At this point the path is more costly than the alternate path

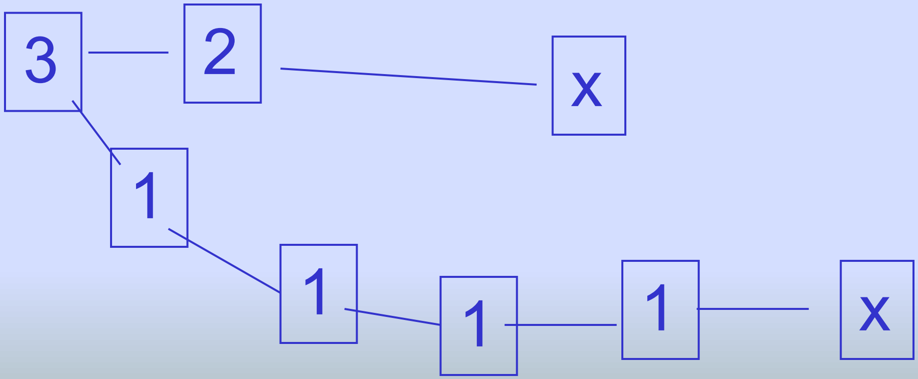


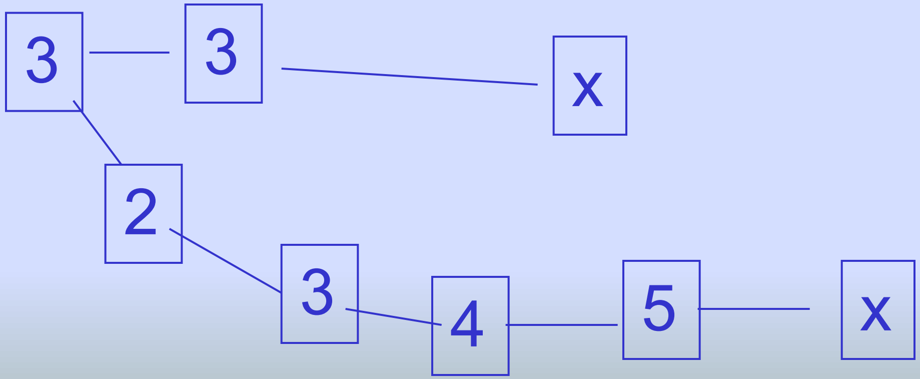
The A* Algorithm

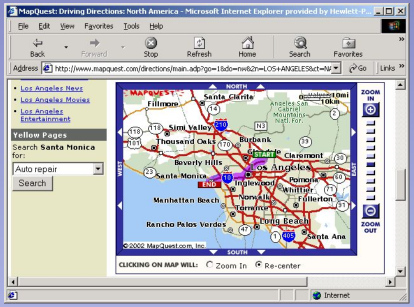
- Consider the overall cost of the solution.

  f(n) = g(n) + h(n)   where g(n) is the path cost to node n

  think of f(n) as an estimate of the cost of the best solution *going through the node n*

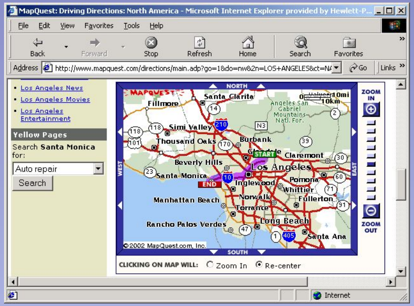






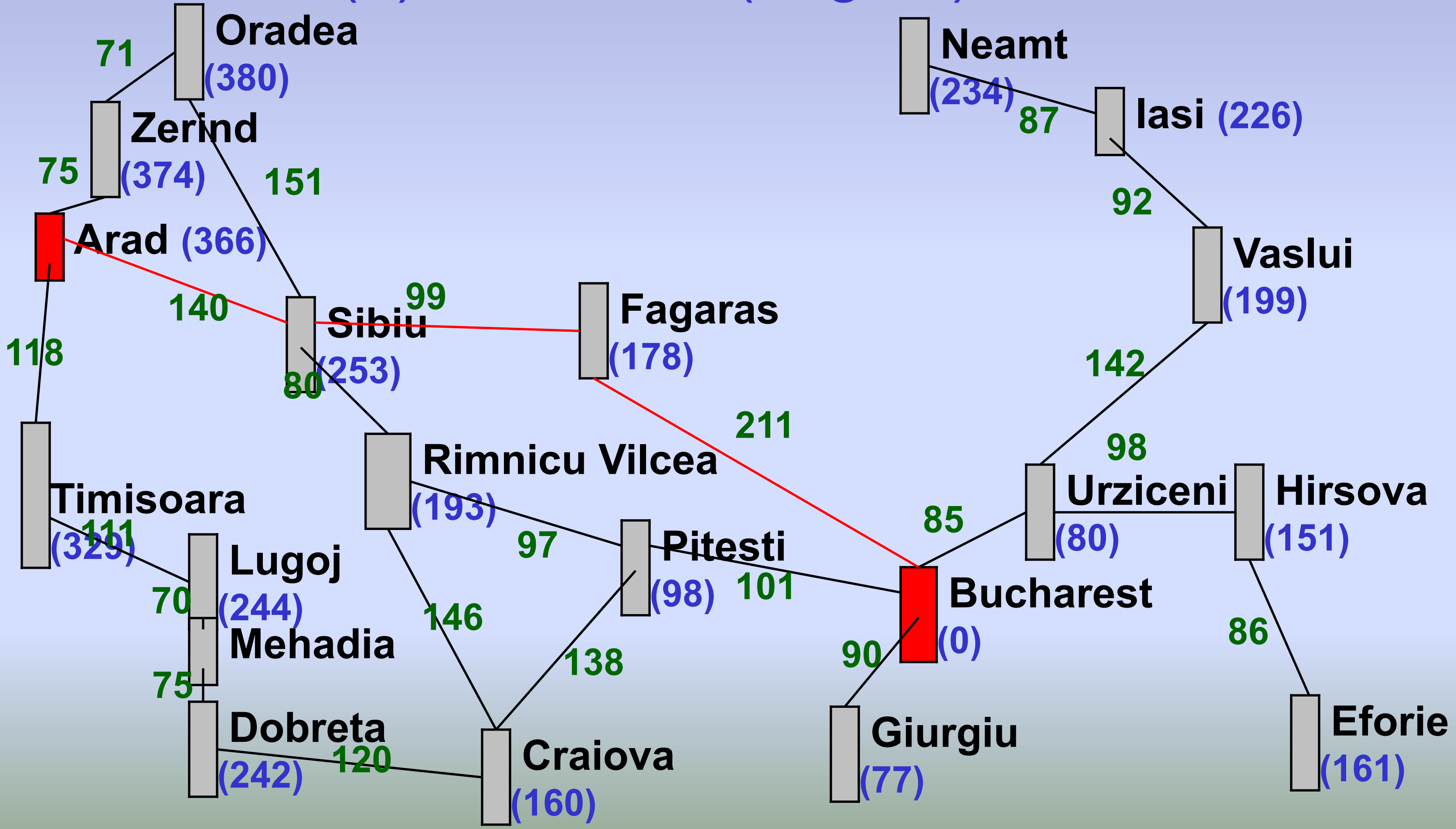
The A* Algorithm

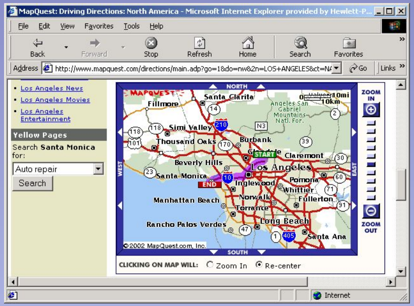
```
A*-Search(initial-test)                    ;; functions cost, h, succ, and GoalTest are defined
   open <- MakePriorityQueue(initial-state, NIL, 0, h(initial-state), h(initial-state))
                        ;; (state, parent, g, h, f)
    while (not(empty(open)))
         node □ pop(open), state □ node-state(node)
         closed □ push (closed, node)
         if GoalTest(state) succeeds return node
         for each child in succ(state)
              new-cost □ node-g(node) + cost(state,child)
              if child in open
                if new-cost < g value of child
                  update(open, child, node, new-cost, h(child), new-cost+h(child))
             elseif child in closed
                if new-cost < g value of child
                  insert(open, child, node, new-cost, h(child), new-cost+h(child))
                  delete(closed,child)
              else
                 open □ push(child, node, new-cost, h(child), new-cost+h(child))
    return failure
```



A* Search: Example

- Travel: h(n) = distance(n, goal)





A* Search : Example

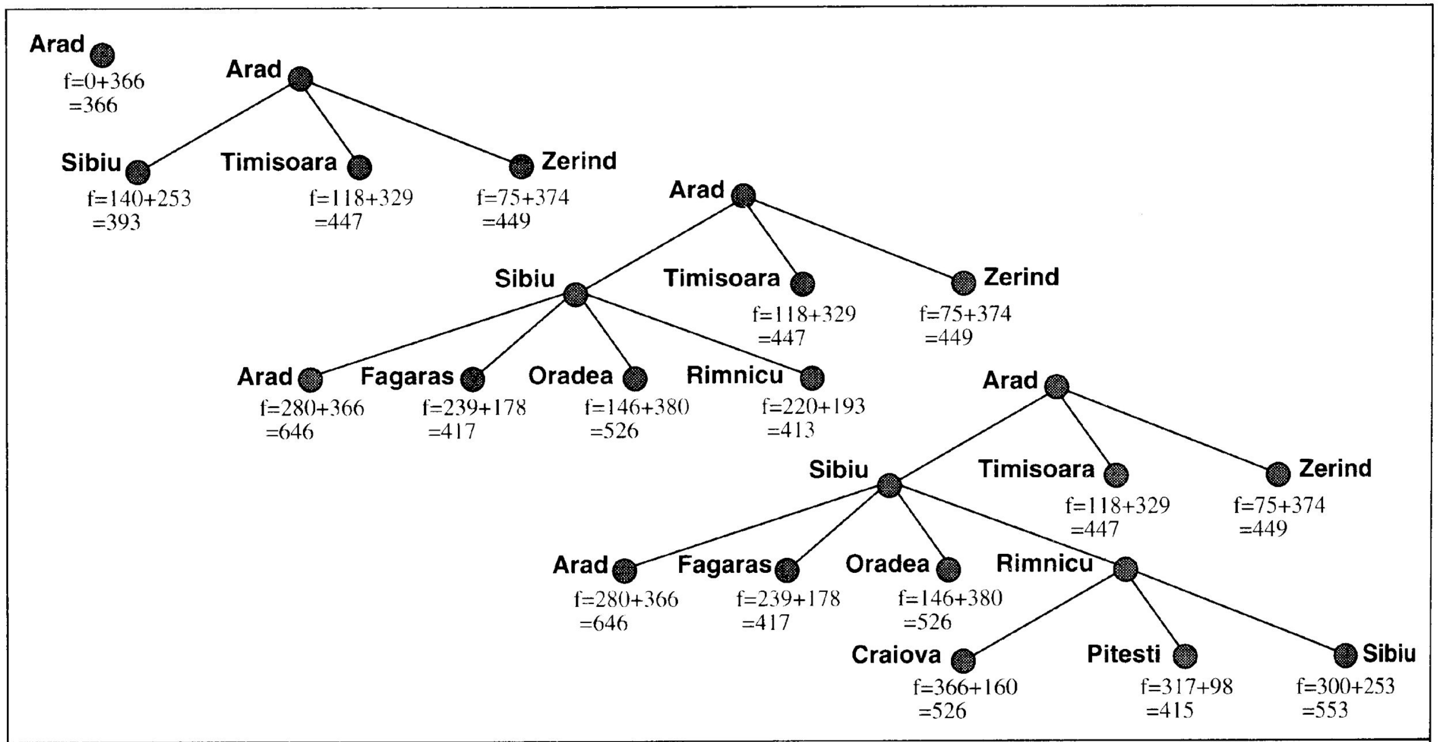
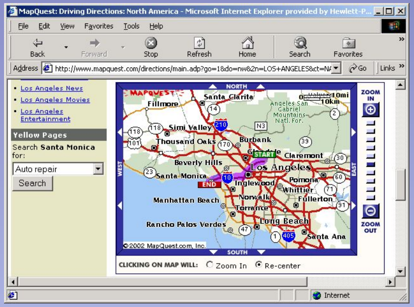
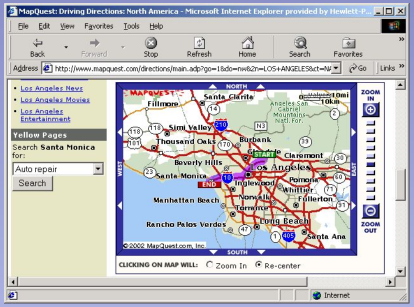


Figure 4.4 Stages in an A* search for Bucharest. Nodes are labelled with $f = g + h$. The h values are the straight-line distances to Bucharest taken from Figure 4.1.



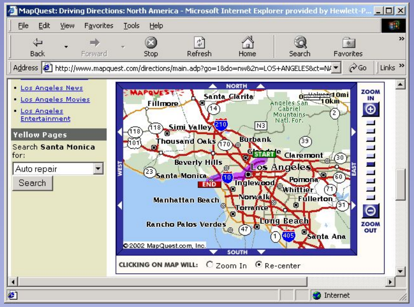
Admissible Heuristics

- we also require h be *admissible:*
  - a heuristic h is admissible if h(n) < h*(n) for all nodes n,
  - where h* is the actual cost of the optimal path from n to the goal
- Examples:
  - travel distance straight line distance must be shorter than actual travel path
  - tiles out of place each move can reorder at most one tile distance of each out of place tile from the correct place each move moves a tile at most one place toward correct place


MapQuest: Driving Directions: North America - Microsoft Internet Explorer provided by Hewlett-P...
Los Angeles News
Los Angeles Movies
Los Angeles Entertainment
Yellow Pages
Search Santa Monica for:
Auto repair
Search
START
END
CLICKING ON MAP WILL: Zoom In Re-center

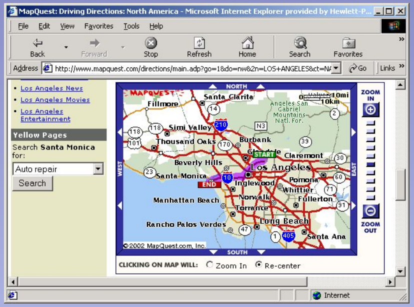
Optimality of A*

- Let us assume that f is non-decreasing along each path
  - if not, simply use parent's value
  - if that's the case, we can think of A* as expanding f contours toward the goal; better heuristics make this contour more "eccentric"
- Let G be an optimal goal state with path cost f*
- Let G_2 be a suboptimal goal state with path cost $g(G_2) > f^*$.
  - suppose A* picks G_2 before G (A* is *not* optimal)
  - suppose n is a leaf node on the path to G when G_2 is chosen
  - if h is admissible, then f* >= f(n)
  - since n was not chosen, it must be the case that $f(n) >= f(G_2)$
  - therefore $f^* >= f(G_2)$, but since G_2 is a goal, $h(G_2)=0$, so $f^* >= g(G_2)$
  - But this is a contradiction --- G_2 is a better goal node than G
  - Thus, our supposition is false and A* is optimal.



Completeness of A*

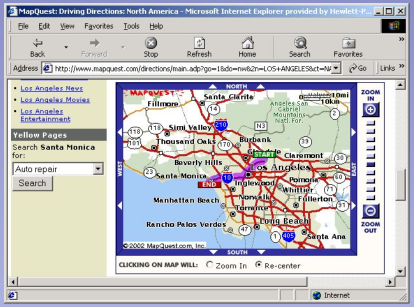
- Suppose there is a goal state G with path cost f*
  - Intuitively: since A* expands nodes in order of increasing f, it must eventually expand node G
- If A* stops and fails
  - Prove by contradiction that this is impossible.
  - There exists a path from the initial state to the node state
  - Let n be the last node expanded along the solution path
  - n has at least one child, that child should be in the open nodes
  - A* does not stop until there are open list is empty (unless it finds a goal state). Contradiction.
- A* is on an infinite path
  - Recall that cost(s1,s2) > δ
  - Let n be the last node expanded along the solution path
  - After f(n)/δ the cumulative cost of the path becomes large enough that A* will expand n. Contradiction.

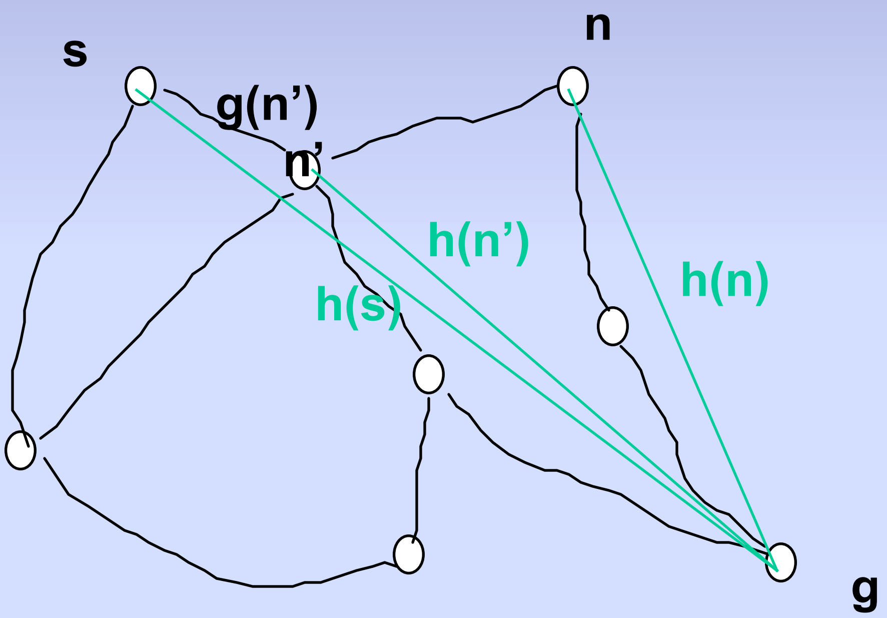


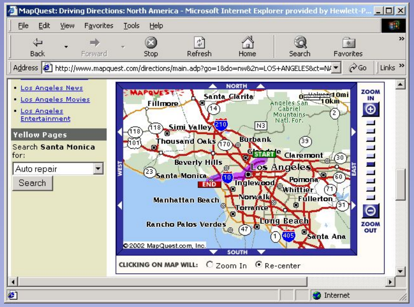
UCS, BFS, Best-First, and A*

- $f = g + h$ => A* Search
- $h = 0$ => Uniform cost search
- $g = 1$, $h = 0$ => Breadth-First search
- $g = 0$ => Best-First search

Road Map Problem


MapQuest: Driving Directions: North America - Microsoft Internet Explorer provided by Hewlett-P...
Los Angeles News
Los Angeles Movies
Los Angeles Entertainment
Yellow Pages
Search Santa Monica for:
Auto repair
Search
START
END
CLICKING ON MAP WILL: Zoom In Re-center



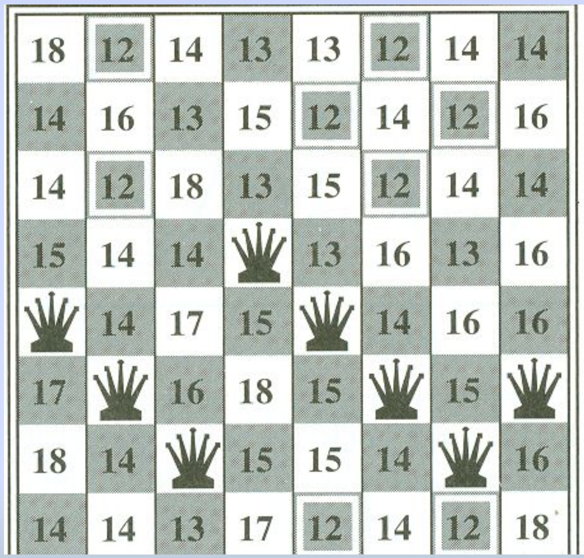

MapQuest: Driving Directions: North America - Microsoft Internet Explorer provided by Hewlett-P...
http://www.mapquest.com/directions/main.adp?go=1&do=nw&2n=LOS+ANGELES&ct=NA
Yellow Pages
Search Santa Monica for:
Auto repair
Search

8-queens

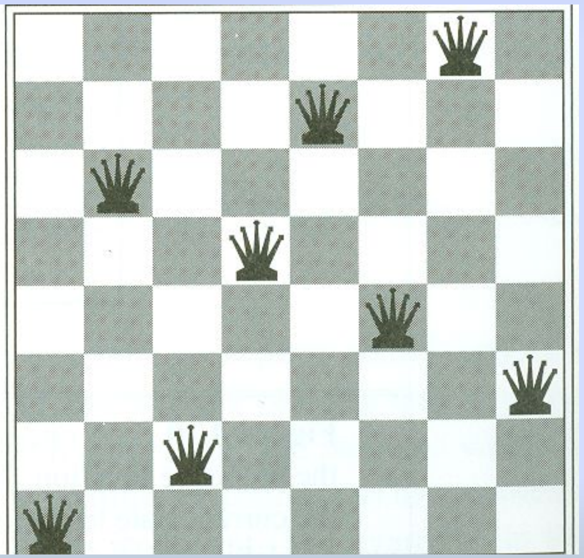
State contains 8 queens on the board

Successor function returns all states generated by moving a single queen to another square in the same column (8*7 = 56 next states)

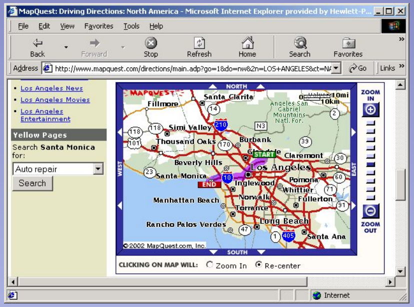
h(s) = number of queens that attack each other in state s.



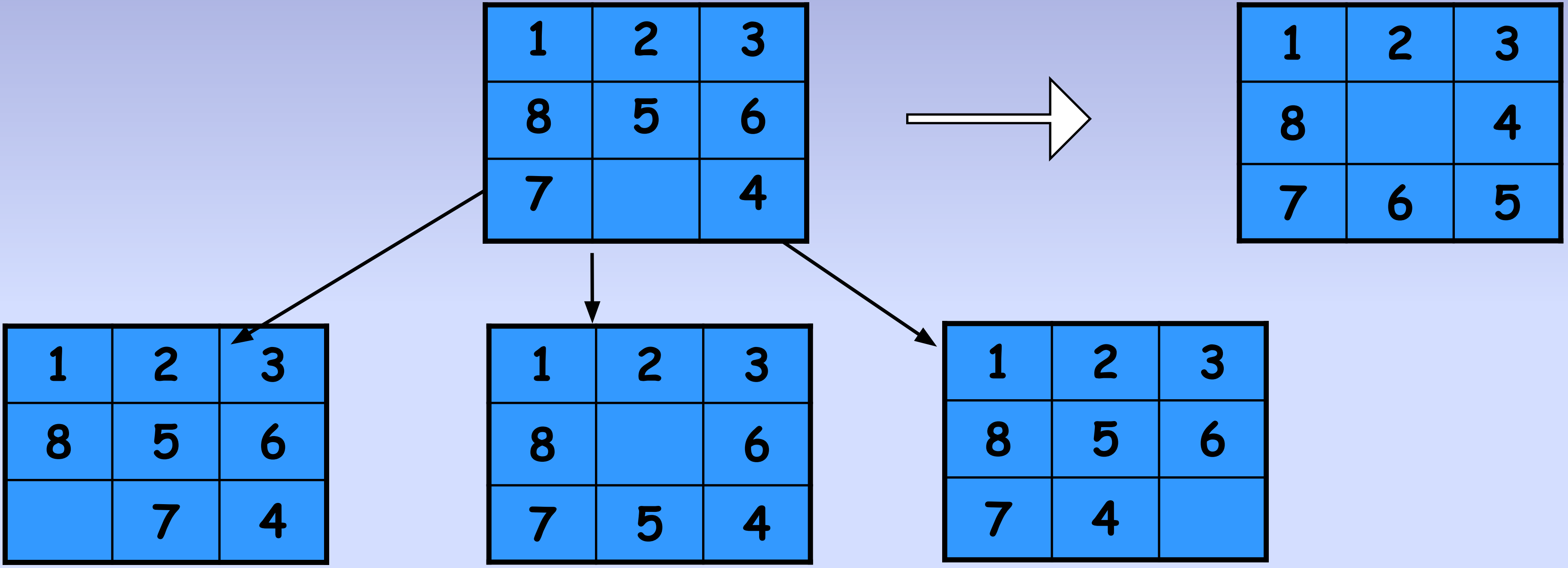
H(s) = 17

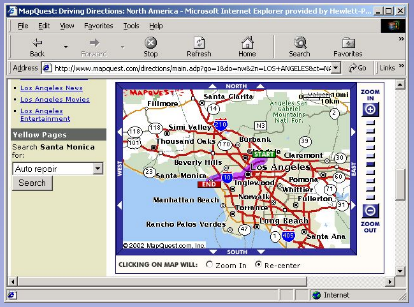


H(s) = 1


MapQuest: Driving Directions: North America - Microsoft Internet Explorer provided by Hewlett-P...
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://www.mapquest.com/directions/main.adp?go=1&do=nw&2n=LOS+ANGELES&ct=NA
Los Angeles News
Los Angeles Movies
Los Angeles Entertainment
Yellow Pages
Search Santa Monica for:
Auto repair
Search
START
END
CLICKING ON MAP WILL: Zoom In Re-center
Internet

Heuristics : 8 Puzzle





8 Puzzle

- Reachable state : 9!/2 = 181,440

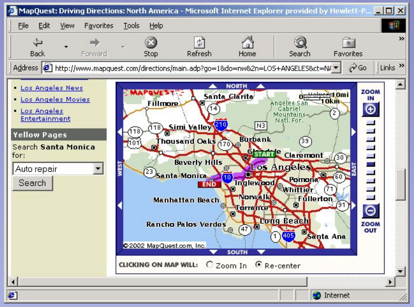
- Use of heuristics
  - h1 : # of tiles that are in the wrong position
  - h2 : sum of Manhattan distance

1	2	3
8	5	6
7		4

h1 = 3

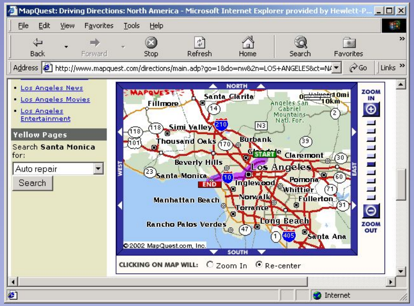
h2 = 1+2+2=5

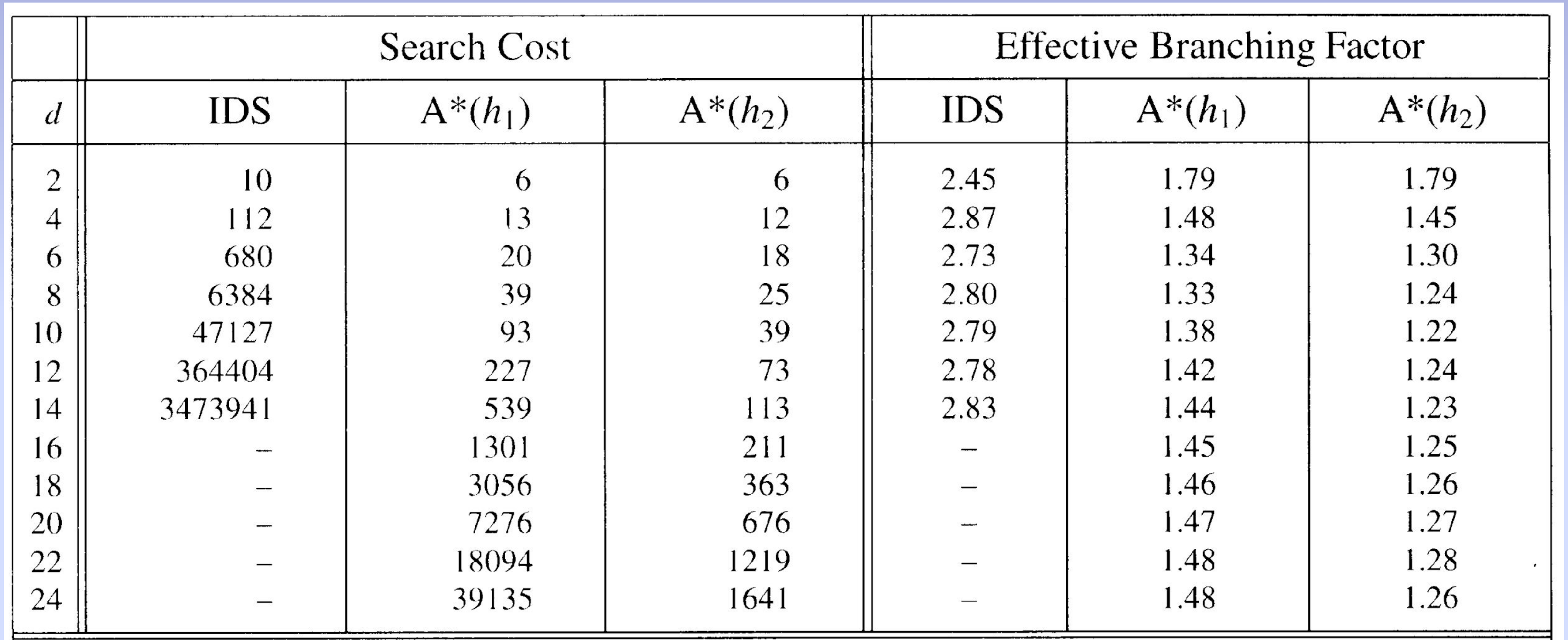
1	2	3
8		4
7	6	5



Effect of Heuristic Accuracy on Performance

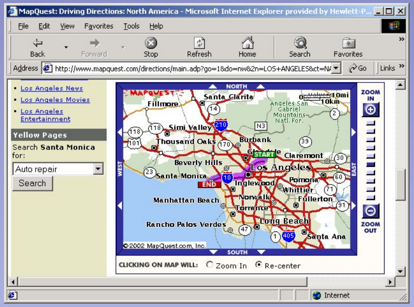
- Well-designed heuristic have its branch close to 1
- h_2 dominates h_1 iff
  $h_2(n) \geq h_1(n), \forall\ n$
- It is always better to use a heuristic function with higher values, as long as it does not overestimate
- Inventing heuristic functions
  - Cost of an exact solution to a relaxed problem is a good heuristic for the original problem
  - collection of admissible heuristics
    $h^*(n) = \max(h_1(n), h_2(n), \ldots, h_k(n))$





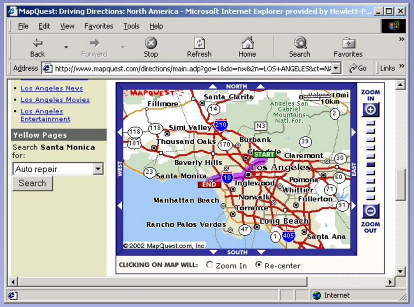
	Search Cost			Effective Branching Factor		
d	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Figure 4.8 Comparison of the search costs and effective branching factors for the ITERATIVE-DEEPENING-SEARCH and A* algorithms with h_1, h_2. Data are averaged over 100 instances of the 8-puzzle, for various solution lengths.



A* summary

- Completeness
  - provided finite branching factor and finite cost per operator
- Optimality
  - provided we use an admissible heuristic
- Time complexity
  - worst case is still $O(b^d)$ in some special cases we can do better for a given heuristic
- Space complexity
  - worst case is still $O(b^d)$

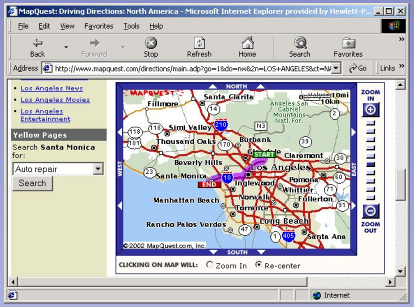

MapQuest: Driving Directions: North America - Microsoft Internet Explorer
Los Angeles News
Los Angeles Movies
Los Angeles Entertainment
Yellow Pages
Search Santa Monica for:
Auto repair
Search
START
END
Los Angeles
Santa Monica
ZOOM IN
ZOOM OUT
CLICKING ON MAP WILL: Zoom In Re-center

Relax Optimality

- Goals:
  - Minimizing search cost
  - Satisficing solution, i.e. bounded error in the solution

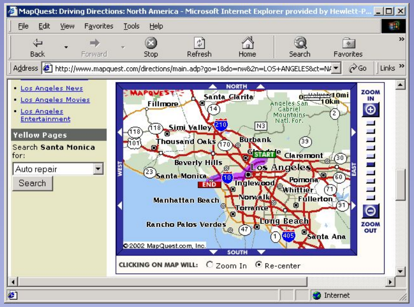
$f(s) = (1-w)\, g(s) + w\, h(s)$

- g can be thought of as the breadth first component
- w = 1 => Best-First search
- w = .5 => A* search
- w = 0 => Uniform search



Iterative Deepening A*

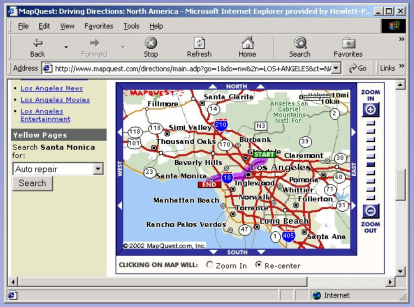
- Goals
  - A storage efficient algorithm that we can use in practice
  - Still complete and optimal
- Modification of A*
  - use f-cost limit as depth bound
  - increase threshold as minimum of f(.) of previous cycle
- Each iteration expands all nodes inside the contour for current f-cost
- same order of node expansion


MapQuest: Driving Directions: North America - Microsoft Internet Explorer provided by Hewlett-P...
Los Angeles News
Los Angeles Movies
Los Angeles Entertainment
Yellow Pages
Search Santa Monica for:
Auto repair
Search
START
END
Los Angeles
Santa Monica
Long Beach
CLICKING ON MAP WILL: Zoom In Re-center

IDA* Algorithm

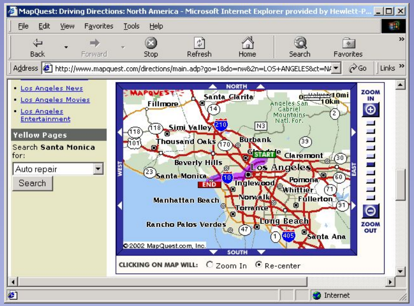
```
IDA* (state,h) returns solution
    f-limit <- h(state)
    loop do
      solution, f-limit □ DFS-Contour(state, f-limit)
      if solution is non-null return solution
      if f-limit = ∞ return failure
    end

DFS-Contour (node,f-limit) returns solution
    if f (node) > f-limit return null, f(node)
    if GoalTest(node) return node, f-limit
    next-f □ ∞
    for each node s in succ(node) do
      solution, new-f □ DFS-Contour(s, f-limit)
      if solution is non-null return solution, f-limit
      next-f □ Min(next-f, new-f)
    end
    return null, next-f
```



IDA* Properties

- Complete:
  - if shortest path fits into memory
- Optimal:
  - if shortest optimal path fits into memory
- Time Complexity: $O(b^{2d})$
- Space Complexity: $O(bd)$


MapQuest: Driving Directions: North America - Microsoft Internet Explorer provided by Hewlett-P...
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://www.mapquest.com/directions/main.adp?go=1&do=nw&2n=LOS+ANGELES&ct=NA
Go
Links
Los Angeles News
Los Angeles Movies
Los Angeles Entertainment
Yellow Pages
Search Santa Monica for:
Auto repair
Search
NORTH
SOUTH
WEST
EAST
Fillmore
Santa Clarita
Angeles San Gabriel Mountains Natl. For.
Simi Valley
Thousand Oaks
Burbank
START
Claremont
Beverly Hills
Los Angeles
Santa Monica
END
Inglewood
Pomona
Whittier
Manhattan Beach
Norwalk
Fullerton
Torrance
Rancho Palos Verdes
Long Beach
Santa Ana
©2002 MapQuest.com, Inc.
ZOOM IN
ZOOM OUT
CLICKING ON MAP WILL: Zoom In Re-center
Internet

Mapquest

- http://www.mapquest.com/
- MapQuest uses a "double Dijkstra" algorithm for its driving directions, working backward from both the starting and ending points at once. MapQuest uses a "double Dijkstra" algorithm for its driving directions, working backward from both the starting and ending points at once.
- the algorithm uses heuristic tricks to minimize the size of the graph that must be searched.