## CS 106B

Lecture 25: Dijkstra's Algorithm and the A* Algorithm
Friday, May 25, 2018

Programming Abstractions
Spring 2018
Stanford University
Computer Science Department
Lecturer: Chris Gregg

reading:
Programming Abstractions in C++, Chapter 18.6

## Today's Topics

-Logistics

- YEAH hours next Tuesday.
- More on Graphs:
-Dijkstra's Algorithm
- $A^{*}$ Algorithm


## Last time: DFS and BFS

Depth First Search: Keep searching along a path until we need to backtrack: not guaranteed shortest path.

Breadth First Search: Look at paths containing neighbor of distance 1, then neighbors of distance 2, etc., until a path is found: guaranteed shortest path.


## No Weights!

Depth First Search: Keep searching along a path until we need to backtrack: not guaranteed shortest path.

Breadth First Search: Look at paths containing neighbor of distance 1, then neighbors of distance 2, etc., until a path is found: guaranteed shortest path.

Neither DFS or BFS dealt with weights!


## Search Without Weights

Search without weights: What is the shortest path from A to D?


## Search With Weights

Search without weights: What is the shortest path from A to D?


Shortest Path: A-D

## Search With Weights

Search with weights: What is the shortest path from A to D?
(Assume the numbers are distances, and we want to minimize the overall path distance)


## Search With Weights

Search with weights: What is the shortest path from A to D?
(Assume the numbers are distances, and we want to minimize the overall path distance)


Shortest Path: A-B-C-D

## Search With Weights

Search with weights: What is the shortest path from A to D?
(Assume the numbers are distances, and we want to minimize the overall path distance)


Shortest Path: A-B-C-D

Our BFS would break! The "shortest" path with weights depends on the weight!

## BFS without weights...

If we use BFS to find the path (disregarding weights), we would use a queue to enqueue each path.


## Dijkstra's Algorithm

A different algorithm, called "Dijstra's Algorithm" (after the computer scientist Edsger Dijkstra) uses a priority queue to enqueue each path.


## Breadth First Search

bfs from $v_{1}$ to $v_{2}$ :
create a queue of paths (a vector), q
q.enqueue( $v_{1}$ path)
while $q$ is not empty and $v_{2}$ is not yet visited:
path = q.dequeue()
$\mathrm{v}=$ last element in path
mark $v$ as visited
if $v$ is the end vertex, we can stop.
for each unvisited neighbor of $v$ : make new path with v's neighbor as last element
 enqueue new path onto $q$

## Dijkstra's Algorithm

dijkstra's from $v_{1}$ to $v_{2}$ :
create a priority queue of paths (a vector), $q$ q.enqueue( $v_{1}$ path)
while $q$ is not empty and $v_{2}$ is not yet visited:
path = q.dequeue()
$v=$ last element in path
mark $v$ as visited
if $v$ is the end vertex, we can stop.
for each unvisited neighbor of $v$ :
make new path with v's neighbor as last element
 enqueue new path onto $q$

## Dijkstra's Algorithm

Dijkstra's algorithm is what we call a "greedy" algorithm.
This means that the algorithm always takes the path that is best at the given time -- e.g., starting from A, you would prioritize the path from A-B (10) over the path from A-D (12345). This is why we use a priority queue, because the prioritization is handled with a priority queue.


## Djjkstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

Let's look at Dijkstra from a to i:

| priority |
| :--- |
| queue: |

Vector<Vertex *> startPath startPath.add(A,0)
pq.enqueue(startPath)


Visited Set: (empty)

## Dijkstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

Let's look at Dijkstra from a to i:
priority

in while loop:
curPath $=$ pq.dequeue() (path is $A$, priority is 0 )
$v=$ last element in curPath ( v is A )
mark v as visited
enqueue all unvisited neighbor paths onto q, with updated priorities based on new edge length

## Dikstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

Let's look at Dijkstra from a to i:
priority
queue:

|  |  |  |  |  | front |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Path: |  |  |  | ADG | AB | ADE |
| Total Cost: |  |  |  | 12 | 6 | 4 |


in while loop:
curPath = pq. dequeue() (path is AD, priority is 3 )
$v=$ last element in curPath ( $v$ is $D$ )
mark v as visited
enqueue all unvisited neighbor paths onto q, with updated priorities based on new edge length

Visited Set: A, D

## Dikstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

in while loop:
curPath = pq.dequeue) (path is ADE, priority is 4)
$v=$ last element in curPath ( v is E)
mark v as visited
enqueue all unvisited neighbor paths onto q, with updated priorities based on new edge length

Visited Set: A, D, E

## Dikstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

in while loop:
curPath = pq.dequeue() (path is ADEB, priority is 5)
$v=$ last element in curPath ( v is B )
mark v as visited
enqueue all unvisited neighbor paths onto q, with updated priorities based on new edge length

Visited Set: A, D, E, B

## Dikstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

in while loop:
curPath $=$ pq.dequeue() (path is $A B$, priority is 6 )
$v=$ last element in curPath ( v is B )
mark v as visited
enqueue all unvisited neighbor paths onto q, with updated priorities based on new edge length

Visited Set: A, D, E, B

## Dijkstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

Let's look at Dijkstra from a to i:
priority
queue:
$\left.\begin{array}{|c|c|c|c|c|c|c|}\hline & & & & & & \\ \text { Path: } & & \text { ADEHI } & \text { ADG } & \text { ADEF } & \text { ADEHG } & \text { ABC } \\ \text { ADEBC } \\ \hline \text { Total Cost: } & & 13 & 12 & 11 & 10 & 9\end{array}\right) 8$

in while loop:
curPath = pq.dequeue() (path is ADEH, priority is 8)
$v=$ last element in curPath ( v is H )
mark v as visited
enqueue all unvisited neighbor paths onto q, with updated priorities based on new edge length

Note: cannot stop yet! ADEHI might not be shortest!

## Dijkstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

Let's look at Dijkstra from a to i:
priority
queue:

|  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Path: |  | ADEHI | ADG | ADEF | ADEHG |  |
| Total Cost: |  | 13 | 12 | 11 | 10 | 9 |


in while loop:
curPath = pq.dequeue) (path is ADEBC, priority is 8 )
$v=$ last element in curPath ( v is C )
mark v as visited
enqueue all unvisited neighbor paths onto q, with updated priorities based on new edge length

Visited Set: A, D, E, B, H, C

## Dijkstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

Let's look at Dijkstra from a to i:
priority
queue:

|  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| front |  |  |  |  |  |  |
| Path: |  | ADEHI | ADG | ADEF | ABCF | ADEHG ADEBCF |
| Total Cost: | 13 | 12 | 11 | 10 | 10 | 9 |

in while loop:
curPath = pq.dequeue() (path is ABC, priority is 9)
$v=$ last element in curPath ( v is C )
mark v as visited
enqueue all unvisited neighbor paths onto q, with updated priorities based on new edge length


Visited Set: A, D, E, B, H, C



## Dijkstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

Let's look at Dijkstra from a to i:
priority
queue:

|  |  |  |  |  | front |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Path: |  | ADEBCFI | ADEHI | ADG | ADEF | ABCF |
| ADEHG |  |  |  |  |  |  |
| Total Cost: |  | 16 | 13 | 12 | 11 | 10 | 10

in while loop:
curPath = pq.dequeue) (path is ADEBCF, priority is 9)
$\mathrm{v}=$ last element in curPath ( v is F )
mark v as visited
enqueue all unvisited neighbor paths onto q, with updated priorities based on new edge length


Visited Set: A, D, E, B, H, C

## Dijkstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

Let's look at Dijkstra from a to i:
priority
queue:

|  |  |  |  |  |  | front |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Path: |  | ADEBCFI | ADEHI | ADG | ADEF | ABCF |
| Total Cost: |  | 16 | 13 | 12 | 11 | 10 |

in while loop:

curPath = pq. dequeue() (path is ADEHG, priority is 10)
$v=$ last element in curPath ( v is G )
mark v as visited
enqueue all unvisited neighbor paths onto $q$, with updated priorities based on new edge length

## Dijkstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

in while loop:
curPath $=$ pq. dequeue() (path is ABCF, priority is 10)
$v=$ last element in curPath ( v is F )
mark v as visited
enqueue all unvisited neighbor paths onto q, with updated priorities based on new edge length

Visited Set: A, D, E, B, H, C, F, G



## Dijkstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

Let's look at Dijkstra from a to i:
priority
queue:

|  |  |  |  |  |  | front |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Path: |  | ADEFI | ABCFI ADEBCFI | ADEHI | ADG |  |
| Total Cost: |  | 18 | 17 | 16 | 13 | 12 |


in while loop:
curPath = pq.dequeue() (path is ADEF, priority is 11)
$\mathrm{v}=$ last element in curPath ( v is F )
mark v as visited
enqueue all unvisited neighbor paths onto q, with updated priorities based on new edge length

Visited Set: A, D, E, B, H, C, F, G

## Dijkstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

Let's look at Dijkstra from a to i:
priority
queue:

|  |  |  |  |  | front |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Path: |  |  | ADEFI | ABCFI | ADEBCFI | ADEHI |
| Total Cost: |  |  | 18 | 17 | 16 | 13 |


in while loop:
curPath = pq.dequeue() (path is ADG, priority is 12)
$\mathrm{v}=$ last element in curPath ( v is F )
mark v as visited
enqueue all unvisited neighbor paths onto $q$, with updated priorities based on new edge length

## Dijkstra's Algorithm in Practice

- From the start vertex, explore the neighbor nodes first, before moving to the next level neighbors, in priority order. From A to I:

Let's look at Dijkstra from a to i:
priority
queue:

|  |  |  |  |  | front |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Path: |  |  |  | ADEFI | ABCF | ADG |
| Total Cost: |  |  |  | 18 | 16 | 12 |


in while loop:
curPath = pq.dequeue() (path is ADEHI, priority is 13)
$v=$ last element in curPath ( v is I)
Stop! We've found the shortest path! Visited Set: A, D, E, B, H, C, F, G ADEHI

Who Was Edsgar Dijkstra?

History of Computing Tidbit: Edsger Dijkstra

- The Dutch academic Edsger Dijkstra was another giant in the field of computer science.
- He was one of the first scientists to call himself a "programmer" (and he almost couldn't get married because of it!)
- He started out with a degree in Theoretical Physics, but became enthralled with computers in the early 1950 s.

Who Was Edsgar Dijkstra?

Edsger Dijkstra

- Dijkstra was immensely influential in many fields of computing: compilers, operating systems, concurrent programming, software engineering, programming languages, algorithm design, and teaching (among others!)
- It would be hard to pin down what he is most famous for because he has influenced so much $C S$.


Goto Considered Harmful

Edsger Dijkstra

- Dijkstra was also influential in making programming more structured -- he wrote a seminal paper titled, "Soto Considered Harmful" where he lambasted the idea of the "goto" statement (which exists in C++ -you will rarely, if ever, use it!)


EH, SCREW GOOD PRACTICE. HOW BAD CAN IT BE?


Other Cool Dijkstra Facts

Other Reasons Dijkstra is cool:

- Might actually be Walter White
- Has the letters "ijk" adjacent in his name (is that why we use $i, j, k$ in our loops??)
- The Edsgar Dijkstra font! His early papers were hand-written, and he had beautiful handwriting. This font is the "Edsger
 Dijkstra" font!


## Dijkstra's is great, but we can do better!

If we want to travel from Stanford to San Francisco, Dijkstra's algorithm will look at path distances around Stanford. But, we know something about how to get to San Francisco -we know that we generally need to go Northwest from Stanford.

This is more information! Let's not only prioritize by weights, but also give some priority to the direction we want to go. E.g., we will add more information based on a heuristic, which could be direction in the case of a street map.


Dijkstra where each edge has cost 1

Dijkstra where each edge has cost 1

Dijkstra where each edge has cost 1
1?
1? $1 ?$
1?

Dijkstra where each edge has cost 1

Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1

| $1 ?$ |  |
| :---: | :---: |
| $1 ?$ | 1? |
| $2 ?$ | 1 |
| $2 ?$ | $2 ?$ |
| 2 |  |

Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1

| $2 ?$ | $1 ?$ |  |
| :---: | :---: | :---: | :---: |
| $2 ?$ | 1 | $1 ?$ |
| $2 ?$ | 1 | $2 ?$ |
| $2 ?$ |  |  |

Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1

|  | 2? 3? |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | 2? | 1 | 2 | 3? |
| 2? | 1 | \% | 1 | 2? |
|  | 2? | 1 | 2? |  |
|  |  | 2? |  |  |

Dijkstra where each edge has cost 1

|  | 2? 3? |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | 2? | 1 | 2 | 3? |
| $2 ?$ | 1 | is | 1 | 2? |
|  | 2? | 1 | 2 |  |
|  | 2? |  |  |  |

Dijkstra where each edge has cost 1

|  | 2? 3? |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | 2? | 1 | 2 | 3? |
| 2? | 1 | t ${ }^{\text {c }}$ | 1 | 2? |
|  | 2? | 1 | 2 | 3? |
|  | 2? 3? |  |  |  |

Dijkstra where each edge has cost 1

|  | 2? 3? |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | 2? | 1 | 2 | 3? |
| 2? | 1 |  | 1 | 2 |
|  | 2? | 1 | 2 | $3 ?$ |
|  |  |  | $3 ?$ |  |

Dijkstra where each edge has cost 1

|  | 2? 3? |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | 2? | 1 | 2 | 3? |  |
| 2? | 1 | 繧 | 1 | 2 | $3 ?$ |
|  | 2? | 1 | 2 | 3? |  |
|  |  |  | 3? |  |  |

Dijkstra where each edge has cost 1

|  | 2? 3? |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | 2? | 1 | 2 | 3? |  |
| 2? | 1 | स | 1 | 2 | $3 ?$ |
|  | 2 | 1 | 2 | $3 ?$ |  |
|  |  |  | 3 ? |  |  |

Dijkstra where each edge has cost 1

| 2? 3? |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | $2 ?$ | 1 | 2 | 3 ? |
| 2? | 1 | स | 1 | 2 |
| 3 ? | 2 | 1 | 2 | 3? |
| 3? 2? 3? |  |  |  |  |

Dijkstra where each edge has cost 1

| $2 ?$ |  |  |  |
| :---: | :---: | :---: | :---: |
| $2 ?$ | $3 ?$ |  |  |
| $2 ?$ | 1 | 2 | $3 ?$ |
| $2 ?$ | 1 |  | 1 |

Dijkstra where each edge has cost 1

|  |  | 2? 3? |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $3 ?$ | $2 ?$ | 1 | 2 | 3? |  |
| 3? | 2 | 1 | 絞 | 1 | 2 | 3? |
|  | 3? | 2 | 1 | 2 | 3? |  |
|  |  | 3 ? | 2 | 3? |  |  |
|  |  |  | 3? |  |  |  |

Dijkstra where each edge has cost 1

|  | 3? 2? 3? |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $3 ?$ | 2 | 1 | 2 | 3? |  |
| 3? | 2 | 1 | 动 | 1 | 2 | 3? |
|  | 3? | 2 | 1 | 2 | $3 ?$ |  |
|  |  | 3 ? | 2 | 3? |  |  |
|  |  |  | 3? |  |  |  |

Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1

|  |  |  | 3? |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 3 ? | 2 | $3 ?$ |  |  |
|  | 3 ? | 2 | 1 | 2 | $3 ?$ |  |
| 3 ? | 2 | 1 | 紓 | 1 | 2 | 3? |
| 4? | 3 | 2 | 1 | 2 | 3 | 4? |
|  | 4? | 3? | 2 | 3 | 4? |  |
|  |  |  | 3? | 4? |  |  |

Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1

|  |  | 3? 4 ? |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | ? 3 ? | 2 | 3 | 4? |
|  | 4 ? 3 | 2 | 1 | 2 | 3? |
| 4? 3 | 32 | 2 | \% | 1 | 2 |
|  | 4 ? 3 | 3 | 1 | 2 | 3 |
|  |  | ? 3 ? | 2 | 3 | 4? |
|  |  |  | 3? |  |  |

Dijkstra where each edge has cost 1

|  |  | 3? 4? |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 4? | $3 ?$ | 2 | 3 | 4? |  |
| $4 ?$ | 3 | 2 | 1 | 2 | 3 | 4? |
| 4? 3 | 2 | 1 | W | 1 | 2 | 3? |
| 4? | 3 | 2 | 1 | 2 | 3 | 4? |
|  | 4? | 3 ? | 2 | 3 | 4? |  |
|  |  |  | 3 ? | 4? |  |  |

Dijkstra where each edge has cost 1

|  | 4? 3? 4? |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 4 ? | 3 | 2 | 3 | 4? |  |
| $4 ?$ | 3 | 2 | 1 | 2 | 3 | 4? |
| 4? 3 | 2 | 1 | \% | 1 | 2 | 3? |
| 4? | 3 | 2 | 1 | 2 | 3 | 4? |
|  | 4 ? | 3 ? | 2 | 3 | 4? |  |
|  |  |  |  |  |  |  |

Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1

|  | 4 ? |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 4? | 3 | 4 ? |  |  |  |
|  | 4 ? | 3 | 2 | 3 | 4? |  |  |
| 4 ? | 3 | 2 | 1 | 2 | 3 | 4? |  |
| 4? 3 | 2 | 1 | H2 | 1 | 2 | 3 | ? |
| 4 ? | 3 | 2 | 1 | 2 | 3 | 4? |  |
|  | 4? | 3 | 2 | 3 | 4? |  |  |
|  |  | 4? | 3 | 4? |  |  |  |
|  |  |  | 4 ? |  |  |  |  |

Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1

|  |  |  | 4 ? |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 5? | 4? | 3 | 4? |  |  |  |
| $5 ?$ | ? | 3 | 2 | 3 | 4 | 5? |  |
| 4 ? | ? 3 | 2 | 1 | 2 | 3 | 4? 5 |  |
| 4? 3 |  | 1 | \% | 1 | 2 | 34 | 4 5? |
| 4 ? | ? 3 | 2 | 1 | 2 | 3 | 4? 5 |  |
|  | 4 ? | 3 | 2 | 3 | 4? |  |  |
|  |  | 4? | 3 | 4? |  |  |  |
|  |  |  | 4 ? |  |  |  |  |

Dijkstra where each edge has cost 1

|  |  |  | 4? |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 4? | 3 | 4? |  |  |  |
| $5 ?$ | ? 4 | 3 | 2 | 3 | 4 | 5? |  |
| 4 ? | ? 3 | 2 | 1 | 2 | 3 | 4 | 5? |
| 4? 3 | 3 | 1 | \% | 1 | 2 | 3 | 45 |
| 4? | ? 3 | 2 | 1 | 2 | 3 | 4? | 5? |
|  | 4? |  | 2 | 3 | 4? |  |  |
|  |  | 4? | 3 | 4? |  |  |  |
|  |  |  | 4 ? |  |  |  |  |

Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1

| 4 ? |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | 5? |  | 3 | 4? |  |  |  |
|  | 5? | 4 | 3 | 2 | 3 | 4 | 5? |  |
| 5? | 4 | 3 | 2 | 1 | 2 | 3 | 4 |  |
| 4? | 3 | 2 | 1 | \% | 1 | 2 | 3 | 5 |
|  | 4 ? | 3 | 2 | 1 | 2 | 3 | 4? 5 |  |
|  | 5 ? | 4 | 3 | 2 | 3 | 4? |  |  |
|  |  | 5? | 4? | 3 | 4? |  |  |  |
|  |  |  |  | 4 ? |  |  |  |  |

Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1 5? 4?

|  |  | 5? | 4 | 3 |  | ? 5 |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $5 ?$ | 4 | 3 | 2 | 3 | 4 | 5 | ? |  |
| 5 ? | 4 | 3 | 2 | 1 | 2 | 3 |  | 45 |  |
| 4 | 3 | 2 | 1 | स | 1 | 2 | 3 | 34 | 5? |
| 5? | 4 | 3 | 2 | 1 | 2 | 3 |  | 4? 5 |  |
|  | 5? |  | 3 | 2 | 3 | 4? |  |  |  |
| 5? 4? 3 4? |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 4? |  |  |  |  |  |

Dijkstra where each edge has cost 1 5? 4?

|  |  | 5? | 4 | 3 | 4 ? | $5 ?$ |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 5? | 4 | 3 | 2 | 3 | 4 |  | ? |  |
| 5? | 4 | 3 | 2 | 1 | 2 | 3 |  | 45 |  |
| 4 | 3 | 2 | 1 | \% | 1 | 2 | 3 | 3 | 5? |
| 5 ? | 4 | 3 | 2 | 1 | 2 | 3 |  | 45 |  |
|  | 5? | 4 | 3 | 2 | 3 |  | ? 5 |  |  |
| 5? 4? 3 4? |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Dijkstra where each edge has cost 1 5? 4?


Dijkstra where each edge has cost 1 5? 4?

|  |  | $5 ?$ | 4 | 3 | 4 ? | $5 ?$ |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 5? | 4 | 3 | 2 | 3 | 4 | 5 |  |  |
| 5? | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5? |  |
| 4 | 3 | 2 | 1 | 动 | 1 | 2 | 3 | 4 | 5? |
| 5 ? | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5? |  |
|  | 5? | 4 | 3 | 2 | 3 | 4? | 5 |  |  |
|  |  | 5? |  | 3 | 4 | 5? |  |  |  |
|  |  |  | 5? | 4 | 5 ? |  |  |  |  |

Dijkstra where each edge has cost 1 5? 4?

| 5? |  |  | 4 | 3 4? 5? |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 5? | 4 | 3 | 2 | 3 | 4 |  | 5? |  |
| 5? | 4 | 3 | 2 | 1 | 2 | 3 |  | 4 | ? |
| 4 | 3 | 2 | 1 | H | 1 | 2 |  | 3 | 4 |
| 5 ? | 4 | 3 | 2 | 1 | 2 | 3 |  | 4 | ? |
|  | $5 ?$ | 4 | 3 | 2 | 3 | 4 ? | ? 5 |  |  |
|  |  | 5 ? | 4 | 3 | 4 | 5 |  |  |  |
|  |  |  | 5? |  | 5? |  |  |  |  |

Dijkstra where each edge has cost 1 5? 4?

|  |  | 5? | 4 | 3 | 4 | ? 5 |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 5? | 4 | 3 | 2 | 3 | 3 |  | 5? |  |  |
| 5? | 4 | 3 | 2 | 1 | 2 | 2 | 3 | 4 | 5? |  |
| 4 | 3 | 2 | 1 | \% | 1 | 1 | 2 | 3 | 4 | $5 ?$ |
| 5 ? | 4 | 3 | 2 | 1 | 2 |  | 3 | 4 | 5? |  |
|  | $5 ?$ | 4 | 3 | 2 | 3 | 3 | 4 | $5 ?$ |  |  |
|  |  | 5 ? | 4 | 3 | 4 | 4 | 5? |  |  |  |
|  |  |  | 5 ? |  | 5 |  |  |  |  |  |

Dijkstra where each edge has cost 1
5? 4? 5?

|  |  | 5? | 4 | 3 | 4 | $5 ?$ |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 5? | 4 | 3 | 2 | 3 | 4 | $5 ?$ |  |  |
| 5? | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5? |  |
| 4 | 3 | 2 | 1 | त | 1 | 2 | 3 | 4 | $5 ?$ |
| 5? | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5? |  |
|  | 5 ? | 4 | 3 | 2 | 3 | 4 | 5? |  |  |
|  |  | 5? | 4 | 3 | 4 | 5? |  |  |  |
|  |  |  | 5 ? | 4 | 5 ? |  |  |  |  |

Dijkstra where each edge has cost 1
5? 4 5?

|  |  | 5? | 4 | 3 | 4 | 5 ? |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 5? | 4 | 3 | 2 | 3 | 4 | 5? |  |
| 5? | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5? |
| 4 | 3 | 2 | 1 | H | 1 | 2 | 3 | 4 |
| 5 ? | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5? |
|  | 5? | 4 | 3 | 2 | 3 | 4 | 5? |  |
|  |  | 5? | 4 | 3 | 4 | 5? |  |  |
|  |  |  | 5? |  | 5? |  |  |  |

Dijkstra where each edge has cost 1
5? 4 5? 6?


Dijkstra where each edge has cost 1
5? 4 5? 6?


Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1

|  |  |  |  | 4 | 5? |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 6 ? |  | 4 | 3 | 4 | 5 | $6 ?$ |  |
| 6 ? | 5 | 4 | 3 | 2 | 3 | 4 | 5? |  |
| 5? | 4 | 3 | 2 | 1 | 2 | 3 | 45 |  |
| 4 | 3 | 2 | 1 | \# | 1 | 2 | 3 |  |
| 5? | 4 | 3 | 2 | 1 | 2 | 3 | 45 |  |
|  | 5 ? | 4 | 3 | 2 | 3 | 4 | 5 |  |
|  | 6 ? | 5 | 4 | 3 | 4 |  |  |  |
|  |  |  | 5? |  |  |  |  |  |

Dijkstra where each edge has cost 1


Dijkstra where each edge has cost 1
5? 4 5? 6?

| 6 Why are we looking 5 in this direction? |  |  |  |  |  |  | 6? |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |  |  |
|  |  |  | 2 | 1 | 2 | 3 | 4 | $5 ?$ |
| 4 | 3 | 2 | 1 | \% | 1 | 2 | 3 | 4 |
| 5 ? | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5? |
|  | 5? | 4 | 3 | 2 | 3 | 4 | 5 | 6? |
|  | 6 ? | 5 | 4 | 3 | 4 | 5? | $6 ?$ |  |
|  |  |  |  | 4 | 5? |  |  |  |

## Dijkstra where each edge has cost 1

 5? 4 5? 6?| 6 Why arewe looking 5 |  |  |  |  |  | 6? |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 5 |  |  |  |  |  |  |
|  | 13 | 2 | 1 | 2 | 3 | 4 | 5? |
| 4 | 32 | 1 | w | 1 | 2 | 3 | 4 |
| 5 ? | 43 | 2 | 1 | 2 | 3 | 4 | $5 ?$ |
|  | 5 |  |  |  | 4 | 5 | 6? |
|  |  | m | that |  |  | 6? |  |

## Dijkstra Priority

## priority(u) = distance(s, u)



Priority of the path that ends in $u$

## Ideal Priority

## priority(u) = distance(s, u) + futureCost(u, t)



Priority of the path that ends in u

## Future Cost?











|  |  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | $0$ |  |  |
|  |  |  |  |  | - |  |  |  | ■ |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |



(1)



|  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  | $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 2+ \\ & 5 ? \end{aligned}$ |  |  |  |  |  |  |  |
|  | $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | む | 1 | $\begin{aligned} & 2+ \\ & 3 ? \end{aligned}$ |  |  | $\pm$ |  |  |  |
|  |  | $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 2+ \\ & 5 ? \end{aligned}$ |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |




(1)


|  | $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 2+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 3+ \\ & 4 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 3 ? \end{aligned}$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | \% | 1 | 2 | 3 | $\begin{aligned} & 4+ \\ & 1 ? \end{aligned}$ | * |
|  | $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | $2+$ | $\begin{aligned} & 3+ \\ & 4 ? \\ & \hline \end{aligned}$ | $\begin{aligned} & 4+ \\ & 3 ? \end{aligned}$ |  |  |


| $1+$ | $2+$ | $3+$ | $4+$ |  |  |
| :--- | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |
| $6 ?$ | $5 ?$ | $4 ?$ | $3 ?$ |  |  |
| $1+$ | 5 | 1 | 2 | 3 | $4+$ |
| $6 ?$ | $1 ?$ |  |  |  |  |
| $1+$ | $2+$ | $3+$ | $4+$ |  |  |
|  | $6 ?$ | $5 ?$ | $4 ?$ | $3 ?$ |  |

(1)

|  | $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 2+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 3+ \\ & 4 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 3 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 2 ? \end{aligned}$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | c | 1 | 2 | 3 | 4 | $\begin{aligned} & 5+ \\ & 0 ? \end{aligned}$ |
|  | $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 2+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 3+ \\ & 4 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 3 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & \text { 2? } \end{aligned}$ |  |



|  | $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 2+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 3+ \\ & 4 ? \end{aligned}$ | $4+$ | $\begin{aligned} & 5+ \\ & 2 ? \end{aligned}$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $1+$ | U | 1 | 2 | 3 | 4 | $\Sigma$ |
|  | $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 2+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 3+ \\ & 4 ? \\ & \hline \end{aligned}$ | $\begin{aligned} & 4+ \\ & 3 ? \end{aligned}$ | $5+$ |  |

That was easy...

## ... a little too easy



## Ideal Priority

## priority(u) = distance(s, $u)+$ futureCost( $u, t)$



Priority of the path that ends in u

## A* Priority

## priority(u) = distance(s, u) + heuristic(u, t)



Priority of the path that ends in $u$

## Admissible Heuristic

## Definition: An admissible heuristic always underestimates the true cost.

Thus: "even in the best case scenario, this path is still terrible..."

## Admissible Heuristic





## "Manhattan" distance

```
function h(start,goal)
    dRows = abs(start.row - goal.row);
    dCols = abs(start.col - goal.col);
    return dRows + dCols
}
```


## Recall Djjkstra...

Make a PriorityQueue todo-list of paths
Put a path with just the start in the todo-list
While the todo-list isn't empty

1. Take a path out of the todo-list
2. Call the last node in the path "currNode"
3. If "currNode" is the goal, you are done.
4. If you have seen currNode before, skip it.
5. for all neighbors of currNode

Make a newPath $=$ path + neighbor
Add the new path to the todo-list Priority = pathLength

## A Star

Make a PriorityQueue todo-list of paths
Put a path with just the start in the todo-list
While the todo-list isn't empty

1. Take a path out of the todo-list
2. Call the last node in the path "currNode"
3. If "currNode" is the goal, you are done.
4. If you have seen currNode before, skip it.
5. for all neighbors of currNode

Make a newPath $=$ path + neighbor
Add the new path to the todo-list

```
Priority = pathLength + h(neighbor, goal)
```


## A Star

Make a PriorityQueue todo-list of paths
Put a path with just the start in the todo-list
While the kodo-list isn't empty

1. Take a path out of the todo-list
2. Call the fast node in the path "currNode"
3. If "chWhat is the priority of the start e done.
4. If yo path? skip it.
5. for all neighbors of currNode

Make a newPath $=$ path + neighbor
Add the new path to the todo-list

```
Priority = pathLength + h(neighbor, goal)
```


## A Star

Make a PriorityQueue todo-list of paths
Put a path with just the start in the todo-list
While the kodo-list isn't empty

1. Take a path out of the todo-list
2. Call the fast node in the path "currNode"
3. If "ciWhat is the priority of the start done.
4. If yop path? skip it.

Make a newPath $=$ path + neighbor
Add the new path to the todo-list
```
Priority = pathLength + h(neighbor, goal)
```




(1)


|  |  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | $1$ | $\begin{aligned} & 24 \\ & 5 ? \end{aligned}$ |  |  |  |  |  |  |  |
|  |  |  |  |  | $1$ |  |  |  |  |  |  |  |
|  |  |  |  | $1$ | $\begin{aligned} & 2+ \\ & 5 ? \end{aligned}$ |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |



|  | $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 2+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 3+ \\ & 4 ? \end{aligned}$ |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $1+$ | $\pm$ | 1 | 2 | I |  |  |  |  |
|  | $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 2+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 3+ \\ & 4 ? \end{aligned}$ |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |



|  | $\begin{aligned} & 2+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 2+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 3+ \\ & 4 ? \end{aligned}$ |
| :---: | :---: | :---: | :---: | :---: |
| $\begin{aligned} & 2+ \\ & 7 ? \end{aligned}$ | 1 | \% | 1 | 2 |
|  | $\begin{aligned} & 2+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 1+ \\ & 6 ? \end{aligned}$ | $2+$ | $\begin{aligned} & 3+ \\ & 4 ? \end{aligned}$ |



|  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | $\square$ |  |
|  | $\begin{aligned} & 2+ \\ & 7 ? \end{aligned}$ |  |  |  |  |  |  |  | I |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | $\begin{aligned} & 34 \\ & 6 ? \end{aligned}$ |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |








(1)




|  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | \\| | $2$ |  |  |  |  |  |  |  |
|  |  |  | $\xi$ |  |  |  |  | $\underset{w}{ }$ |  |  |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ |  | I |  |  |  |  |  |  |  |  |
|  |  |  | $\begin{aligned} & 2+ \\ & 7 ? \end{aligned}$ |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |









|  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  | $\begin{aligned} & 34 \\ & 8 ? \end{aligned}$ |  |  |  |  |  |  |  |  |
|  |  |  |  | $\begin{aligned} & 34 \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 44 \\ & 5 ? \end{aligned}$ |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | I |  |  |
|  |  |  |  |  |  |  |  |  | I |  |  |
|  |  | $2$ |  |  | 3 |  |  |  |  |  |  |
|  |  | $\begin{aligned} & 34 \\ & 8 ? \end{aligned}$ |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |








|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \\ & \hline \end{aligned}$ | 2 | 3 | 4 | $\begin{aligned} & 5+ \\ & 4 ? \end{aligned}$ |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |
| $\begin{aligned} & 3+ \\ & 82 \end{aligned}$ | 2 | 1 | 3 | 1 | 2 |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | $\begin{aligned} & 5+ \\ & 4 ? \end{aligned}$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ |  |


|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $4+$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | $\begin{aligned} & 5+ \\ & 4 ? \end{aligned}$ |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |
| $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 1 | 2 |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ |  |


|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | $\begin{aligned} & 5+ \\ & 4 ? \end{aligned}$ |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |  |
| $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 1 | 2 |  |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | $\begin{aligned} & 6+ \\ & 3 ? \end{aligned}$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ |  |



|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | $\begin{aligned} & 5+ \\ & 4 ? \end{aligned}$ |  |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |  |  |
| $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | W | 1 | 2 |  |  | A |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | $\begin{aligned} & 7+ \\ & 2 ? \end{aligned}$ |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | $\begin{aligned} & 7+ \\ & \text { 2? } \end{aligned}$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 7+ \\ & 4 ? \end{aligned}$ |  |


|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 |  |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |  |  |
| $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 3 | 1 | 2 |  |  | A |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | $\begin{aligned} & 7+ \\ & 2 ? \end{aligned}$ |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | $\begin{aligned} & 7+ \\ & 2 ? \end{aligned}$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 7+ \\ & 4 ? \end{aligned}$ |  |


|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $5+$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | $\begin{aligned} & 6+ \\ & 3 ? \\ & \hline \end{aligned}$ |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |  |  |
| $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 3 | 1 | 2 |  |  | $\pm$ |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | $\begin{aligned} & 7+ \\ & 2 ? \end{aligned}$ |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | $7+$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $5+$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $7+$ |  |


|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | $\begin{aligned} & 6+ \\ & 3 ? \end{aligned}$ |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |  |  |
| $\begin{aligned} & 3+ \\ & 8 ? \\ & \hline \end{aligned}$ | 2 | 1 | 3 | 1 | 2 |  |  | t |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | 7 |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | $\begin{aligned} & 7+ \\ & 2 ? \end{aligned}$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $4+$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 7+ \\ & 4 ? \end{aligned}$ |  |


|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $5+$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | $\begin{aligned} & 6+ \\ & 3 ? \\ & \hline \end{aligned}$ |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |  |  |
| $3+$ | 2 | 1 | 3 | 1 | 2 |  | $\begin{aligned} & 8+ \\ & 1 ? \end{aligned}$ | A |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | 7 | $8+$ $1 ?$ |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | $7+$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $5+$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $7+$ |  |


|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | $\begin{aligned} & 6+ \\ & 3 ? \\ & \hline \end{aligned}$ |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |  |  |
| $3+$ | 2 | 1 | 3 | 1 | 2 |  | $\begin{aligned} & 8+ \\ & 1 ? \end{aligned}$ | T |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | 7 | $\begin{aligned} & 8+ \\ & 1 ? \end{aligned}$ |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | 7 |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $7+$ |  |


|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | $\begin{aligned} & 6+ \\ & 3 ? \end{aligned}$ |  |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |  |  |  |
| $\begin{aligned} & 3+ \\ & 8 ? \\ & \hline \end{aligned}$ | 2 | 1 | 2 | 1 | 2 |  | $\begin{aligned} & 8+ \\ & 1 ? \end{aligned}$ | t |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | 7 | $\begin{aligned} & 8+ \\ & 1 ? \end{aligned}$ |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | 7 | $\begin{aligned} & 8+ \\ & 3 ? \end{aligned}$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 7+ \\ & 4 ? \end{aligned}$ | $\begin{aligned} & 8+ \\ & 3 ? \end{aligned}$ |  |


|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | $\begin{aligned} & 6+ \\ & 3 ? \end{aligned}$ |  |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |  |  |  |
| $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | N | 1 | 2 |  | 8 | $\pm$ |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | 7 | $\begin{aligned} & 8+ \\ & 1 ? \end{aligned}$ |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | 7 | $\begin{aligned} & 8+ \\ & 3 ? \end{aligned}$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 7+ \\ & 4 ? \end{aligned}$ | $\begin{aligned} & 8+ \\ & 3 ? \end{aligned}$ |  |


|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | $\begin{aligned} & 6+ \\ & 3 ? \end{aligned}$ |  |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | $\begin{aligned} & 9+ \\ & \text { 2? } \end{aligned}$ |  |  |
| $\begin{aligned} & 3+ \\ & 8 ? \\ & \hline \end{aligned}$ | 2 | 1 | 2 | 1 | 2 |  | 8 | $\begin{aligned} & 9+ \\ & 0 . ? \end{aligned}$ |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | 7 | $\begin{aligned} & 8+ \\ & 1 ? \end{aligned}$ |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | 7 | $\begin{aligned} & 8+ \\ & 3 ? \end{aligned}$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 7+ \\ & 4 ? \end{aligned}$ | $\begin{aligned} & 8+ \\ & 3 ? \end{aligned}$ |  |


|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 |  |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | $\begin{aligned} & 9+ \\ & \text { 2? } \end{aligned}$ |  |  |
| $\begin{aligned} & 3+ \\ & 8 ? \\ & \hline \end{aligned}$ | 2 | 1 | * | 1 | 2 |  | 8 | $\begin{aligned} & 9+ \\ & 0 ? \end{aligned}$ |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | 7 | $\begin{aligned} & 8+ \\ & 1 ? \end{aligned}$ |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \\ & \hline \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | 7 | $8+$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 7+ \\ & 4 ? \end{aligned}$ | $\begin{aligned} & 8+ \\ & 3 ? \end{aligned}$ |  |


|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 7+ \\ & 4 ? \end{aligned}$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | $7+$ |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | $\begin{array}{\|l} 7+ \\ 2 ? \end{array}$ |  |  |
| $\begin{aligned} & 3+ \\ & 8 ? \\ & \hline \end{aligned}$ | 2 | 1 | A | 1 | 2 |  | 8 | $\begin{aligned} & 9+ \\ & 0+? \end{aligned}$ |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | 7 | $\begin{aligned} & 8+ \\ & 1 ? \end{aligned}$ |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | 7 | $8+$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 7+ \\ & 4 ? \end{aligned}$ | $\begin{aligned} & 8+ \\ & 3 ? \end{aligned}$ |  |


|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 7+ \\ & 4 ? \end{aligned}$ |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | $\begin{aligned} & 7+ \\ & 2 ? \end{aligned}$ |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | $\begin{array}{\|l} 7+ \\ 2 ? \end{array}$ |  |  |
| $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | W | 1 | 2 |  | 8 | W |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | 7 | $\begin{aligned} & 8+ \\ & 1 ? \end{aligned}$ |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | 7 | $\begin{aligned} & 8+ \\ & 3 ? \end{aligned}$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 7+ \\ & 4 ? \end{aligned}$ | $\begin{aligned} & 8+ \\ & 3 ? \end{aligned}$ |  |

What Djjkstra would have selected!

| 8 | 7 | 6 | 5 | 4 | 5 | 6 | 7 | 8 | 9? |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 7 | 6 | 5 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | $9 ?$ |  |
| 6 | 5 | 4 | 3 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9? |
| 5 | 4 | 3 | 2 | 1 | 2 | 3 |  | 7 | 8 | $9 ?$ |  |
| 4 | 3 | 2 | 1 | そ | 1 | 2 |  | 8 | \% |  |  |
| 5 | 4 | 3 | 2 | 1 | 2 | 3 |  | 7 | 8 | 9? |  |
| 6 | 5 | 4 | 3 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | $9 ?$ |
| 7 | 6 | 5 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9? |  |
| 8 | 7 | 6 | 5 | 4 | 5 | 6 | 7 | 8 | 9? |  |  |

Why underestimate?

You only ignore paths that in the best case are worse than your current path.

Using our heuristic, the path from start to goal that goes through this node is at least cost 11

|  |  |  | $3+$ $8 ?$ | $4+$ $7 ?$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 7+ \\ & 4 ? \end{aligned}$ |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | $\begin{aligned} & 7+ \\ & \text { 2? } \end{aligned}$ |  |  |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  | $\begin{aligned} & 7+ \\ & 2 ? \end{aligned}$ |  |  |  |  |
| $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | N | 1 | 2 |  | 8 | \% |  |  |  |
|  | $3+$ $8 ?$ | 2 | 1 | 2 | 3 |  | 7 | $\begin{aligned} & 8+ \\ & 1 ? \end{aligned}$ |  |  |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | 5 | 6 | 7 | $\begin{aligned} & 8+ \\ & 3 ? \end{aligned}$ |  |  |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $5+$ | $\begin{aligned} & 6+ \\ & 5 ? \end{aligned}$ | $\begin{aligned} & 7+ \\ & 4 ? \end{aligned}$ | $\begin{aligned} & 8+ \\ & 3 ? \end{aligned}$ |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

Imagine if we overestimate

|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $5+$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | $\begin{aligned} & 5+ \\ & 4 ? \end{aligned}$ |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |
| $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 3 | 1 | 2 |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | $\begin{aligned} & 5+ \\ & 4 ? \end{aligned}$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ |  |

Imagine if we overestimate

|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $5+$ |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | $\begin{array}{r} 5+ \\ 20 ? \\ \hline \end{array}$ |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |
| $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 3 | 1 | 2 |  |
|  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 1 | 2 | 3 |  |
|  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | 2 | 3 | 4 | $\begin{array}{r} 5+ \\ 20 ? \end{array}$ |
|  |  |  | $\begin{aligned} & 3+ \\ & 8 ? \end{aligned}$ | $\begin{aligned} & 4+ \\ & 7 ? \end{aligned}$ | $\begin{aligned} & 5+ \\ & 6 ? \end{aligned}$ |  |

## More Detail on A*: Choice of Heuristic

## priority(u) = distance(s, u) + heuristic(u, t)



We want to underestimate the cost of our heuristic, by why?
Let's look at the bounds of our choices:
heuristic( $u, t)=0$
heuristic $(u, t)=$ underestimate
heuristic $(u, t)=$ perfect distance
heuristic(u,t) = overestimate

## More Detail on A*: Choice of Heuristic

## priority(u) = distance(s, u) + heuristic(u, t)



We want to underestimate the cost of our heuristic, by why? Let's look at the bounds of our choices:
heuristic $(u, t)=0$
heuristic $(u, t)=$ underestimate
heuristic( $u, t$ ) $=$ perfect distance
Same as Dijkstra heuristic(u,t) = overestimate

## More Detail on $A^{*}$ : Choice of Heuristic

$$
\text { priority }(u)=\operatorname{distance}(s, u)+\text { heuristic }(u, t)
$$



We want to underestimate the cost of our heuristic, by why? Let's look at the bounds of our choices:
heuristic $(u, t)=0$
heuristic $(u, t)=$ underestimate
heuristic(u,t) = perfect distance
heuristic(u,t) = overestimate

Will be the same or faster than Dijkstra, and will find the shortest path (this is the only "admissible" heuristic for $A^{*}$.

## More Detail on $A^{*}$ : Choice of Heuristic

$$
\text { priority }(u)=\operatorname{distance}(s, u)+\text { heuristic }(u, t)
$$



We want to underestimate the cost of our heuristic, by why?
Let's look at the bounds of our choices:
heuristic $(u, t)=0$
heuristic $(u, t)=$ underestimate
heuristic $(u, t)=$ perfect distance heuristic(u,t) = overestimate

Will only follow the best path, and will find the best path fastest (but requires perfect knowledge)

## More Detail on A*: Choice of Heuristic

## priority(u) = distance(s, u) + heuristic(u, t)



We want to underestimate the cost of our heuristic, by why?
Let's look at the bounds of our choices:
heuristic $(u, t)=0$
heuristic $(u, t)=$ underestimate heuristic $(u, t)=$ perfect distance heuristic $(u, t)=$ overestimate

Won't necessarily find shortest path (but might run even faster)

## Admissible Heuristic

## Definition: An admissible heuristic always underestimates the true cost.



## Why doesn't Google Maps Pre-Compute Directions?

- How many nodes are in the Google Maps graph?
- About 75 million
- How many sets of directions would they need to generate?
- (roughly) N2
- How long would that take?
- $6 \times 10^{15}$ seconds
- Or... 190 million years


## What Heuristics Could Google Maps Use?

- As the crow flies
- Calculate the straight-line distance from A to B, and divide by the speed on the fastest highway
- Landmark heuristic
- Find the distance from $A$ and $B$ to a landmark, calculate the difference (distance $<\operatorname{abs}(A-B)$ )
- All of these and more?
- You can use multiple heuristics and choose the max


## Extra Slides

