

## Searching paths whose link lengths vary

Task: Find shortest route to the goal.

A good method: Follow route we think is shortest

This is the strategy of A\*:

- At any N along the route, guess total path length by adding:
  - Length of shortest route found so far from start point to N.
  - A heuristic distance function **h** for distance from N to goal.
- Require that  $h(N)$  is never greater than the real distance, so that we won't ignore good paths because of pessimistic estimates.
- Continue searching even after finding the goal, as long as our estimates suggest that other paths are better than the ones found so far.

1

### Sketch of A\*

Assume we are given a function **h(n)** to estimate cost to the goal.

We will maintain:

- An array **u(n)** giving the cost to reach n from the start state.
- An array **p(n)** recording a pointer to the best (cheapest) node from which it was reached.
- A set OPEN of pending nodes
- A set CLOSED of nodes already reached

3

## Examples of h for driving distance

- Straight-line distance on a map.
- The distance on a distance table (your distance may be worse if you don't know the roads very well).
- Estimate mileage by how long your dare-devil roommate says each route should take.

Other criteria for cost: danger, views, time (which may not match distance)

2

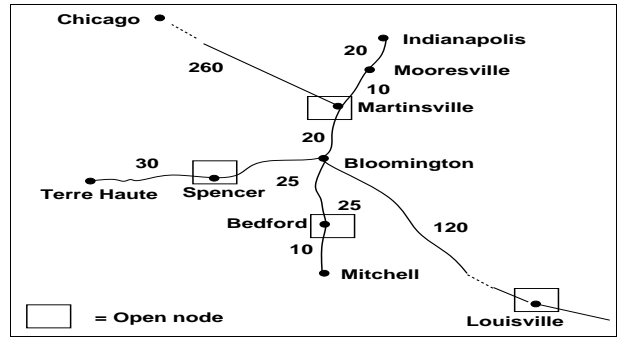
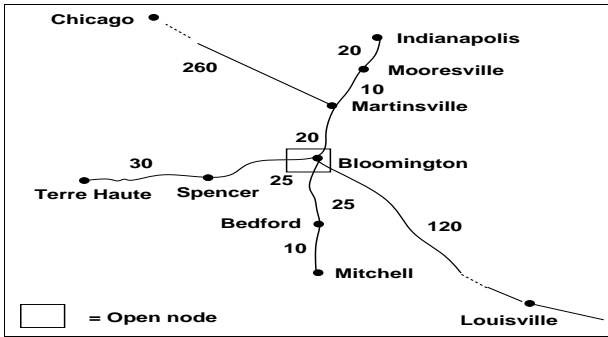
## Search process

1. Set current element N to initial state
2. Is N goal? If so
  - If for each open node, the estimated cost of proceeding through that node is greater than the cost of the shortest path you've found to goal, stop.
3. Generate successors  $S_i$  of N. For each  $S_i$ ,
  - If  $S_i$  is not already in OPEN or CLOSED,  
 $u(S_i) \leftarrow u(N) + \text{distance}(N, S_i)$ .  
Set  $p(S_i)$  to point to N.
  - If  $S_i$  is already on the OPEN or CLOSED list,  
 $u(S_i) \leftarrow \min\{u(N) + \text{distance}(N, S_i), u(S_i)\}$ .  
If new path to  $S_i$  is cheaper, update  $p(S_i)$ .Replace N with its successors on OPEN. Put N in CLOSED.
4. As the next N, take the open node S with the lowest  $u(S) + h(S)$ . Go to 2

4

A\* to find the shortest route Bloomington-Indy  
Initial state:

Step 2:



Open nodes:

Open nodes:

City	Tot. dist. B→city	Est. dist city →Indy	Total
Bloomington	0	50	50

City	Tot. dist. B→city	Est. dist city →Indy	Total
Louisville	120	160	280
Bedford	25	45	70
Spencer	25	60	85
Martinsville	20	27	47

Pick the node with the best total: Bloomington.

Pick node with the best total: Martinsville.

Replace it with successors not yet visited: Louisville, Bedford, Spencer, Martinsville.

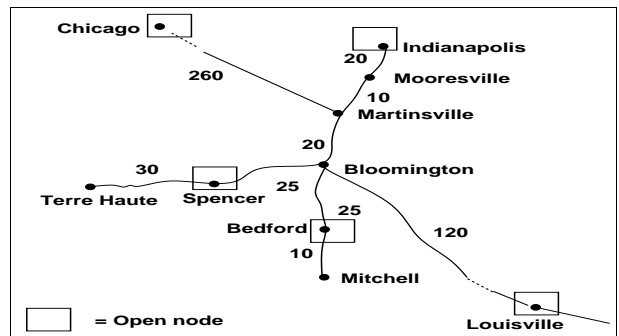
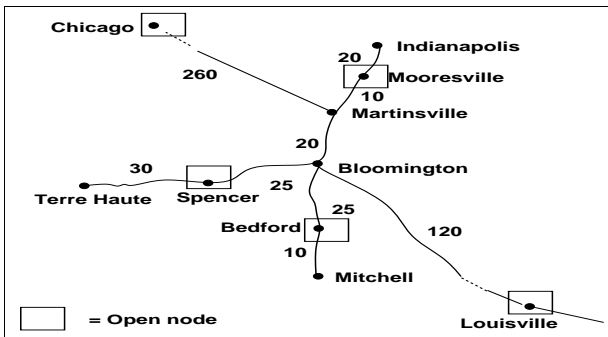
Replace it with successors not yet visited: Chicago, Mooresville

5

6

Step 3:

Step 4:



Open nodes:

Open nodes:

City	Tot. dist. B→city	Est. dist city →Indy	Total
Chicago	280	200	480
Mooresville	30	18	48
Louisville	120	160	280
Bedford	25	45	70
Spencer	25	60	85

City	Tot. dist. B→city	Est. dist city →Indy	Total
Indianapolis	50	0	50
Chicago	280	200	480
Louisville	120	160	280
Bedford	25	45	70
Spencer	25	60	85

Pick node with the best total: Mooresville.

Pick node with the best total: Indianapolis.

Replace it with successors not yet visited: Indianapolis.

No estimates are cheaper: **done!**

7

8