EE5110/6110: Special Topics in Automation & Control—Autonomous systems

Motion planning (II)

Sunan Huang

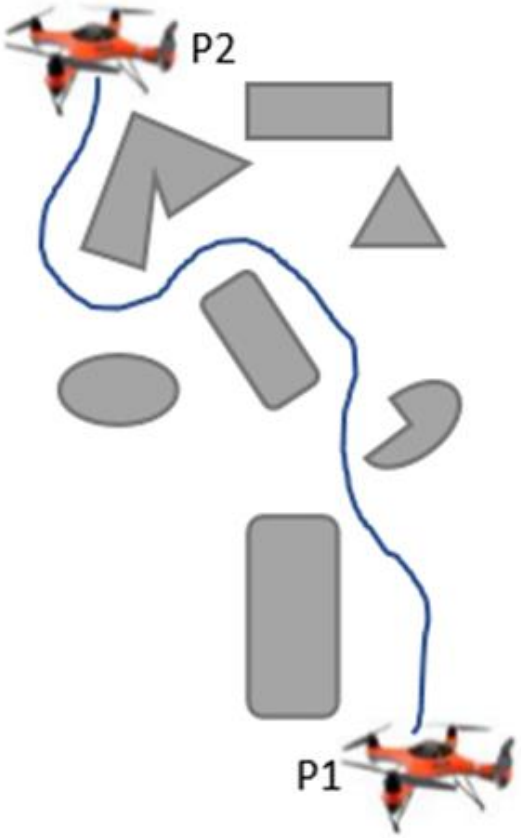
Temasek Laboratories

National University of Singapore

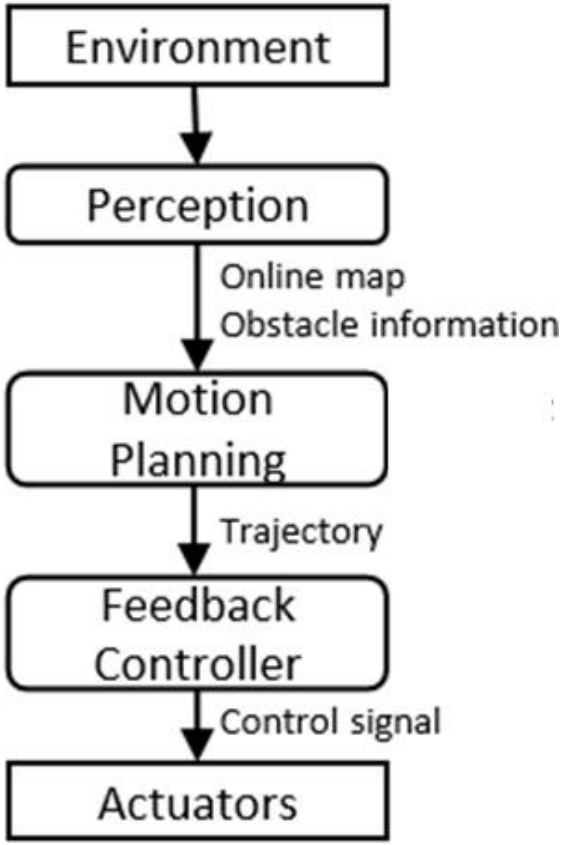
— Oct 6. 2020 —

NUS National University of Singapore

Review: Motion planning concept



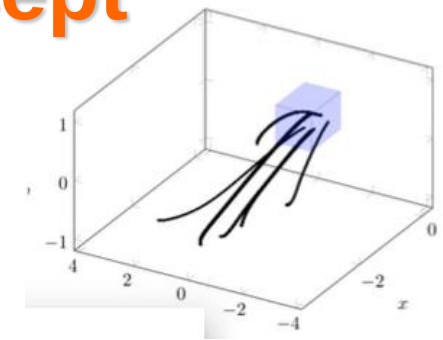
(a)



(b)

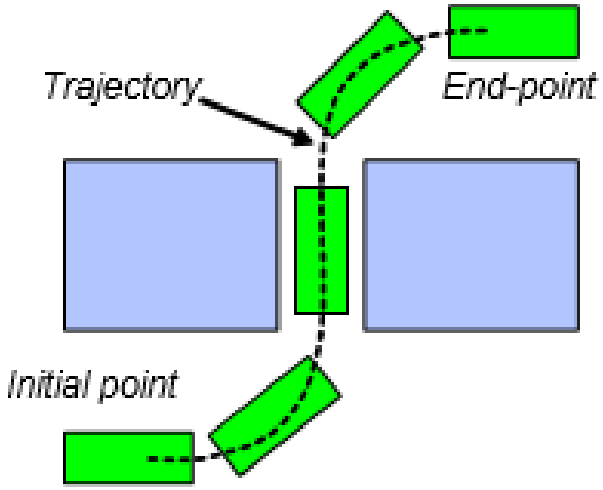
NUS National University of Singapore

Review: Motion planning concept



Motion planning includes

- **C-space**
- **Path planning (planner)**
- **Trajectory planning.**

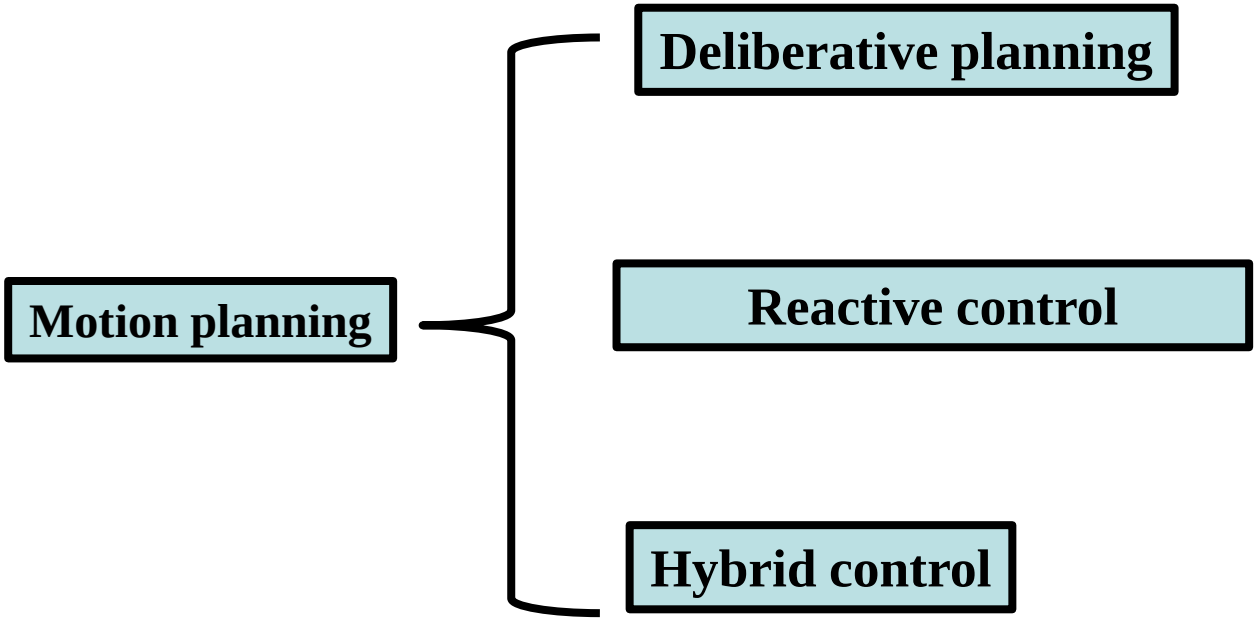


Continuous turning

NUS National University of Singapore

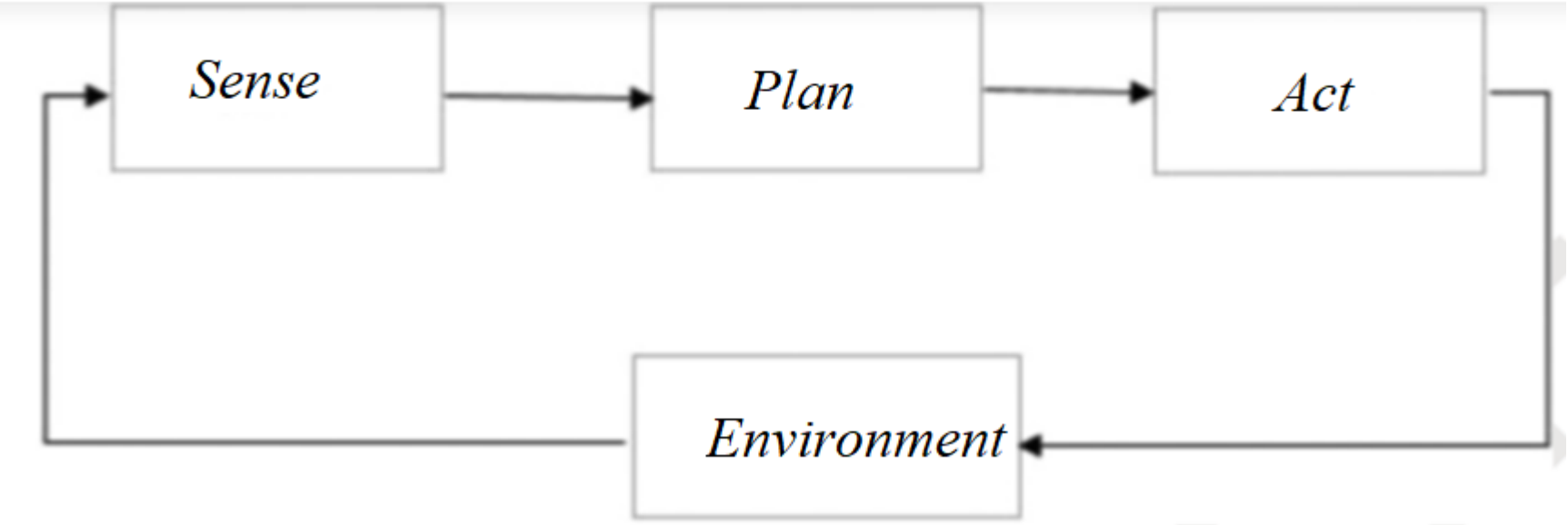
Review: Overview of motion planning architectures (planner)

The control architecture defines how the works of generating actions from perceiving the world around us are organized. Basically, motion planning can be grouped into three classes: deliberative planning architecture, reactive architecture and hybrid architecture. Reactive control=trajectory planning?



NUS National University of Singapore

Deliberative planning architecture (hierarchical form, static obstacles in map)

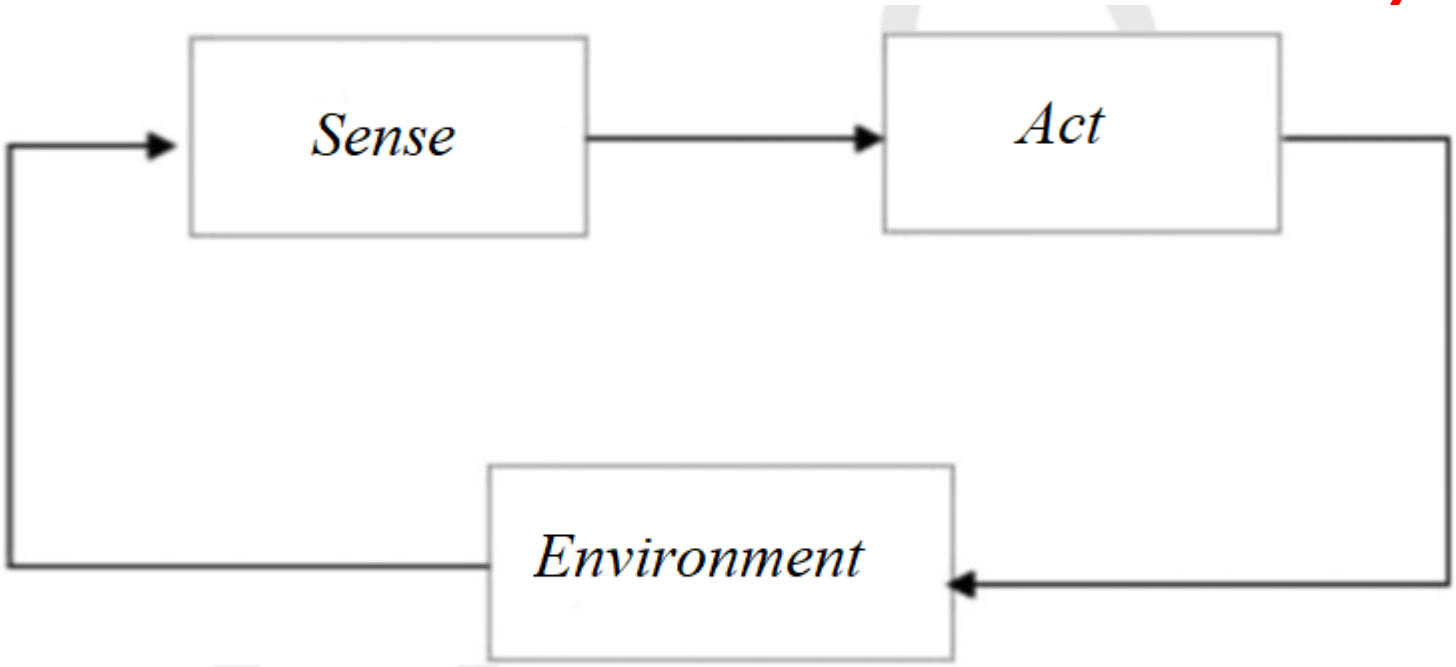


NUS National University of Singapore

Reactive control architecture

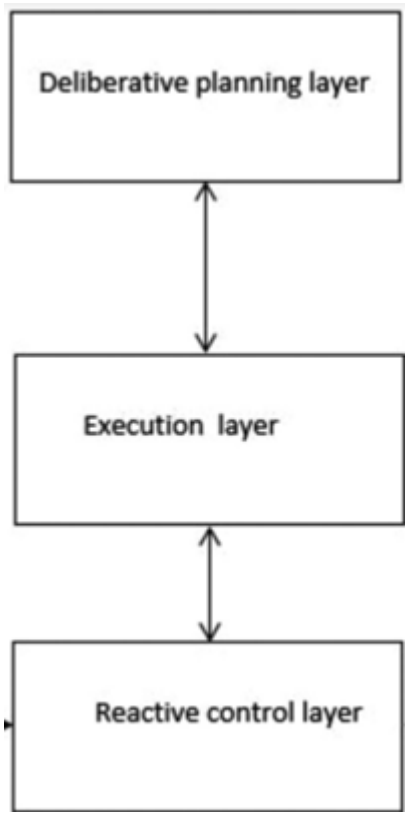
(or behavior based control, updated environment handling moving obstacles)

~~Plan~~



NUS National University of Singapore

Planner: global solution (global planner) and local solution (local planner)



?

- **Global solution**

  (low frequency, static obstacles)

  - Graph search
  - Conflict detection and resolution

- **Local solution**

  (high frequency, moving obstacles)

  - Model predictive control
  - Potential function
  - Geometric guidance

Learning Objective

- Understand the motion planning ------ A* algorithm.
- Can use this algorithm to search the shortest path for a simple environment

NUS National University of Singapore

Topics To Be Covered

1. **Search concept**
2. **Dijkstra algorithm**
3. ***heuristic* function**
4. **A* algorithm**
5. **Sampling-based search algorithm**

NUS National University of Singapore

Autonomous Systems : Motion planning

The core of the autonomous systems is motion planning.

Why? Localization, motion planning, motion control

Finding a path

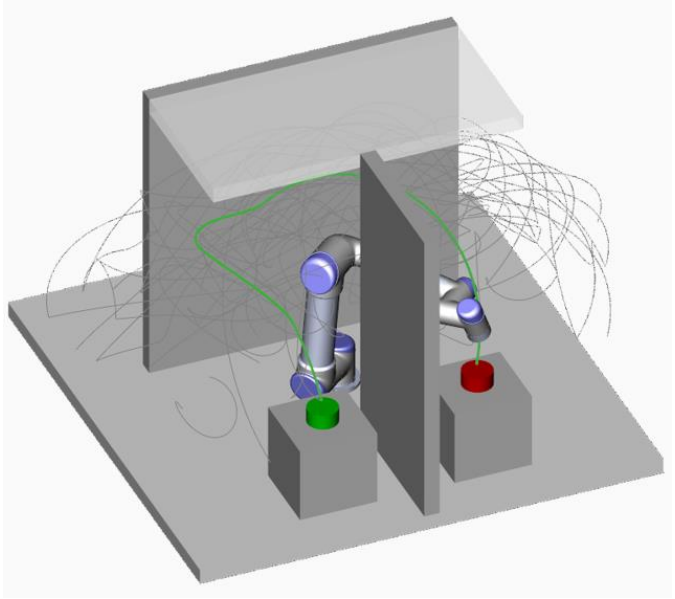
Simple or challenging?

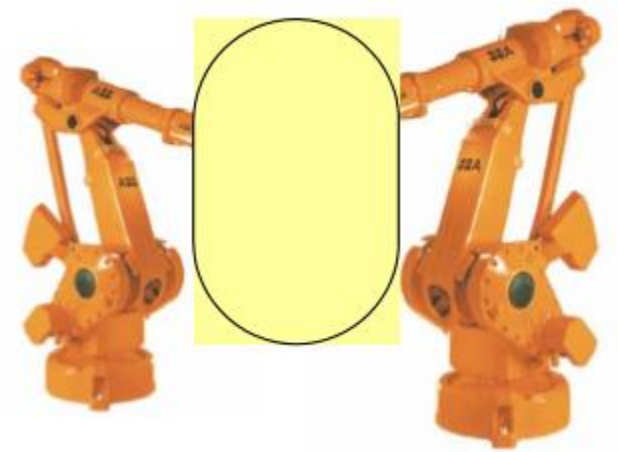
NUS National University of Singapore

Motion planning

Finding the shortest pa

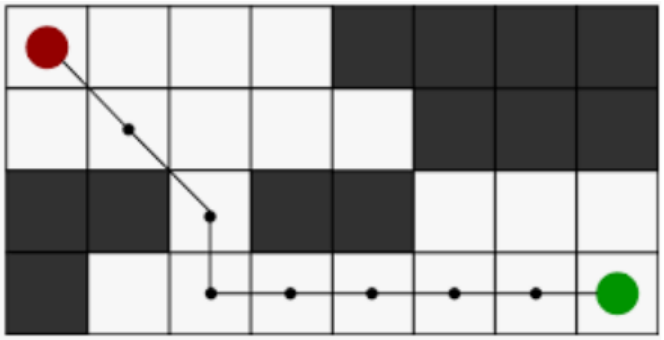
Simple or challenging

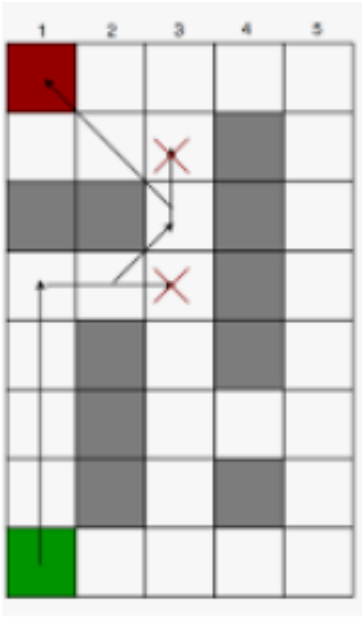


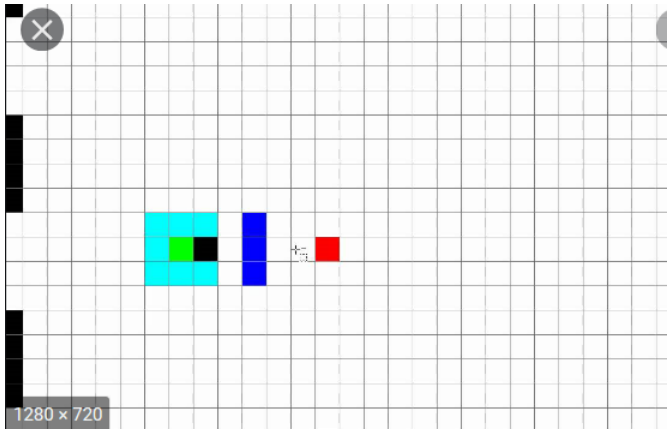


NUS National University of Singapore

Could you find the shortest path?

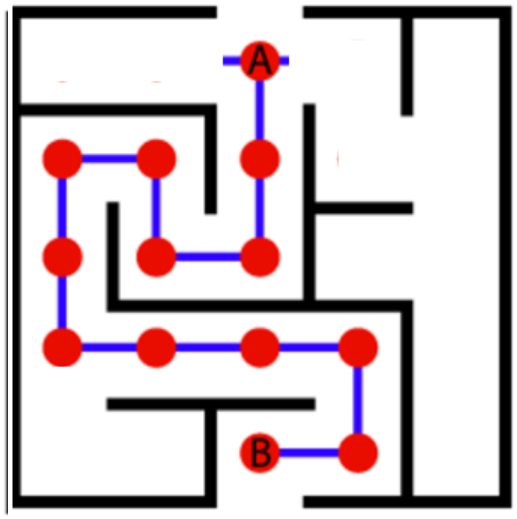


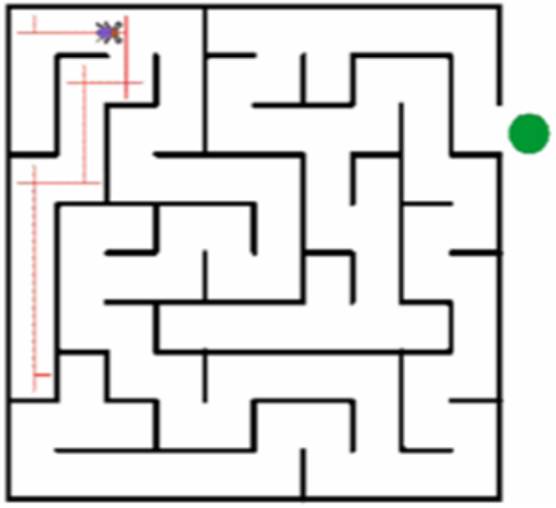


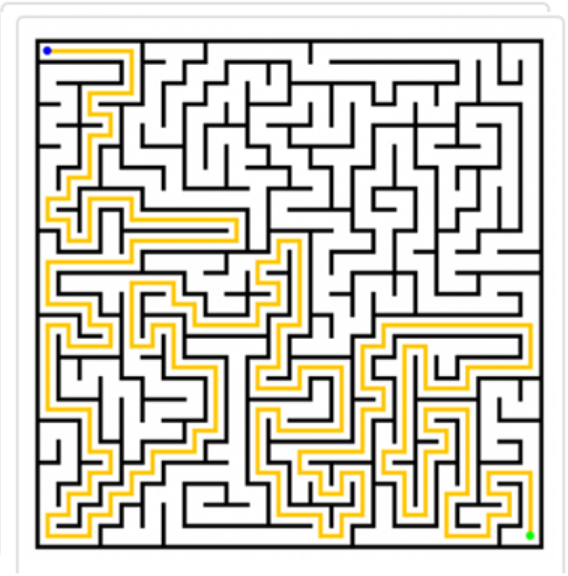


NUS National University of Singapore

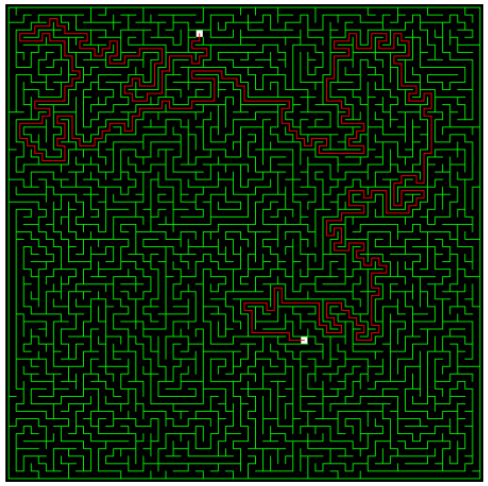
Could you find the shortest path?



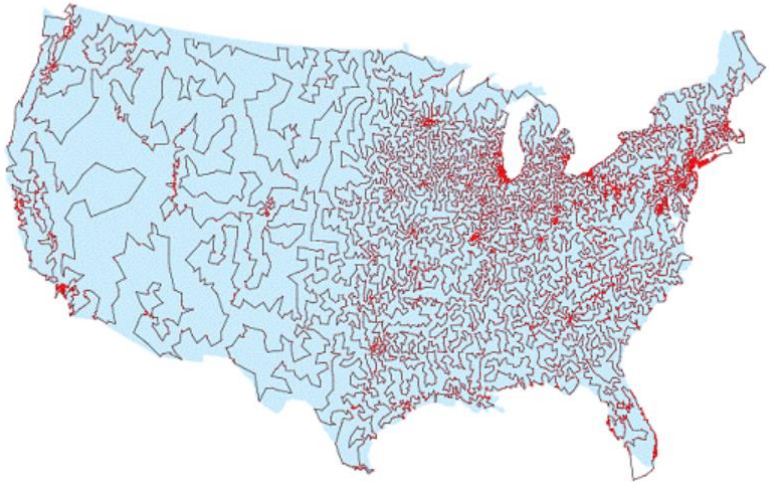




NUS National University of Singapore



Traveling Salesman Problem for Record-Breaking 13,509 Cities



Motion planning is challenging!
We have to develop a search algorithm.

NUS National University of Singapore

What is a search algorithm?

- **A search algorithm is the step-by-step procedure used to locate specific data among a collection of data.**

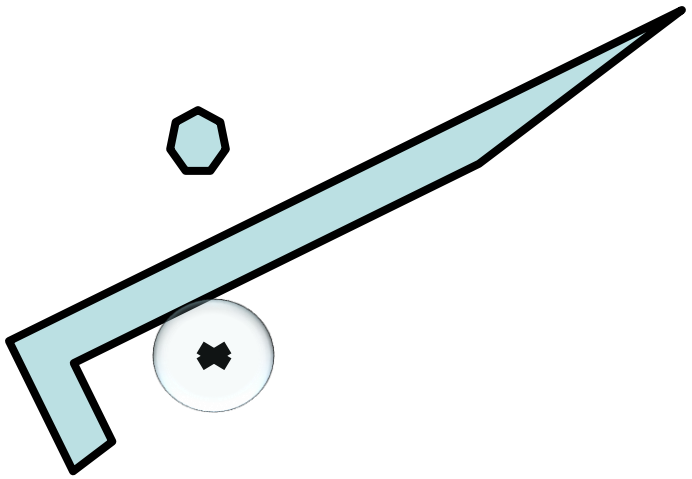
Mission: Moving from one place to another is a task that we humans do almost every day.

We try to find the shortest path that enables us to reach our destinations faster and make the whole process of travelling as efficient as possible. In the old days, we use trial and error method.

NUS National University of Singapore

Why we need to learn search algorithm?

Global solution (global planner)



NUS National University of Singapore

Natural way

Find the shortest path from home to school in the following example 1:

- **Home →C =5**
- **Home→B=2**
- **Home→A=3**

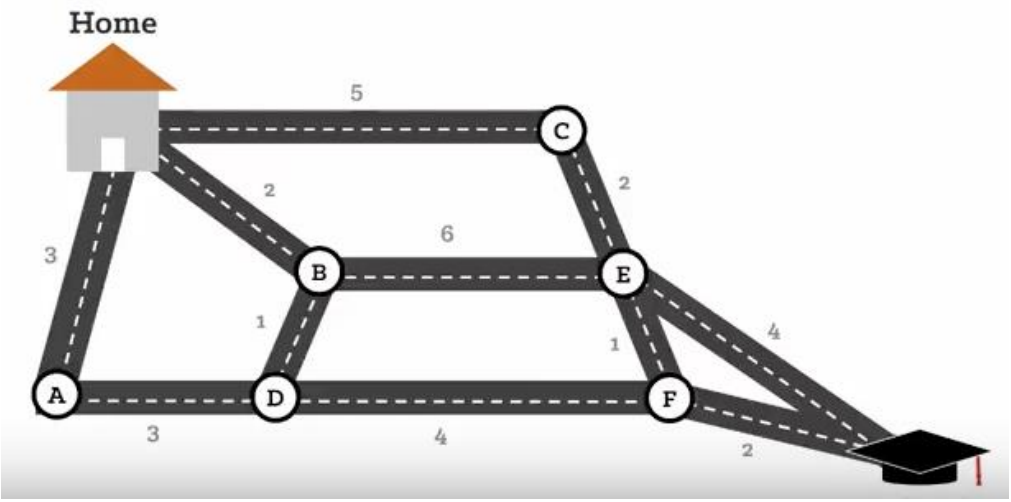
We select Home→B.

- **B→E=6**
- **B→D=1**

We select B→D.

D→F =4 then we select F→School.

Home→B→D→F→School=2+1+4+2=9

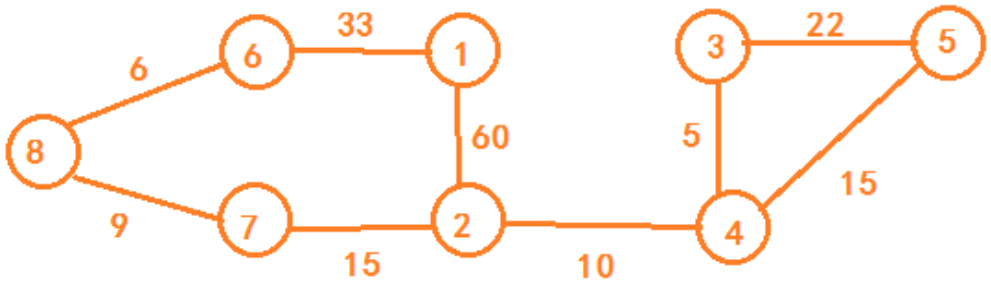


https://brilliant.org/wiki/dijkstras-short-path-finder/

NUS National University of Singapore

Example 2:

We have eight points (or 8 cities) :1,2,3,4,5,6,7,8. 1—starting point; 5—ending point



1—>2=60

1→6=33

We choose 1→6.

There is only one path remained.

We choose

6→8→7→2→4

Then

4→3

4→5

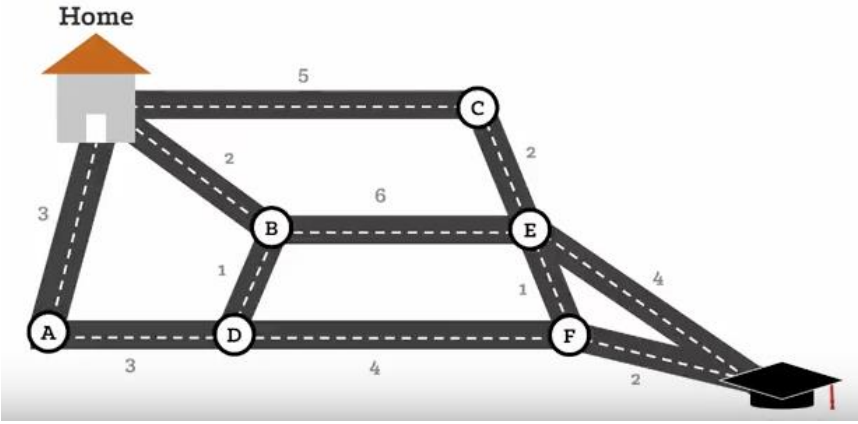
We choose 4→5. Finally, we have 1→6 →8→7→2→4→5= 33+6+9+15+10+15=88

It should be 1→2→4→5=60+10+15=85

NUS National University of Singapore

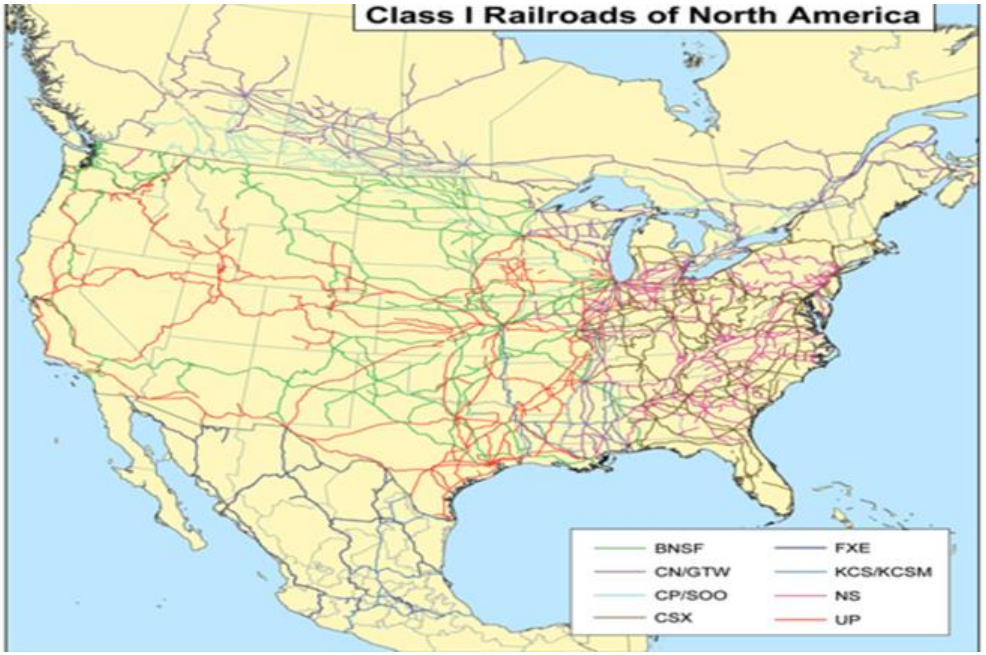
Exhausted search

Select all possible paths.



- Home-C-E-School =11
- Home-C-E-B-D-F-School=20
- Home-C-E-F-School=10
- Home-B-E-School=12
- Home-B-E-F-School=11
- Home-B-D-F-School=9
- Home-B-D-E-School=10
- Home-A-D-F-School=12
- Home-A-D-B-E-School=17
- Home-A-D-B-E-F-School=16

Other possibilities?



Edsger W. Dijkstra

Dijkstra algorithm (1959)-1

*S (visited set); U (unvisited set). **Each step considers neighboring nodes of unvisited set until U is empty.***
*See **example 3**.*

A—starting point;F—ending point

Step1 :S={A},U={B,C,D,E,F}

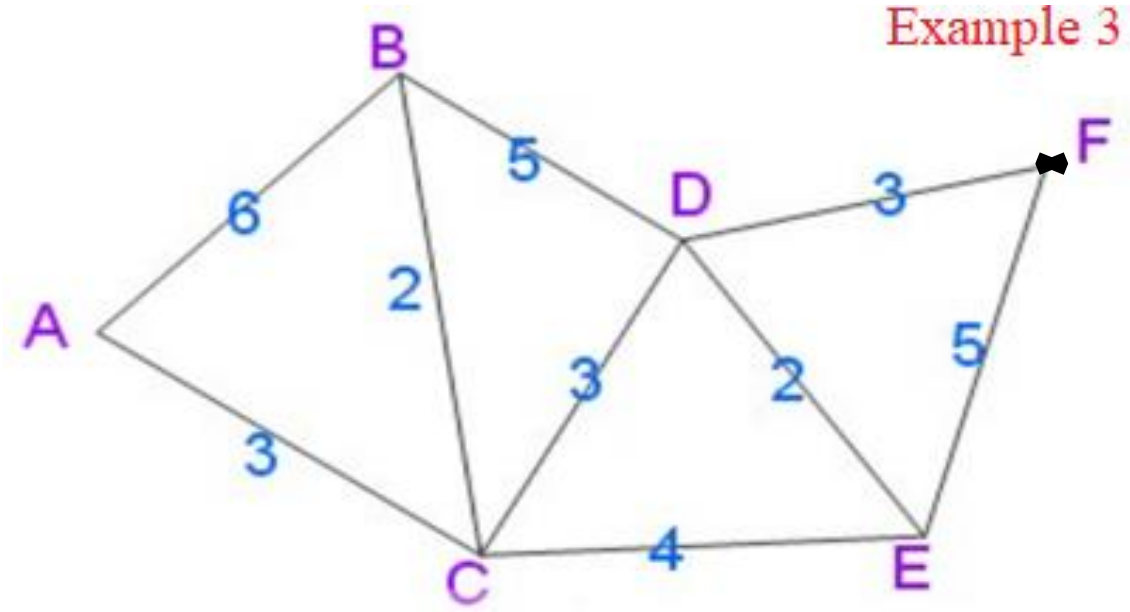
Check the neighboring nodes

- **A->B=6 (A is current node)**
- **A->C=3, stored**

Anything else? No.

We select minimum distance

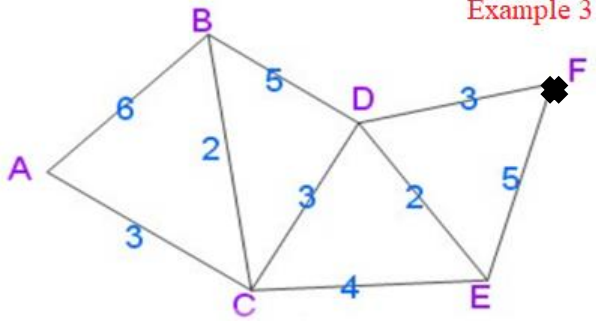
A->C=3



Dijkstra algorithm (1959)-2

S (visited set); U (unvisited set). Each step considers neighboring nodes of unvisited set until U is empty. See example 3.

Example 3



A—starting point;F—ending point

Step2 :S={A,C},U={B,D,E,F}

Check neighboring nodes

- **A->C->B=3+2=5 (current node)**
- **A->C->D=3+3=6 stored**
- **A->C->E=3+4=7, stored.**

Anything else? Wait …..

Note that we need to compare

A->C->B with A->B=6

Anything else? No

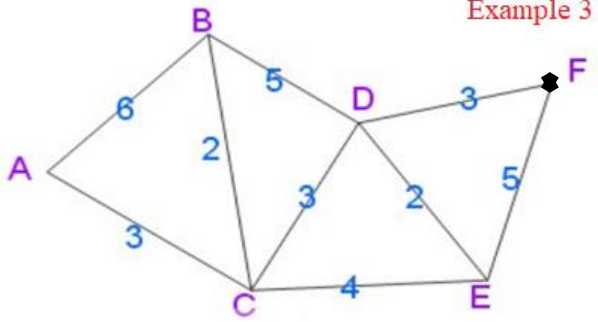
We select minimum A->C->B=5

NUS National University of Singapore

Dijkstra algorithm (1959)-3

S (visited set); U (unvisited set). Each step considers neighboring nodes of unvisited set until U is empty. See example 3.

Example 3



A—starting point;F—ending point

Step3 :S={A,C,B},U={D,E,F}

Check the neighboring nodes

A->C->B->D=3+2+5=10

But A->C->D=6,

Anything else? No

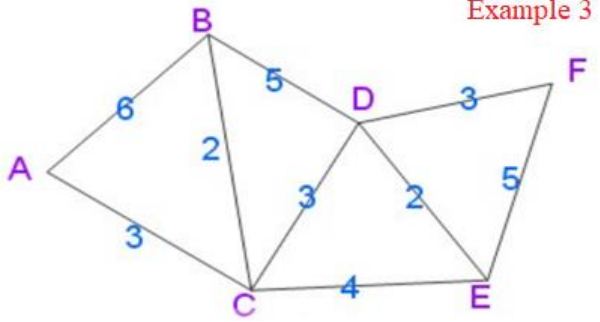
So, we choose A->C->D.

NUS National University of Singapore

Dijkstra algorithm (1959)-4

*S (visited set); U (unvisited set). Each step considers neighboring nodes of unvisited set until U is empty. See **example 3**.*

Example 3



A—starting point;F—ending point

Step4 :S={A,C,B,D},U={E,F}

- **A->C->D->E=3+3+2=8**

  A->C->D->F=3+3+3=9(can we choose this?)

But A->C->E=7,

Anything else? No

We choose A->C->E

(E will become current node)

Dijkstra algorithm (1959)-5

S (visited set); U (unvisited set). Each step considers neighboring nodes of unvisited set until U is empty. See example 3.

Step5:S={A,C,B,D,E},U={F}

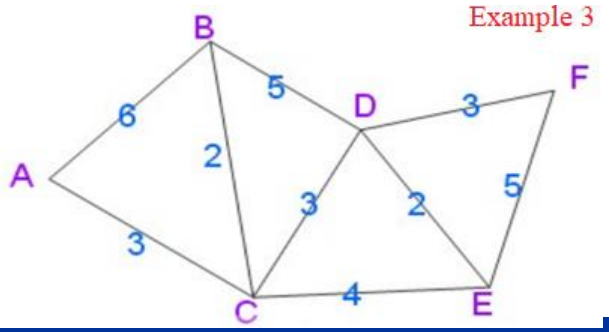
(we have only one node F)

A->C->E->F=3+4+5=12;
But A->C->D->F=3+3+3=9
Anything else? No. Finally, we choose
A->C->D->F

Step6 :S={A,C,B,D,E,F},U={}

(U is empty)

Finally, we choose
A->C->D->F



A—starting point;F—ending point

NUS National University of Singapore

Summary

- Check S and U;
- Check neighboring nodes and weights; calculate the cost;
- When reaching one point, please compare with the stored value before;
- At each step, choose the smallest cost;
- *until U is empty.*

NUS National University of Singapore

Recall example 2-1 Dijkstra alg

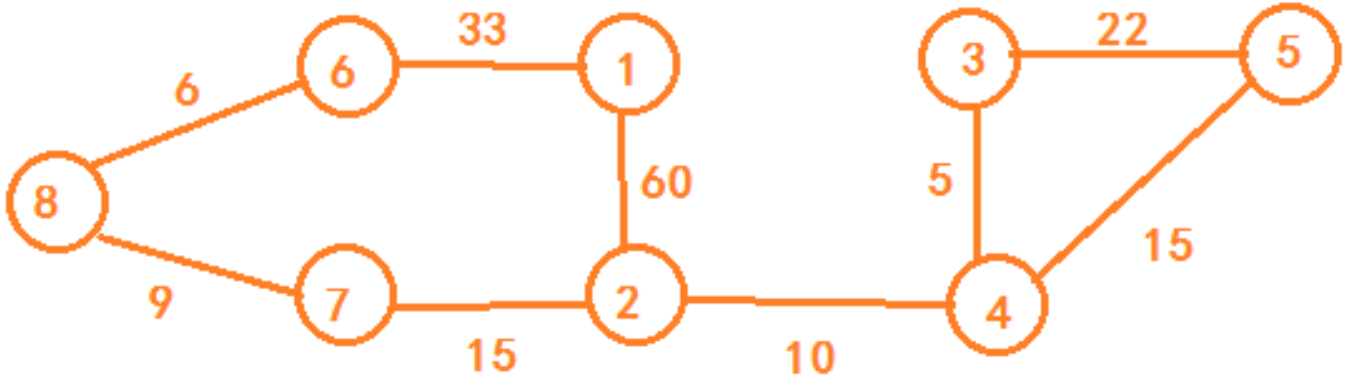
S (visited set); U (unvisited set). Each step considers neighboring nodes of unvisited set until U is empty.

Step 1:S={1} ; U={2,3,4,5,6,7,8}.

Starting point -- 1 ; Ending point -- 5.

- **1->2 =60**
- **1->6 =33**

Select 1->6



Recall example 2-2

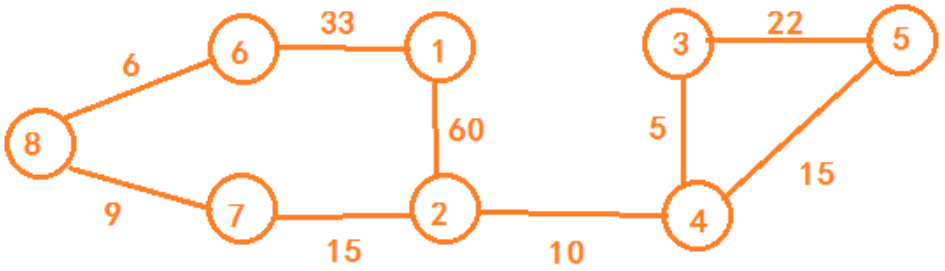
S (visited set); U (unvisited set). Each step considers neighboring nodes of unvisited set until U is empty.

Step 2:S={1,6} ; U={2,3,4,5,7,8}.

- **1->6->8 =33+6=39**

Anything else? No

Select 1->6->8



Starting point -- 1 ; Ending point -- 5

Step 3:S={1,6,8} ; U={2,3,4,5,7}.

- **1->6->8->7 =33+6+9=48**

Anything else? No

Select 1->6->8->7

NUS National University of Singapore

Recall example 2-3

S (visited set); U (unvisited set). Each step considers neighboring nodes of unvisited set until U is empty.

Step 4:S={1,6,8,7} ; U={2,3,4,5}.

- **1->6->8->7->2 =33+6+9+15=63**

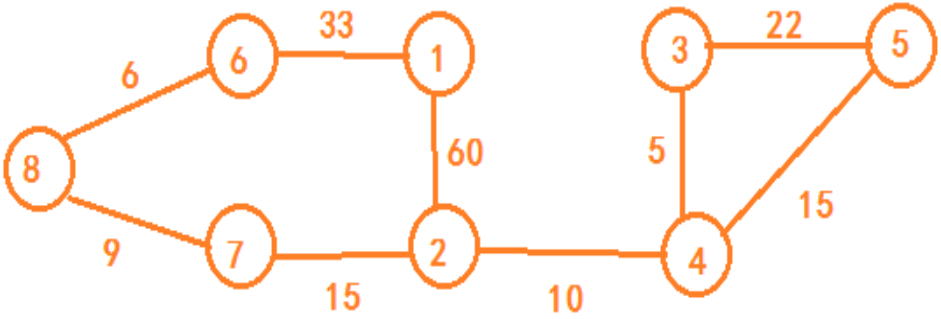
Anything else?

Yes, we have

1->2=60

Compare with this

Select 1->2



Starting point -- 1 ; Ending point -- 5.

NUS National University of Singapore

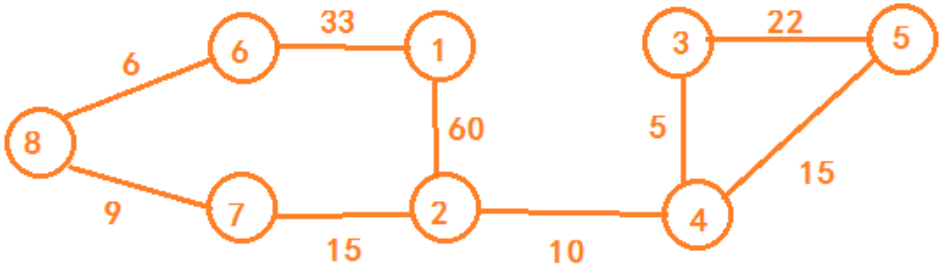
Recall example 2-4

S (visited set); U (unvisited set). Each step considers neighboring nodes of unvisited set until U is empty.

Step 5:S={1,6,8,7,2} ; U={3,4,5}.

- **1->2->4 =60+10**

Select 1->2->4



Step 6:S={1,6,8,7,2,4} ; U={3,5}.

- **1->2->4 ->5=60+10+15=85**
- **1->2->4->3=60+10+5=75**

Select 1->2->4->3

Starting point -- 1 ; Ending point -- 5.

NUS National University of Singapore

Recall example 2-5

S (visited set); U (unvisited set). Each step considers neighboring nodes of unvisited set until U is empty.

Step 7:S={1,6,8,7,2,4,3} ; U={5}.

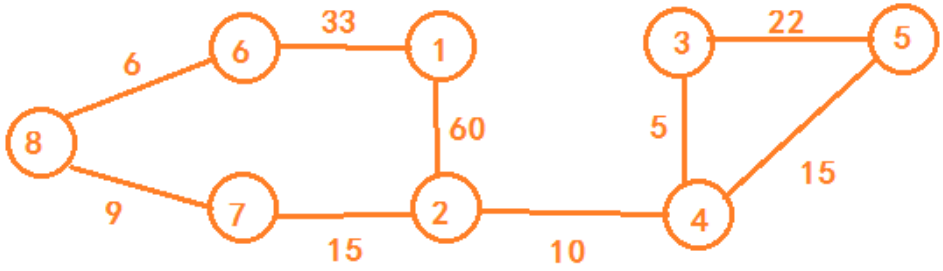
1->2->4->3>5=60+10+5+22=97

Starting point -- 1 ; Ending point -- 5.

But

1->2->4 ->5=60+10+15=85

Compare both



Select 1->2->4->5

NUS National University of Singapore

Recall example 2-6

S (visited set); U (unvisited set). Each step considers neighboring nodes of unvisited set until U is empty.

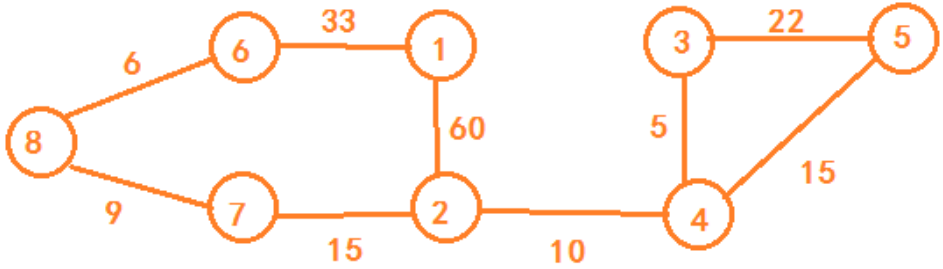
Step 7:S={1,6,8,7,2,4,3} ; U={5}.

- **1->2->4->3->5=60+10+5+22=97**

But 1->2->4 ->5=60+10+15=85

Starting point -- 1 ; Ending point -- 5.

Select 1->2->4->5



Step 8:S={1,6,8,7,2,4,3,5} ; U={}.

Finally, 1->2->4->5.

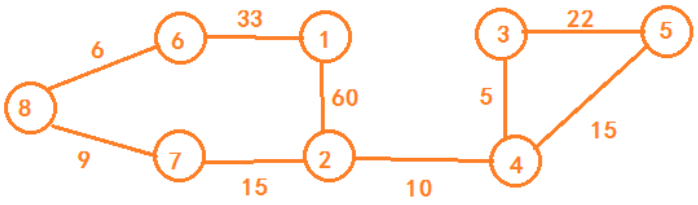
Total distance=85

NUS National University of Singapore

Problem:

Previous algorithms consider the distance from starting node to current node. It is found that the algorithms are not efficient. It expands in all directions.

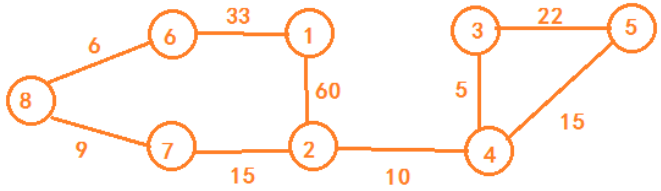
But this algorithm is especially useful when you have multiple target nodes but you don't know which one is the closest.



NUS National University of Singapore

However, a common case is to find a path to only one location. Let's make the expansion towards the goal more than it expands in other directions.

This motivates the AI-based search algorithms

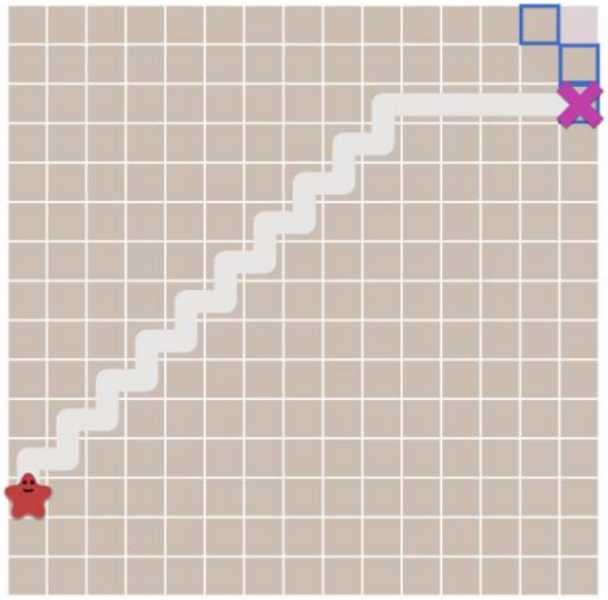


NUS National University of Singapore

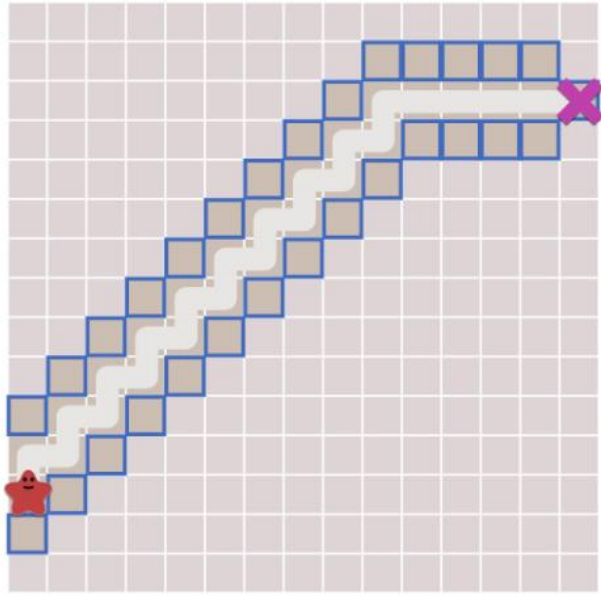
First, we'll define a *heuristic* function that tells us how close we are to the goal.

In Dijkstra's Algorithm we used the actual distance from the *start* for the priority queue ordering. Here instead, we'll use the estimated distance to the *goal* for the priority queue ordering. This is called the heuristic function. The location closest to the goal will be explored first.

Dijkstra algorithm



Heuristic function based algorithm

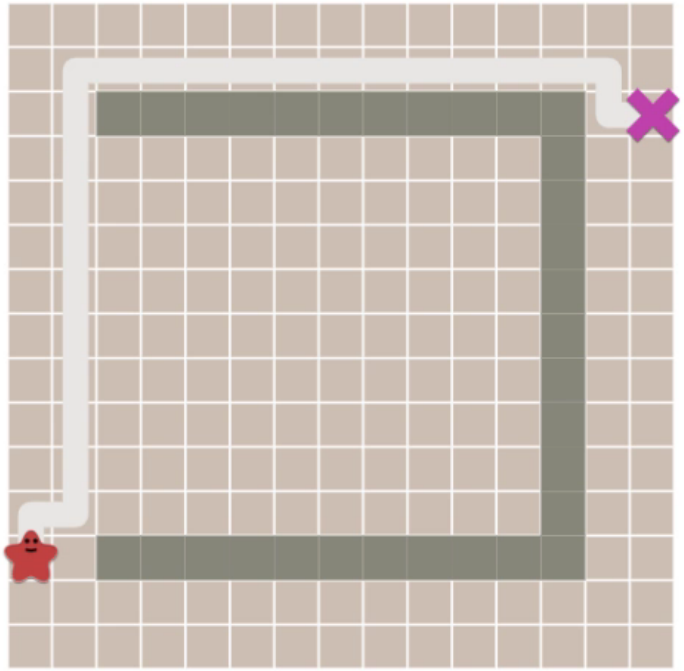


https://www.redblobgames.com/pathfinding/a-star/introduction.html

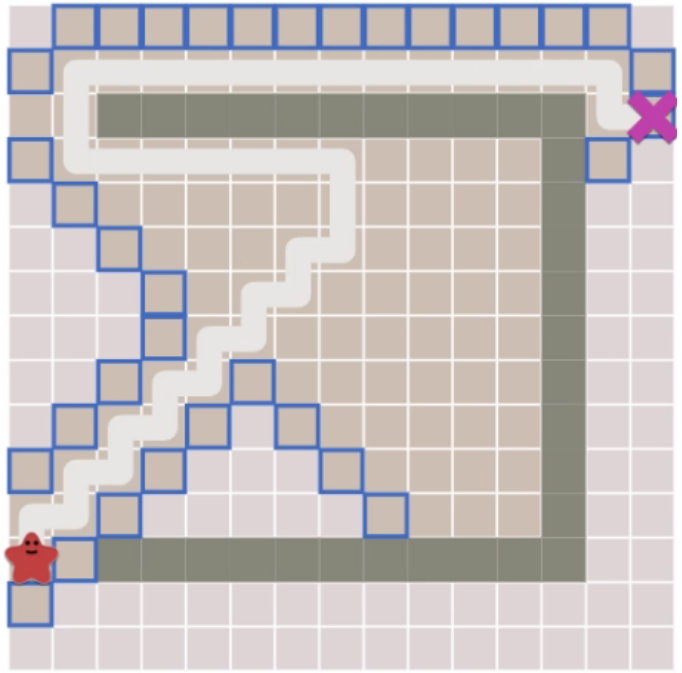
NUS National University of Singapore

But what happens in a more complex map?

Dijkstra algorithm



Heuristic function based algorithm

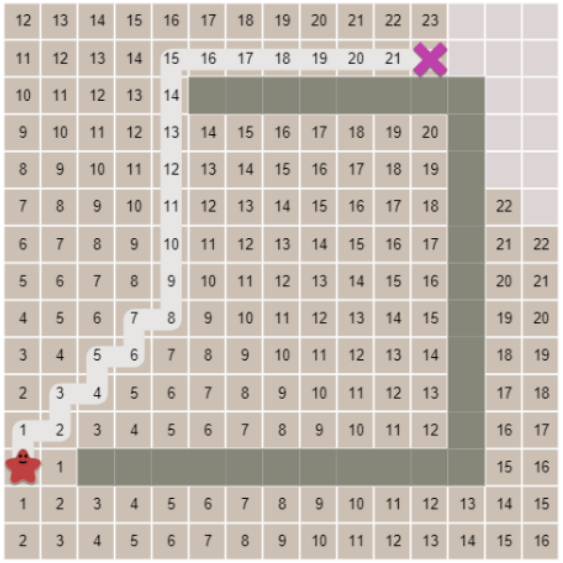


NUS National University of Singapore

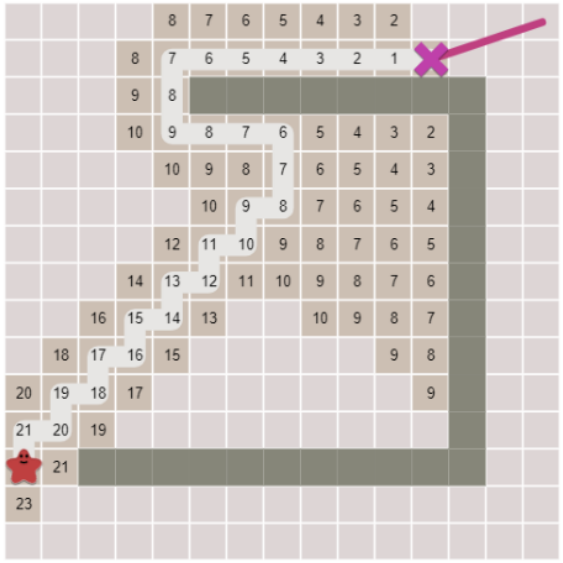
Can we combine both algorithms working together for searching?

Yes! The A* algorithm uses *both* the actual distance from the start and the estimated distance to the goal, i.e., accumulated path cost + heuristic

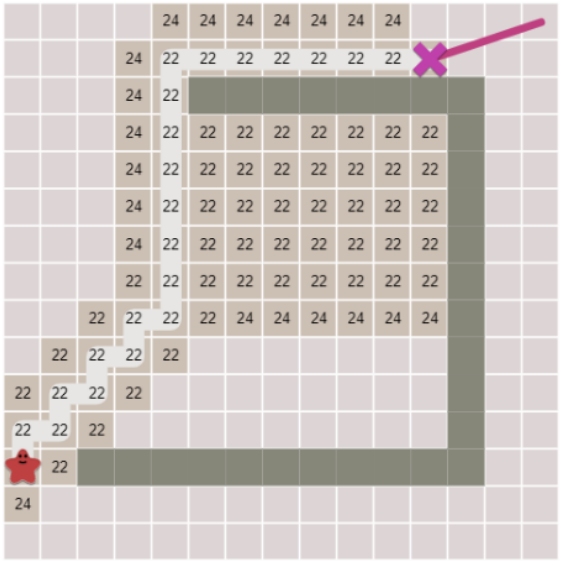
Dijkstra's Algorithm



Heuristic function based algorithm



A* Search



AI-based search algorithm----A* algorithm

What is A* Search Algorithm?

It is an advanced algorithm that searches for the shortest path between the starting node and the ending node.

A* achieves *optimality* and *completeness*, two valuable property of search algorithms.
When a search algorithm has the property of optimality, it means it is guaranteed to find the best possible solution.

When a search algorithm has the property of completeness, it means that if a solution to a given problem exists, the algorithm is guaranteed to find it.

https://www.gamedev.net/reference/articles/article2003.asp

NUS National University of Singapore

Why A* Search Algorithm ?

A* Search algorithms, unlike other techniques, it has "brains" since it introduce "heuristic". What it means is that it is really a smart algorithm which separates it from the other conventional algorithms.

It is worth mentioning that many games and web-based maps use this algorithm to find the shortest path very efficiently (approximation)

Essentially, a heuristic function helps algorithms to make the best decision faster and more efficiently.

https://www.geeksforgeeks.org/a-search-algorithm/

AI-based search algorithm----A* algorithm

The first thing you should notice is that our search area is divided into a square grid. We have the starting point A and ending point B.

- **Starting the Search**
- **Path Sorting based on the function (F=G+H)**
- **Continuing the Search**

A* search

Idea:

--avoid expanding paths that area already expensive

-- focus on paths that show promise

https://www.edureka.co/blog/a-search-algorithm/

- There is one formula that all of you need to remember as it is the heart and soul of the algorithm.
- F = G + H
- F is the cost function of A* which is the sum of the other functions G and H. This parameter is responsible for helping us find the most optimal path.
- G is the cost of moving from the starting point to one node in a given square on the grid, following the path generated to get there. This function changes for every node as we move up to find the most optimal path.
- H is the heuristic/estimated path between the current code to the destination node. This cost is not actual but is, in reality, a guess cost that we use to find which could be the most optimal path between our source and destination

NUS National University of Singapore

Heuristics

We can calculate *g* but how to calculate *h* ?

We can do :

a) exact heuristics--- calculate the exact value of *h* (time consuming), or

b) approximation heuristics--- approximate the value of *h* using some heuristics (less time consuming).

Heuristics

Exact Heuristics – If there are no blocked cells/obstacles then we can just find the exact value of *h*

NUS National University of Singapore

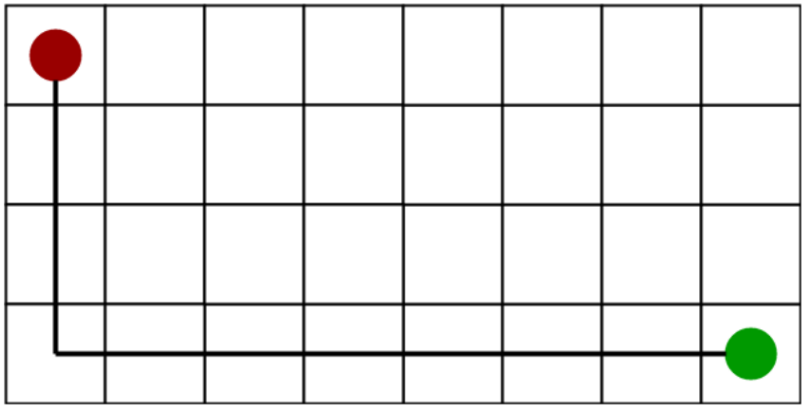
Approximation Heuristics

There are generally three approximation heuristics to calculate h. These are only for reference. It is not absolute rules.

1) Manhattan Distance –

h = abs (current_cell.x – goal.x) + abs (current_cell.y – goal.y)

When to use this heuristic? – When we are allowed to move only in four directions only (right, left, top, bottom).



NUS National University of Singapore

2) Diagonal Distance-

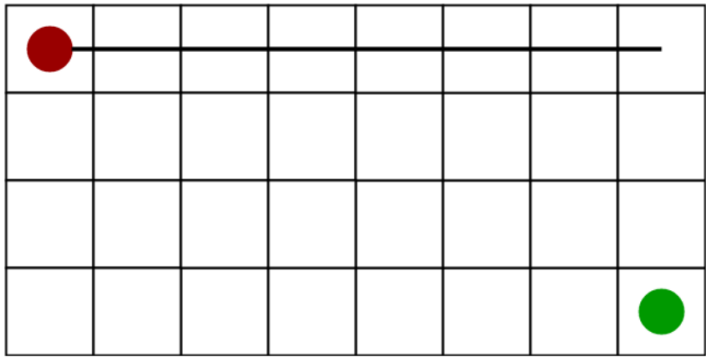
It is nothing but the maximum of absolute values of differences in the goal's x and y coordinates and the current cell's x and y coordinates respectively,

dx = abs(node.x - goal.x) dy = abs(node.y - goal.y),

h=D * (dx + dy) + (D2 - 2 * D) * min(dx, dy), where D is the minimum cost for moving from one space to an adjacent space (D=1). When D2=1, h is the **Chebyshev distance**. When D = 1 and D2 = sqrt(2), this is called the ***octile distance***.

- **h = max { abs(current_cell.x – goal.x), abs(current_cell.y – goal.y) } (Chebyshev distance)**

When to use this heuristic? – When we are allowed to move in eight directions only (similar to a move of a King in Chess)



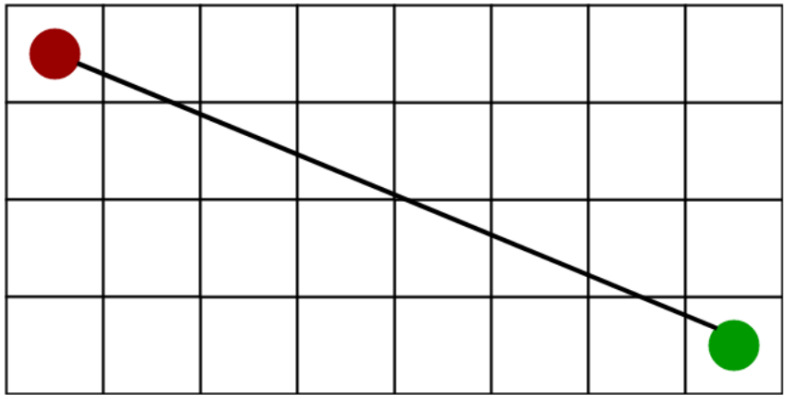
NUS National University of Singapore

3) Euclidean Distance-

As it is clear from its name, it is nothing but the distance between the current cell and the goal cell using the distance formula

$$h = \sqrt[2]{(current.x - goal.x)^2 + (current.y - goal.y)^2}$$

When to use this heuristic? – When we are allowed to move in any directions



Relation (Similarity and Differences) with other algorithms-

Dijkstra is a special case of A* Search Algorithm, where h = 0 for all nodes.

NUS National University of Singapore

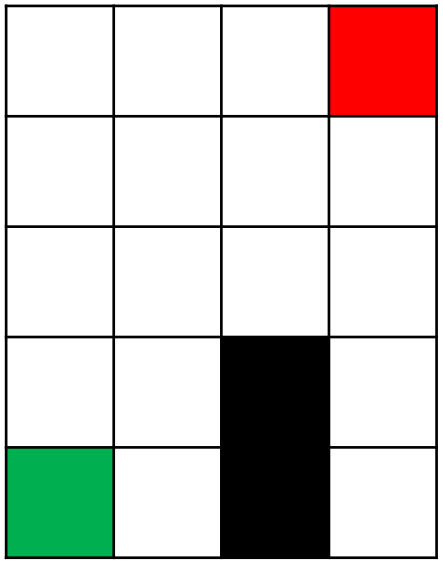
Example:

We can consider a 2D Grid having several obstacles. We start from a starting square (coloured green below) to reach towards a goal square (coloured red below). We consider A* to move only in four directions only (right, left, top, bottom). Thus, Manhattan Distance is defined as a heuristics.

We proceed to the starting square which is called the current cell. Then we proceed to look at all its neighbors and compute

$$f = g + h$$

where g is cost to travel to current node and h is the heuristic value.



Select the neighbor with the lowest f cost. This is our new current cell

NUS National University of Singapore

Rule (priority rank) : Right-> up-> left-> down

Use Manhattan distance (4 directions)

Open set, Close set

Initially, Open={}, Close={}

Step1:Open={(5,1)} Current cell=(5,1)

Put current cell into the close set

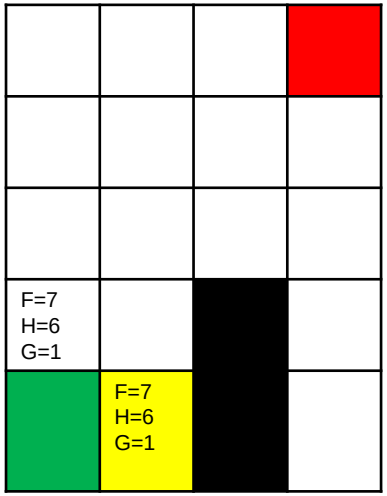
Close={(5,1)}

Two directions:

- **Open={(5,2),(4,1)}, due to four directions**
- **Compute the cost F**

- **According to Rules, select (5,2) and parent is (5,1)**

Remember the parent !!



National University of Singapore

Rule: Right-> up-> left-> down

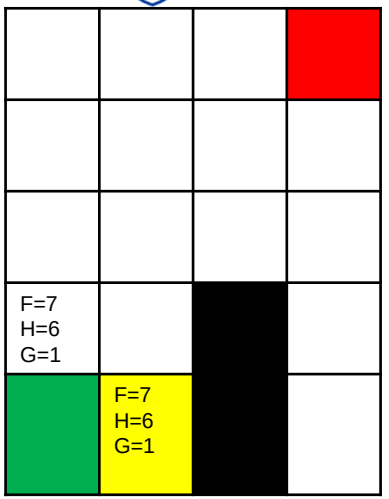
Use Manhattan distance (4 directions)

Open set, Close set

Initially, Open={}, Close={}

Step1:Open={(5,1)} Current cell=(5,1)

- **Open={(5,2),(4,1)},Close={(5,1)}**
- **According to Rules, select (5,2) and parent is (5,1)**



Step2: Open={(5,2),(4,1)}, current cell=(5,2),

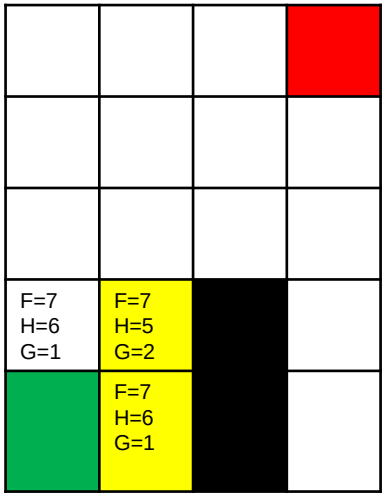
Put (5,2) into the close

- **Close={(5,1),(5,2),(5,3)} ,**

One direction, why?

- **Open={(4,1),(4,2)}**
- **Compute the cost F**

According to Rule, select (4,2) and parent is (5,2)



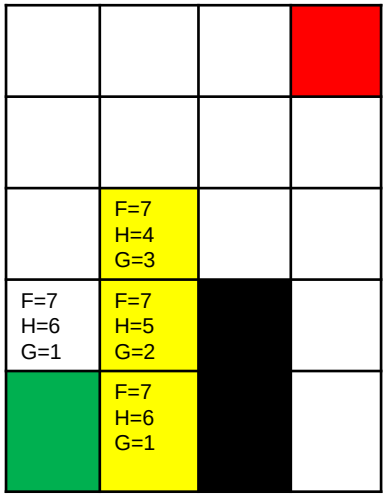
NUS National University of Singapore

Rule: Right-> up-> left-> down

Step3: Open={(4,2),(4,1)}, current cell=(4,2), put (4,2) into the close
Close={(5,1),(5,3),(5,2),(4,3),(4,2)},

Two directions,why?
Open={(4,1),(3,2)}
Compute the cost
Update left cell F=? G=3,H=6,F=9. No

According to Rule, select (3,2) and parent is (4,2)



NUS National University of Singapore

Rule: Right-> up-> left-> down

Step4: Open={(4,1),(3,2)}, current cell=(3,2)

Put (3,2) into the close set

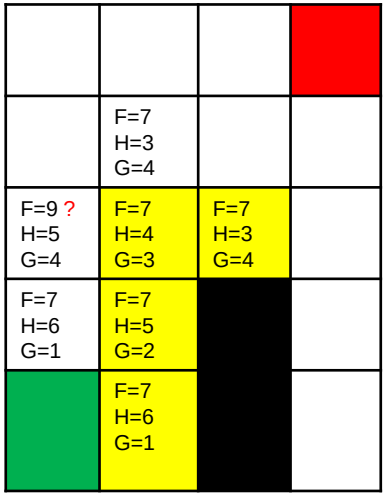
Close={(5,1),(5,3),(5,2),(4,3),(4,2),(3,2)},

Three directions, why?

Open={(4,1),(3,1),(3,3),(2,2)}

Compute the cost

According to Rules, select (3,3) and parent (3,2)



NUS National University of Singapore

Rules: Right-> up-> left-> down

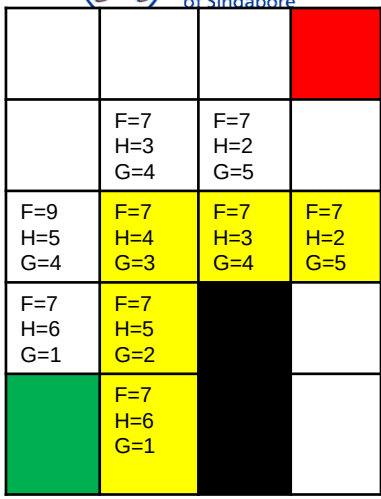
Step5: Open={(4,1),(3,1),(3,3),(2,2)} current cell=(3,3)

Put (3,3) into the close set

- **Close={(5,1),(5,3),(5,2),(4,3),(4,2),(3,2),(3,3) },**

Two directions,why?

- **Open={(4,1),(3,1),(2,2),(3,4),(2,3)}**
- **Compute the cost**
- **According to Rules, select (3,4) and parent (3,3)**



NUS National University of Singapore

Rules: Right-> up-> left-> down

Step6:Open={(4,1),(3,1),(2,2),(3,4),(2,3)} current cell=(3,4)

Put (3,4) into the close set

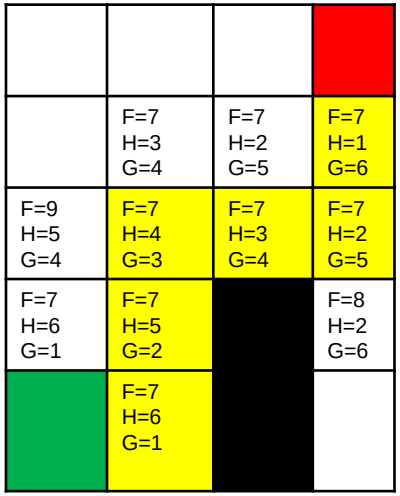
Close={(5,1),(5,3),(5,2),(4,3),(4,2),(3,2),(3,3),(3,4)},

Two directions, why?

Open={(4,1),(3,1),(2,2),(2,3),(2,4),(4,4)}

Compute the cost

According to Rule, select (2,4) and parent (3,4).



Step 7:Open={(4,1),(3,1),(2,2),(2,3),(2,4),(4,4)} current cell=(2,4)

Close= {(5,1),(5,3),(5,2),(4,3),(4,2),(3,2),(3,3),(3,4),(2,4)},

Two directions,why?

Open={(4,1),(3,1),(2,2),(2,3),(4,4),(1,4)}

It reaches target square (1,4) and parent (2,4).

Return to the starting point from the target point.

NUS National University of Singapore

Return to the starting point from the target point

	F=7 H=3 G=4	F=7 H=2 G=5	F=7 H=1 G=6
F=7 H=5 G=2	F=7 H=4 G=3	F=7 H=3 G=4	F=7 H=2 G=5
F=7 H=6 G=1	F=7 H=5 G=2		F=8 H=2 G=6
	F=7 H=6 G=1		

How do we determine the shortest path?

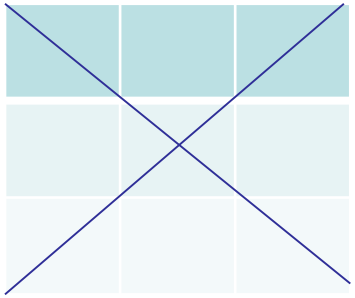
Just start at the red target square, and work backwards moving from one square to its parent, following the route. This will eventually take you back to the starting square, and that's your path.

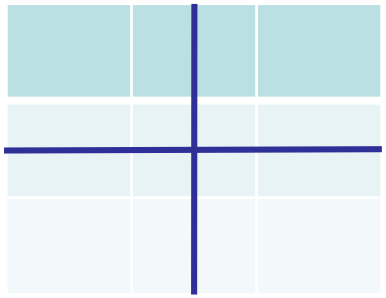
Now it is
(5,1)→(5,2)→(4,2)→(3,2)→(3,3)→(3,4)→

(2,4)→(1,4)

NUS National University of Singapore

If we change moving directions, what happens? Heuristics?





Left right up down=1
Diagonal line =1.4

Left right up down =1

NUS National University of Singapore

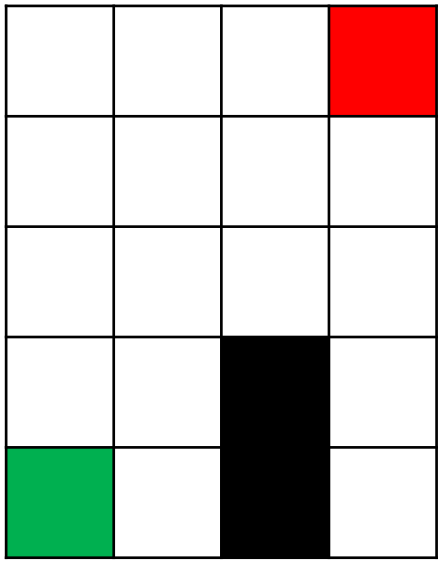
We consider A* to move in eight directions. Thus, Euclidean Distance is defined as a heuristics.

We proceed to the starting square which is called the current cell. Then we proceed to look at all its neighbors and compute

$$f = g + h$$

where g is cost to travel to current node and h is the heuristic value.

Select the neighbor with the lowest f cost. This is our new current cell

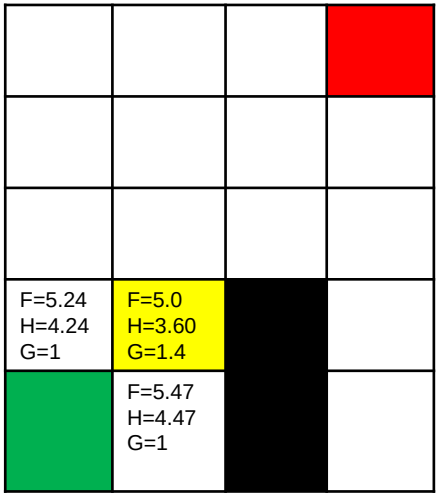


NUS National University of Singapore

Rules: Right-> up-> left-> down

Open set, Close set

Initially, Open={}, Close={}



Step1:Open={(5,1)} Current cell=(5,1)

Put (5,1) to the close set

Close={(5,1)}

Three directions , why?

- **Open={(5,2),(4,1),(4,2)},**

Compute the cost

- **According to Rules, select (4,2) and father is (5,1)**

NUS National University of Singapore

Rules: Right-> up-> left-> down

Open set, Close set

Initially, Open={}, Close={}

Step2: Open={(5,2),(4,1),(4,2)}, current cell=(4,2),

Put (4,2) into the close set

Close={(5,1),(5,3),(4,2),(4,3)},

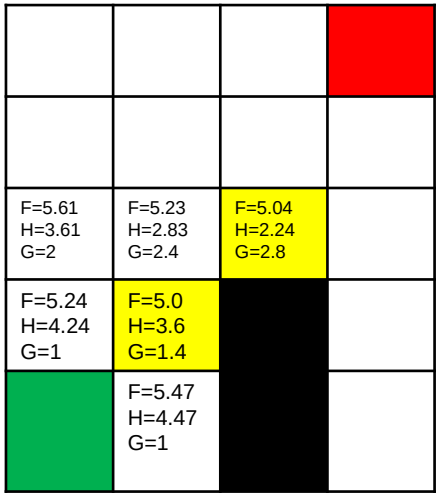
Five directions, why?

Open={(5,2),(4,1),(3,1),(3,2),(3,3)}

Compute the cost

- **Update (3,1)? No**

According to Rules, select (3,3) and father is (4,2)



NUS National University of Singapore

Rules: Right-> up-> left-> down

Step3: Open={(5,2),(4,1), (3,1),(3,2),(3,3)}, current cell=(3,3),

Put (3,3) into the close set

- **Close={(5,1),(5,3),(4,2),(4,3),(3,3)},**

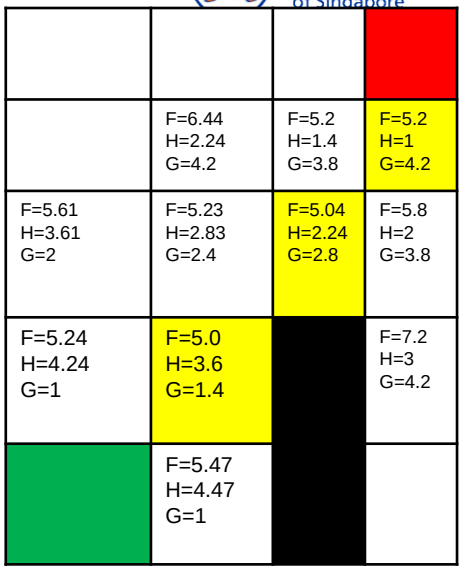
Six directions, why?

- **Open={(5,2),(4,1),(3,1),(3,2),(3,4),(4,4),(2,4),(2,3),(2,2)}**

Compute the cost

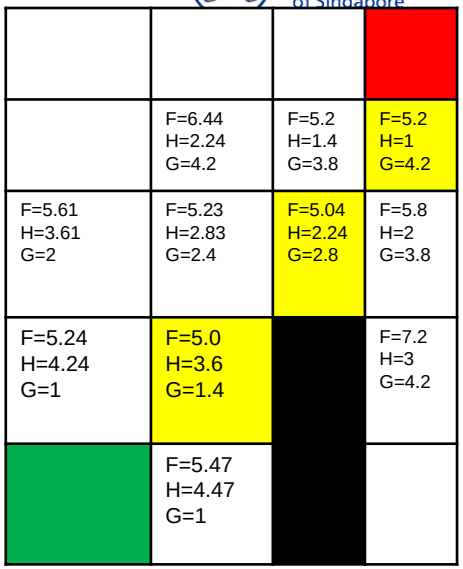
- **Update (3,2)? G=3.8+2.83=6.63, No**

- **According to Rules, select (2,4) and father is (3,3)**



NUS National University of Singapore

Rules: Right-> up-> left-> down



Step4:
Open={(5,2),(4,1),(3,1),(3,2),(3,4),(4,4),(2,4),(2,3),(2,2)},current cell=(2,4)

Put (2,4) into the close set

Close={(5,1),(5,3),(4,2),(4,3),(3,3),(2,4)},

Four directions,why?

Compute the cost

the shortest path is

(5,1)→(4,2)→(3,3)→(2,4)→(1,4)

Compare both methods

NUS National University of Singapore

We don't have a unique solution.

- solution depends on the rule
- solution depends on the heuristic function you chosen
- solution depends on the cell step size.

However, we have the shortest path under these conditions.

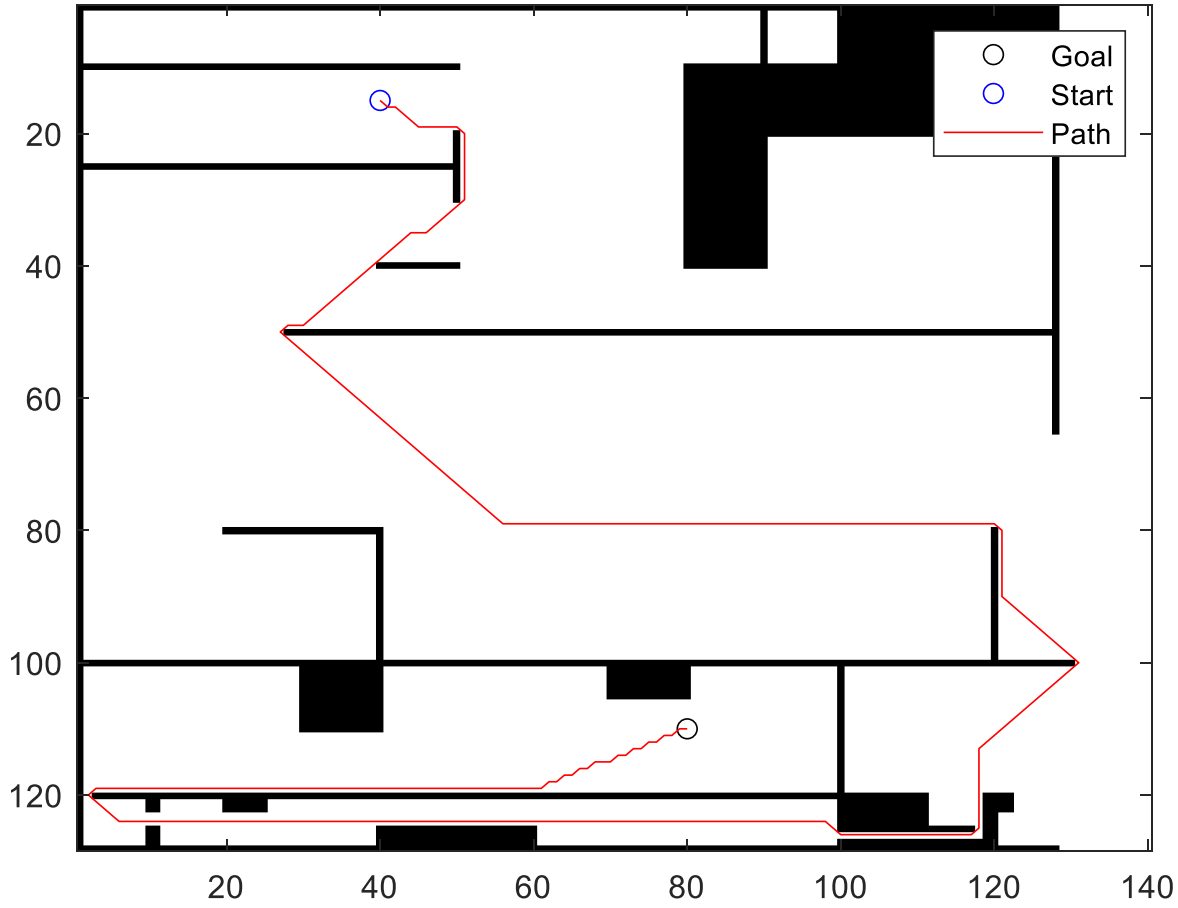
NUS National University of Singapore

How it works

- Node
- Cost
- Open list, and Closed list
- “open list", is a list of all locations immediately adjacent to areas that have already been explored and evaluated (the closed list).
- The closed list is a record of all locations which have been explored and evaluated by the algorithm.
- Rules

NUS National University of Singapore

MATLAB demo of A*



Summary of the A* algorithm

- Starting point is put in open list
- Repeat the following procedures

a. in open list, find the node which is the minimum F and record this node to the close list

b. check all the neighboring nodes

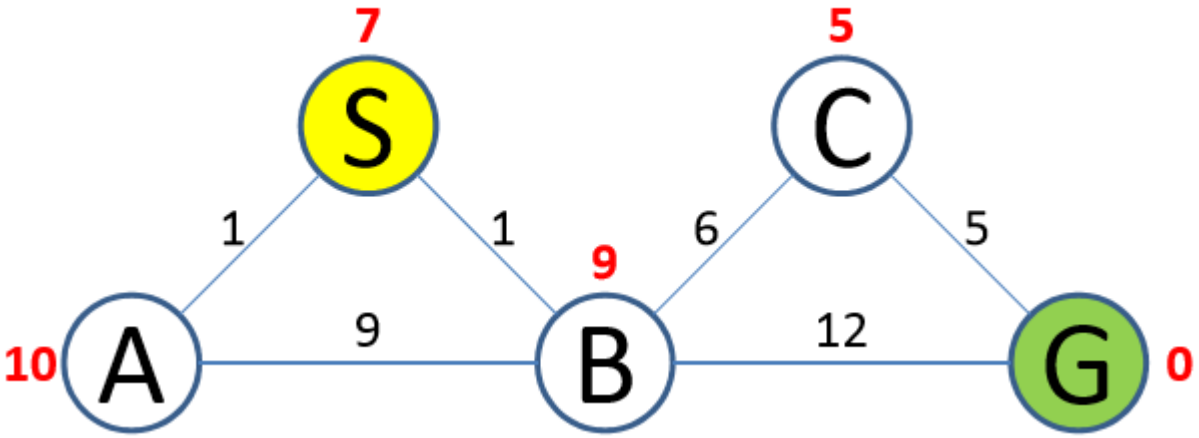
c. when the destination is put in the open list, the path is found; otherwise failure, the open list is empty; no path

d.From the destination to the starting point through father's node,you will find the path.

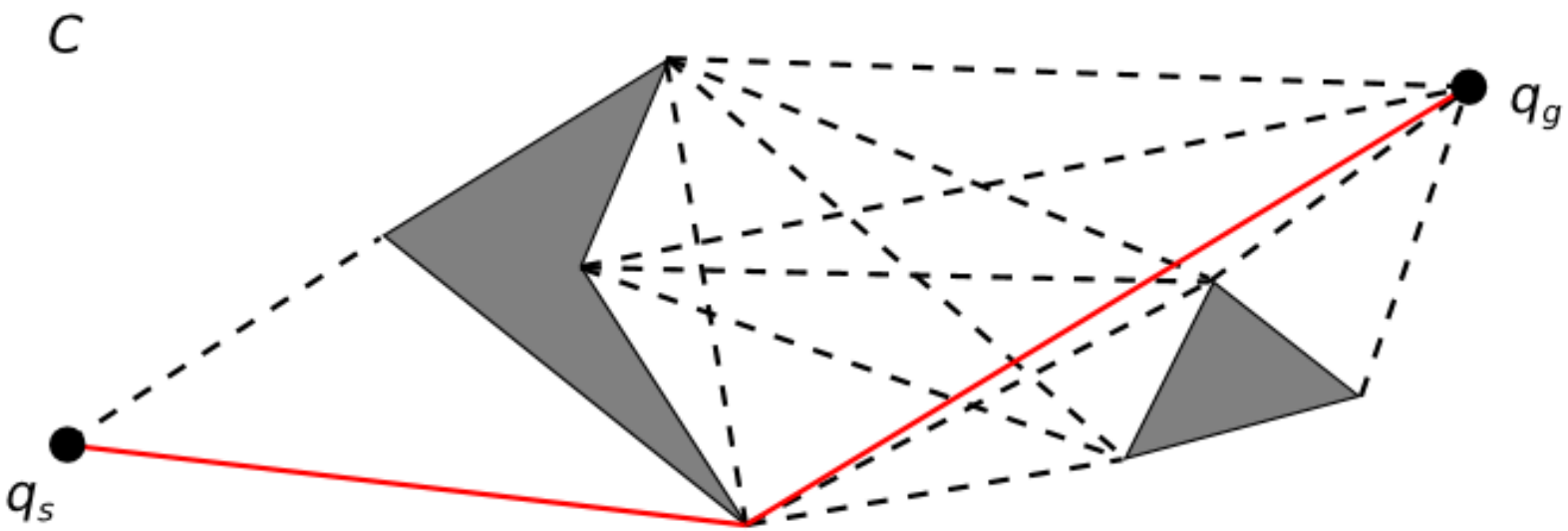
NUS National University of Singapore

Can we use A* to handle the routes?

- S-starting point; G-ending point
- Red- heuristic value ; Edge-distance

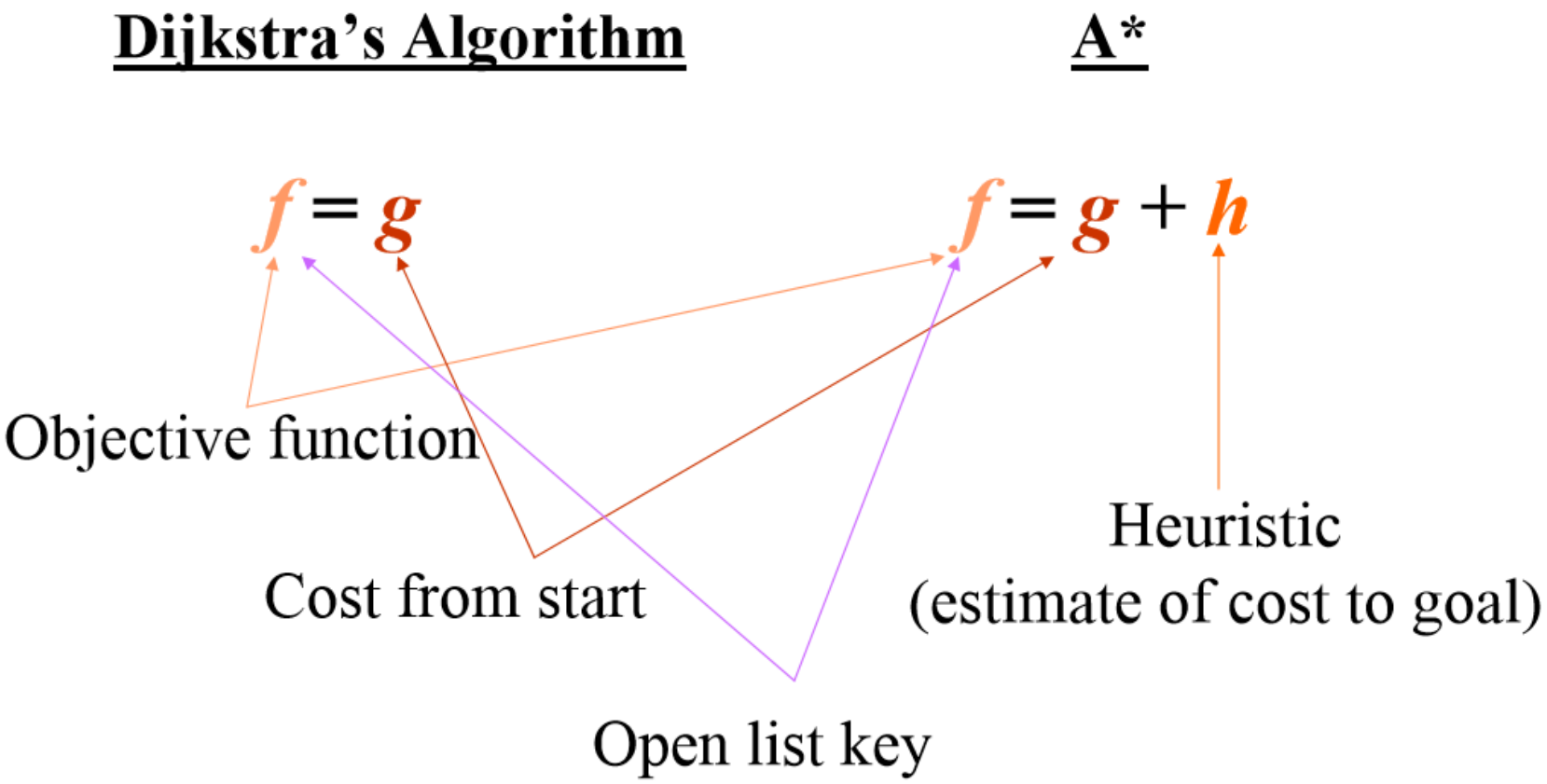


Extension of A*--- Visibility graph search



NUS National University of Singapore

Comparison between D and A*

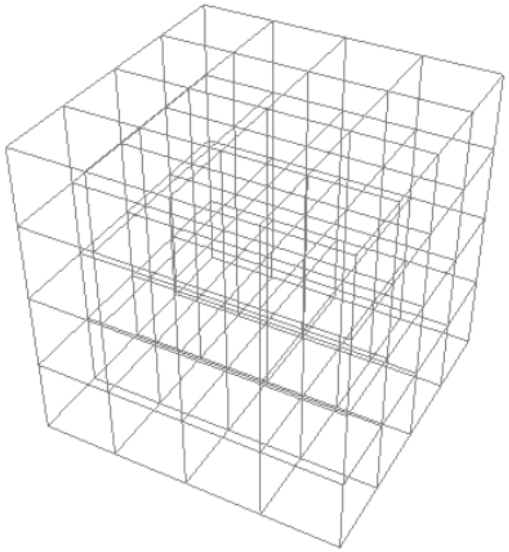


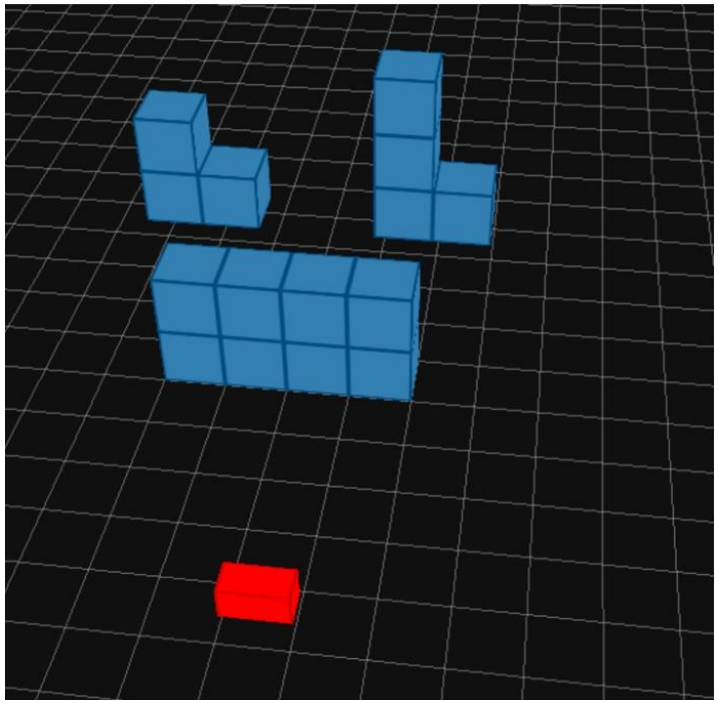
NUS National University of Singapore

How about 3D?

NP hard problem.

NP-problem (nondeterministic polynomial time) problem.
NP-hard therefore means "at least as hard as any NP-problem,“ although it might, in fact, be harder.





NUS National University of Singapore

Summary:

- Dijkstra algorithm
- *Heuristic* function
- A* search algorithm= Dijkstra alg+heuristic function

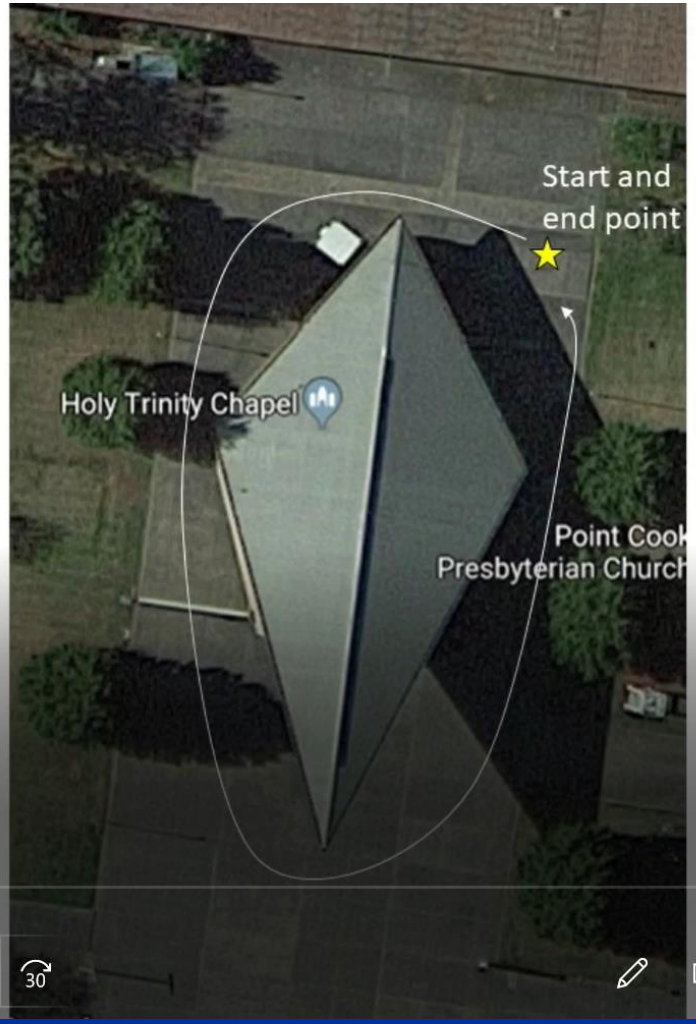
$$F=G+H$$

We have one problem:grid net and complex obstacles.

NUS National University of Singapore

One application of A*

GNSS-less autonomous navigation and obstacle avoidance around building in RAAF Williams, Point Cook Base



video2

:00:00

播放

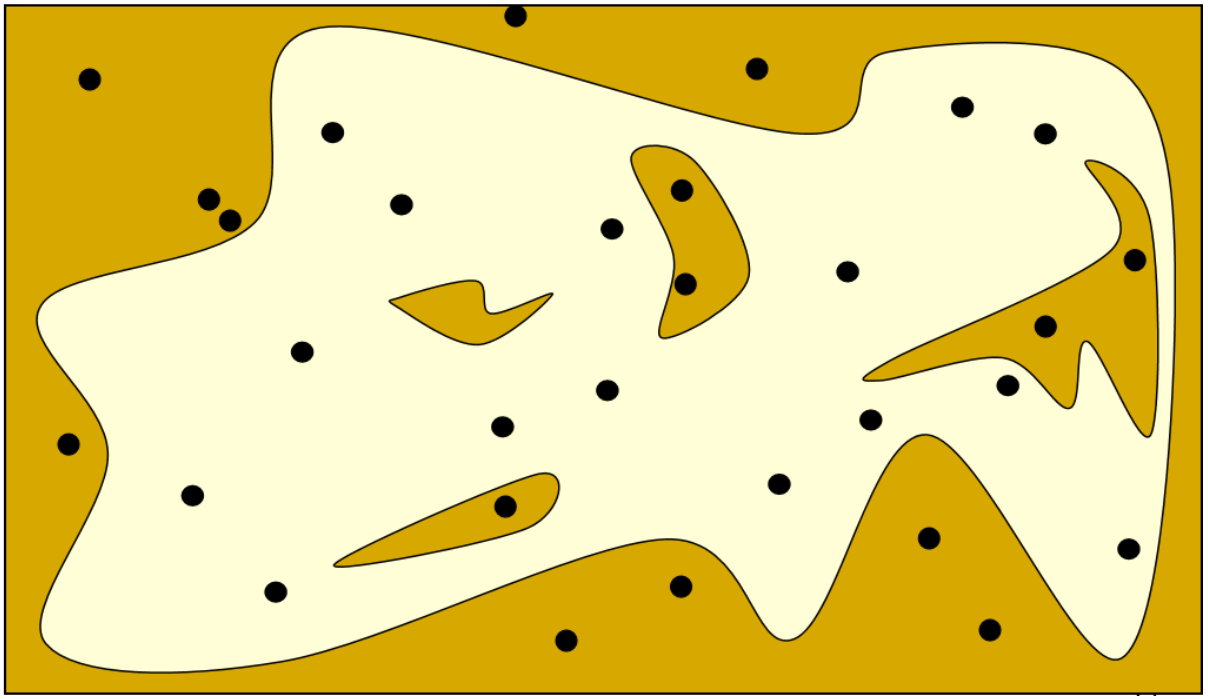
0:03:38

Lydia Kavraki

Sampling based search: Probabilistic Roadmap Method (PRM)

1.C-space or road map;

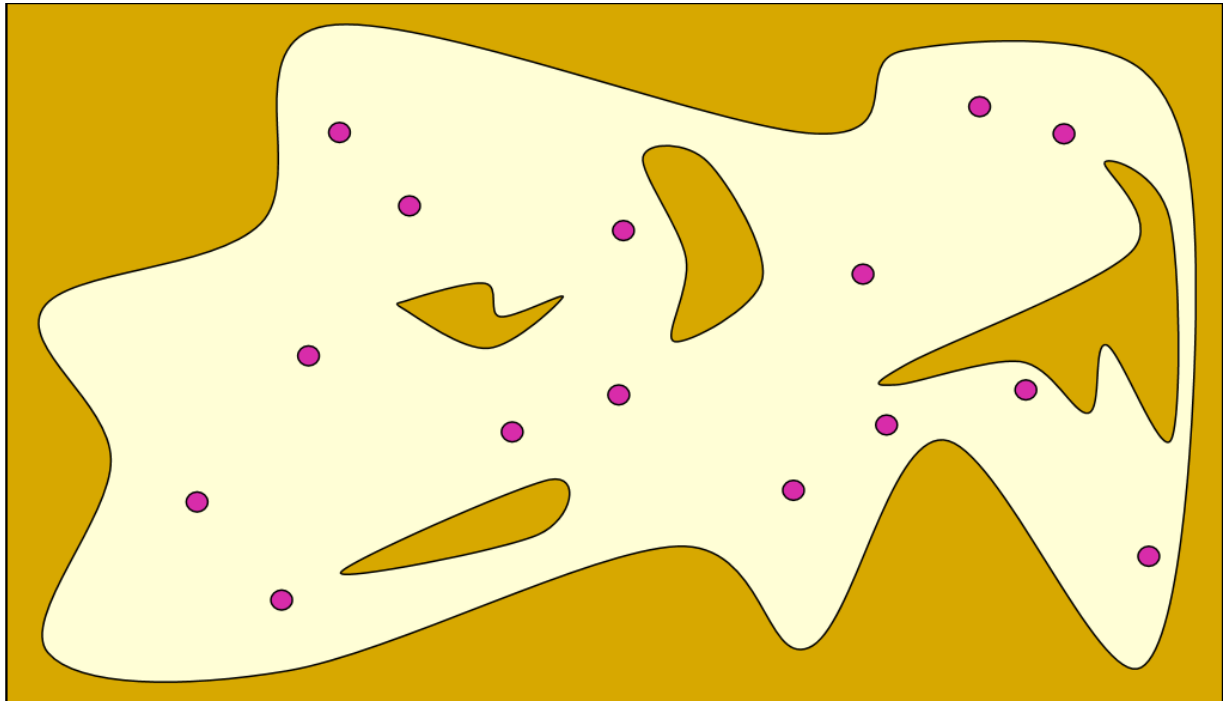
2.generate random sampling points (50,100,or …)



https://people.eecs.berkeley.edu/~pabbeel/cs287-fa12/slides/SamplingBasedMotionPlanning.pdf

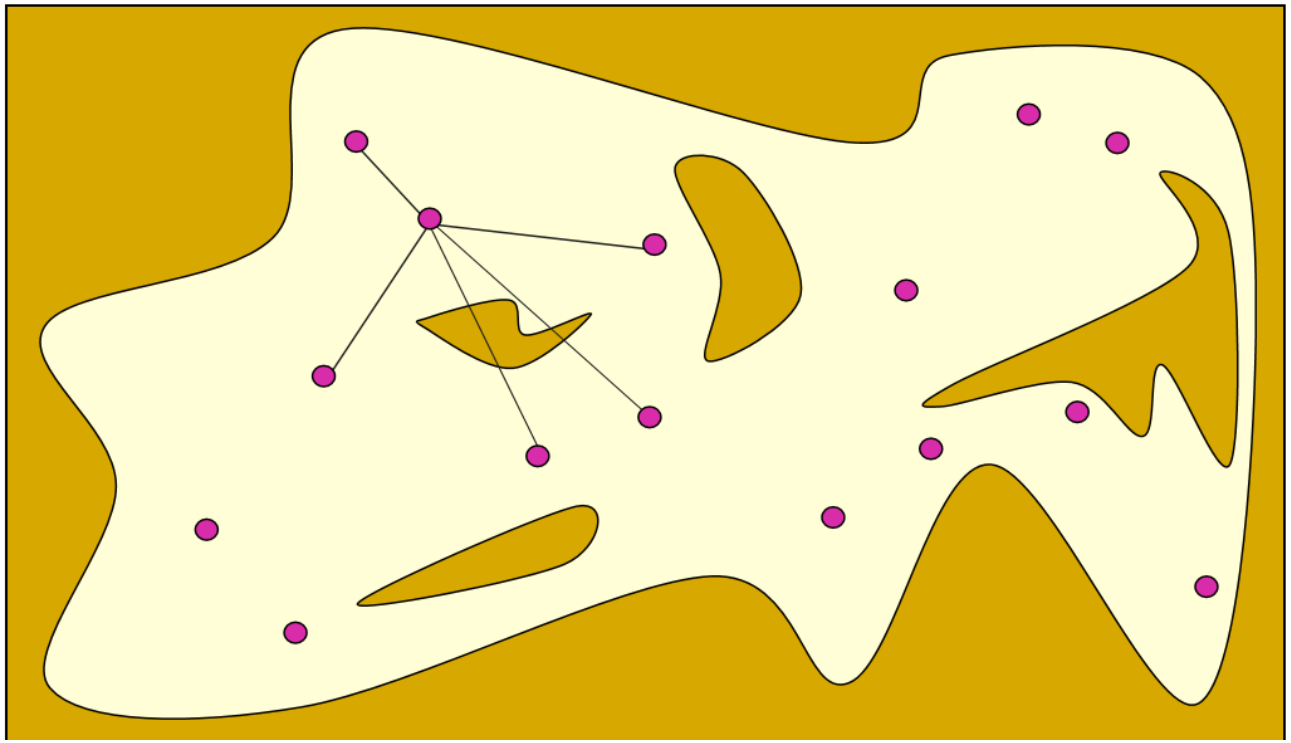
Sampling based search algorithm

3.Sampled points are tested for collision; we get the collision-free configurations



Sampling based search

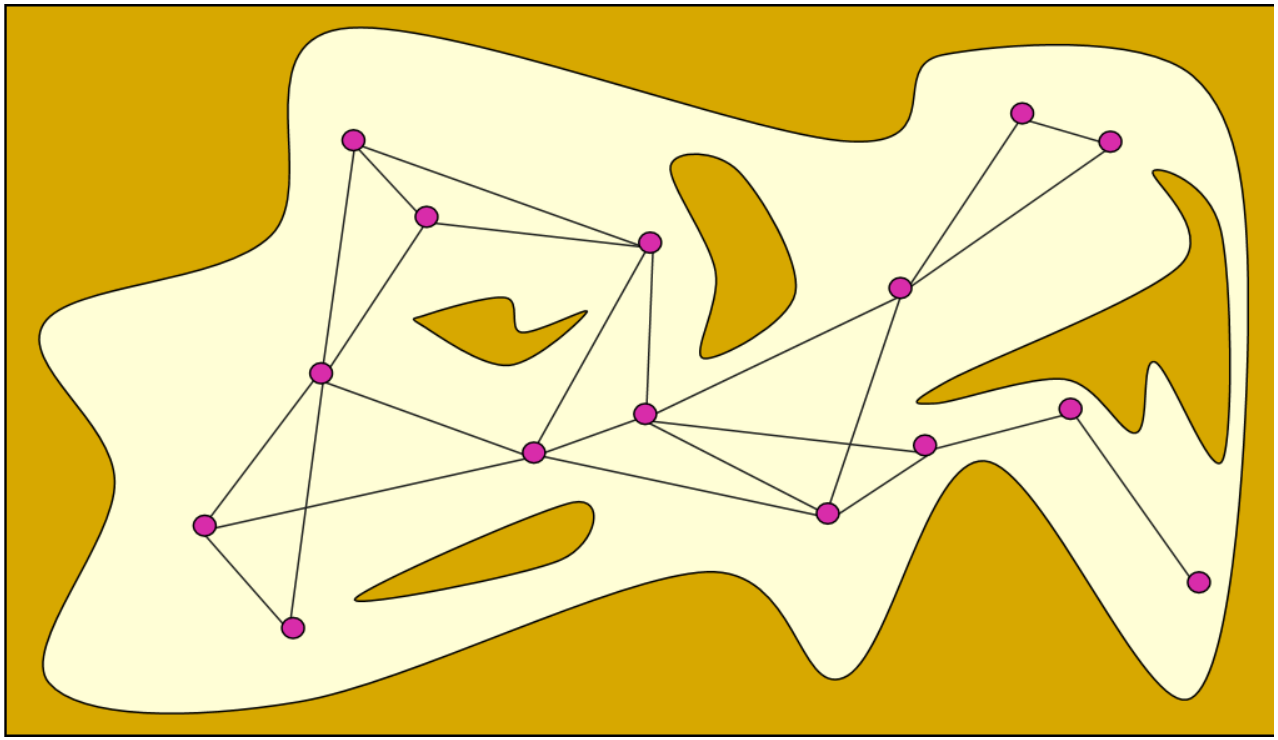
4. Each point is linked by straight paths to its nearest neighbors or remaining other points



NUS National University of Singapore

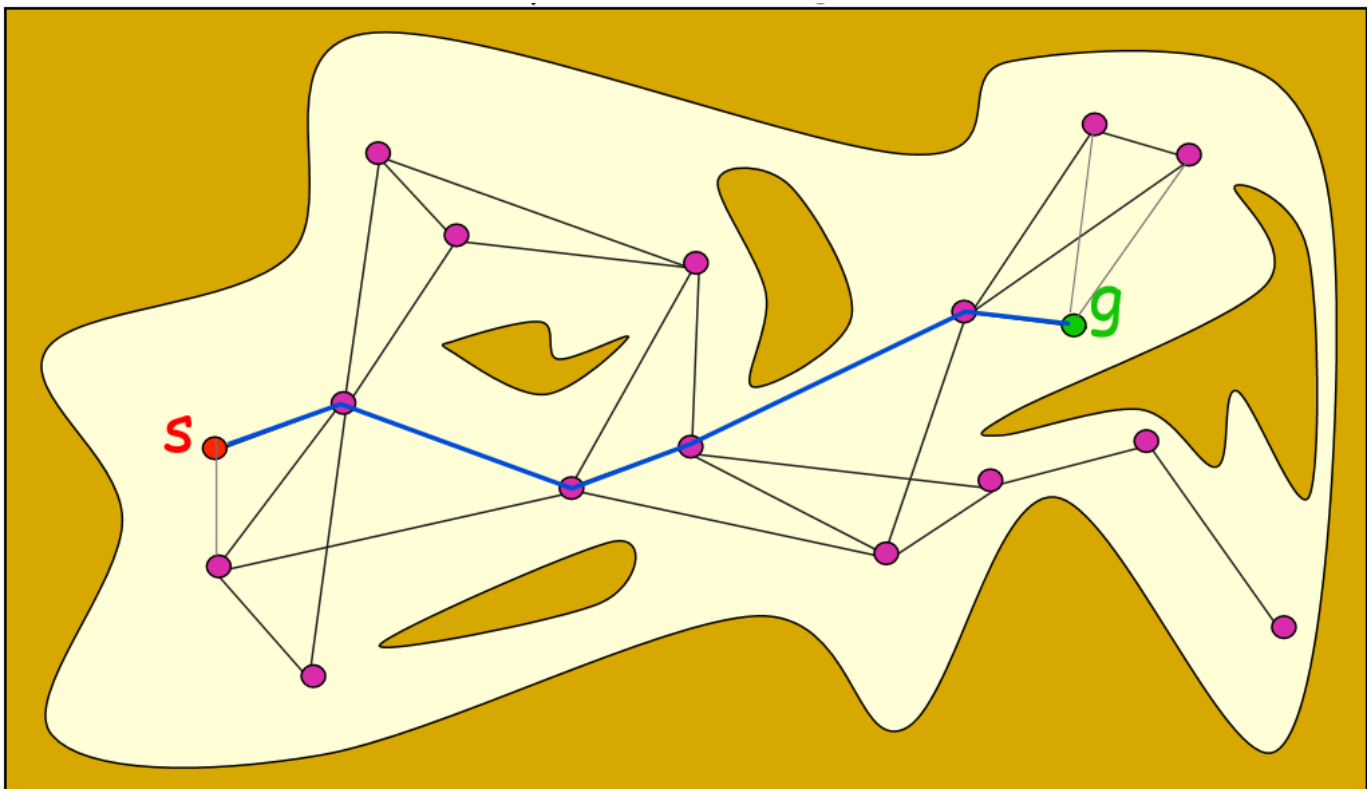
Sampling based search

5. The collision-free links are constructed as local paths



Sampling based search

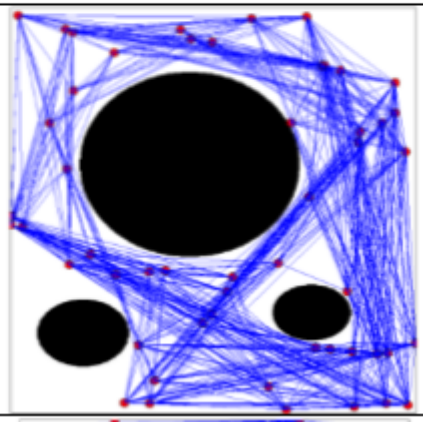
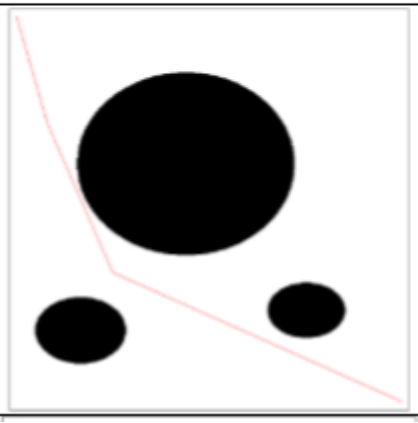
6.The start and goal configurations are included as points



NUS National University of Singapore

MATLAB demo of sampling-based search

Sample Results

S. No.	k	Roadmap	Path	Path Length	Execution Time (sec)*
1.	50			736	3.19

https://github.com/JoshuaFurman/PRM-Algorithm

NUS National University of Singapore

Summary:

- Dijkstra algorithm
- A* algorithm
- Sampling based search

NUS National University of Singapore

References

- **Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Boston, MA, 2005.**
- Jean-Claude Latombe, Robot Motion Planning

NUS National University of Singapore

Q & A

NUS National University of Singapore

Thank you!

NUS National University of Singapore

Recess time

- 15 mins