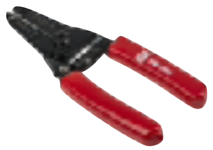
What is a robot?

✓

✗



Question. Are they robots?

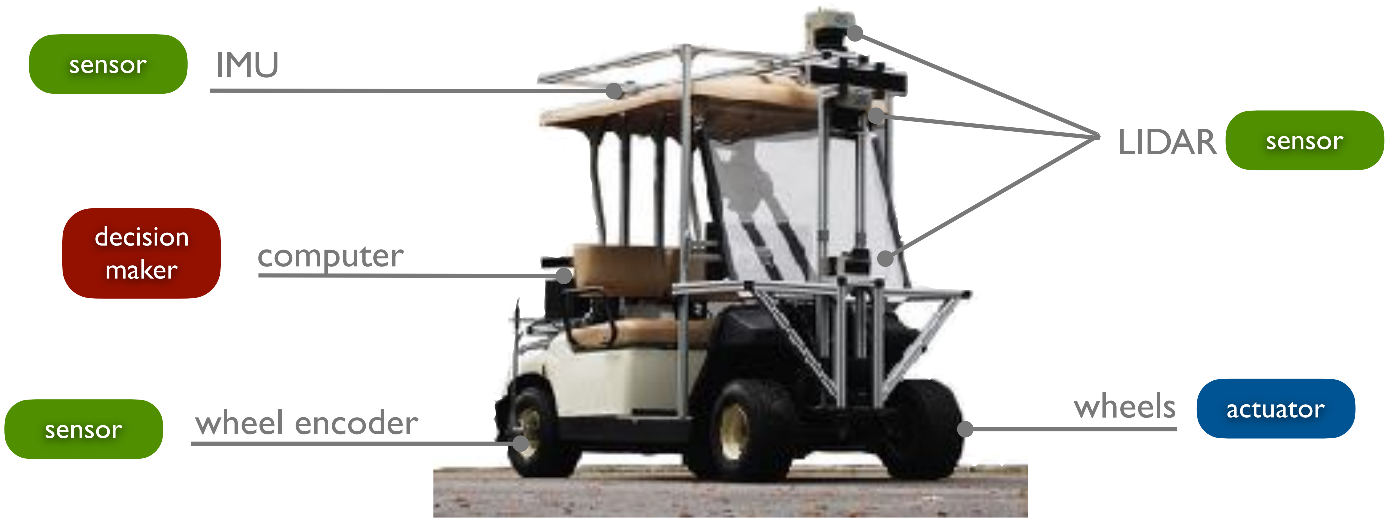
What can robots do for us?

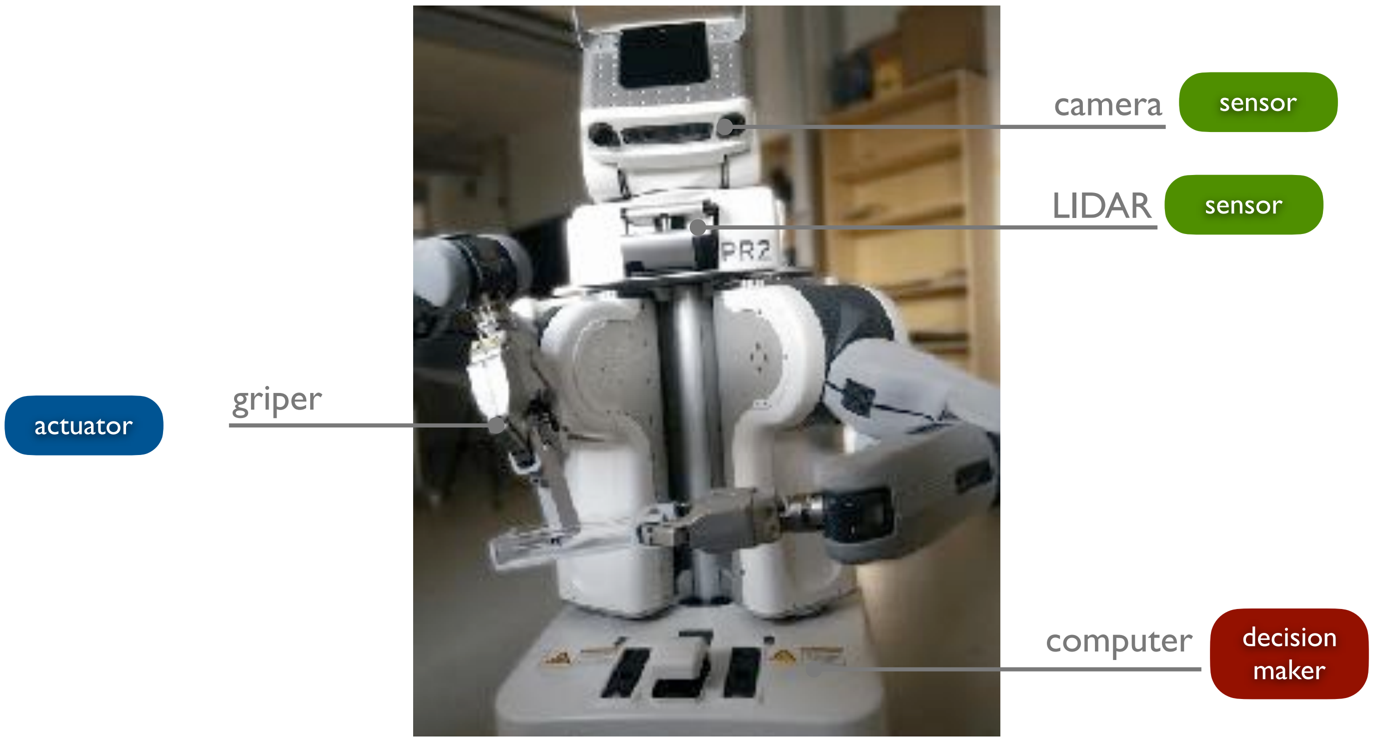
- Industrial automation (Auto assembly lines, …)
- Surgery (da Vinci, …)
- Warehouse automation (Amazon/Kiva, …)
- Autonomous driving (Waymo, …)
- Delivery service
- Home care/hospital care
- Domestic help
- …

Will robots take away jobs from humans?

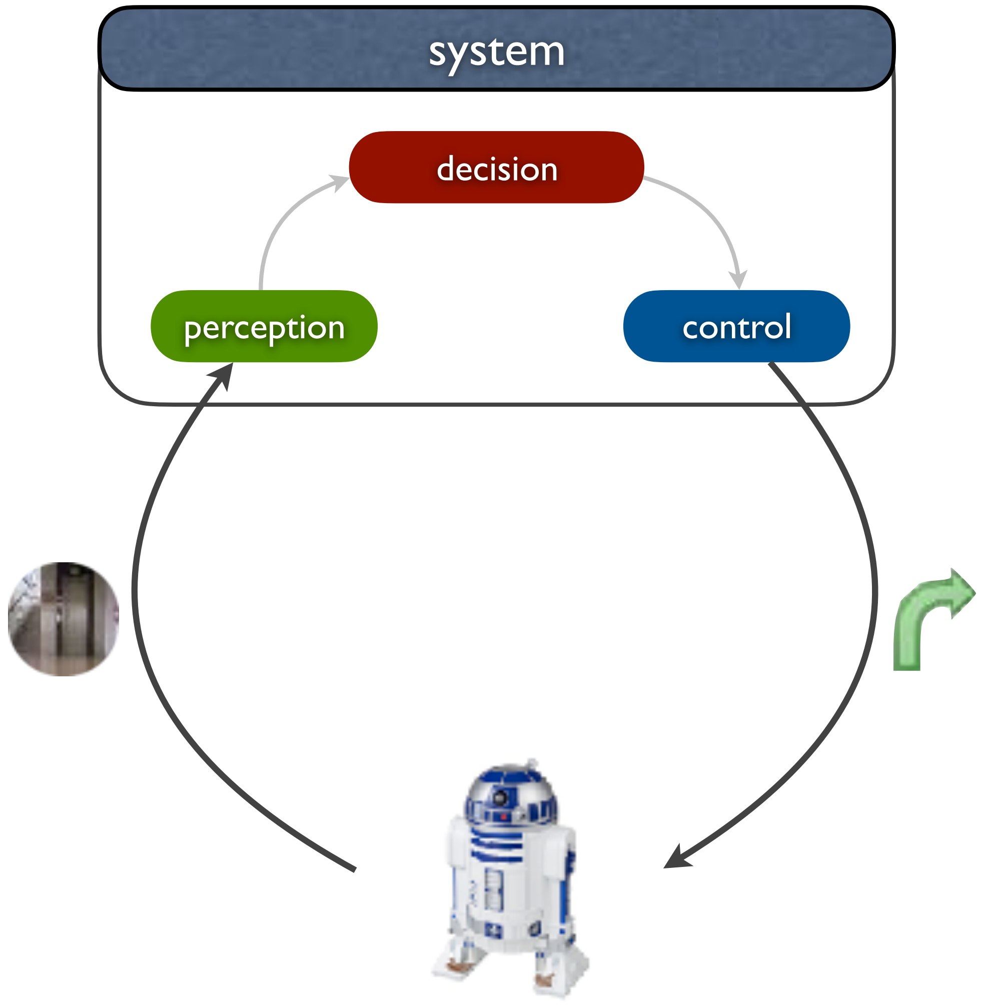
- Possibly, in the future;
- Probably no, if you are doing interesting jobs;
- No, if you study robotics.

What is a robot system made of?





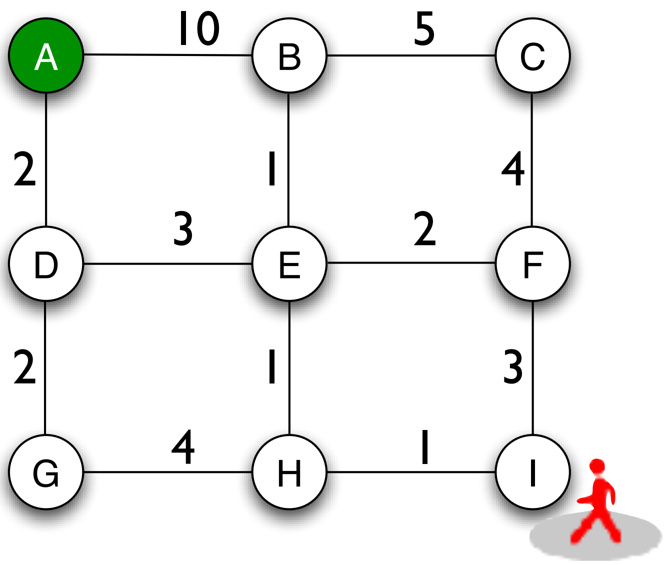
A classic robot architecture.



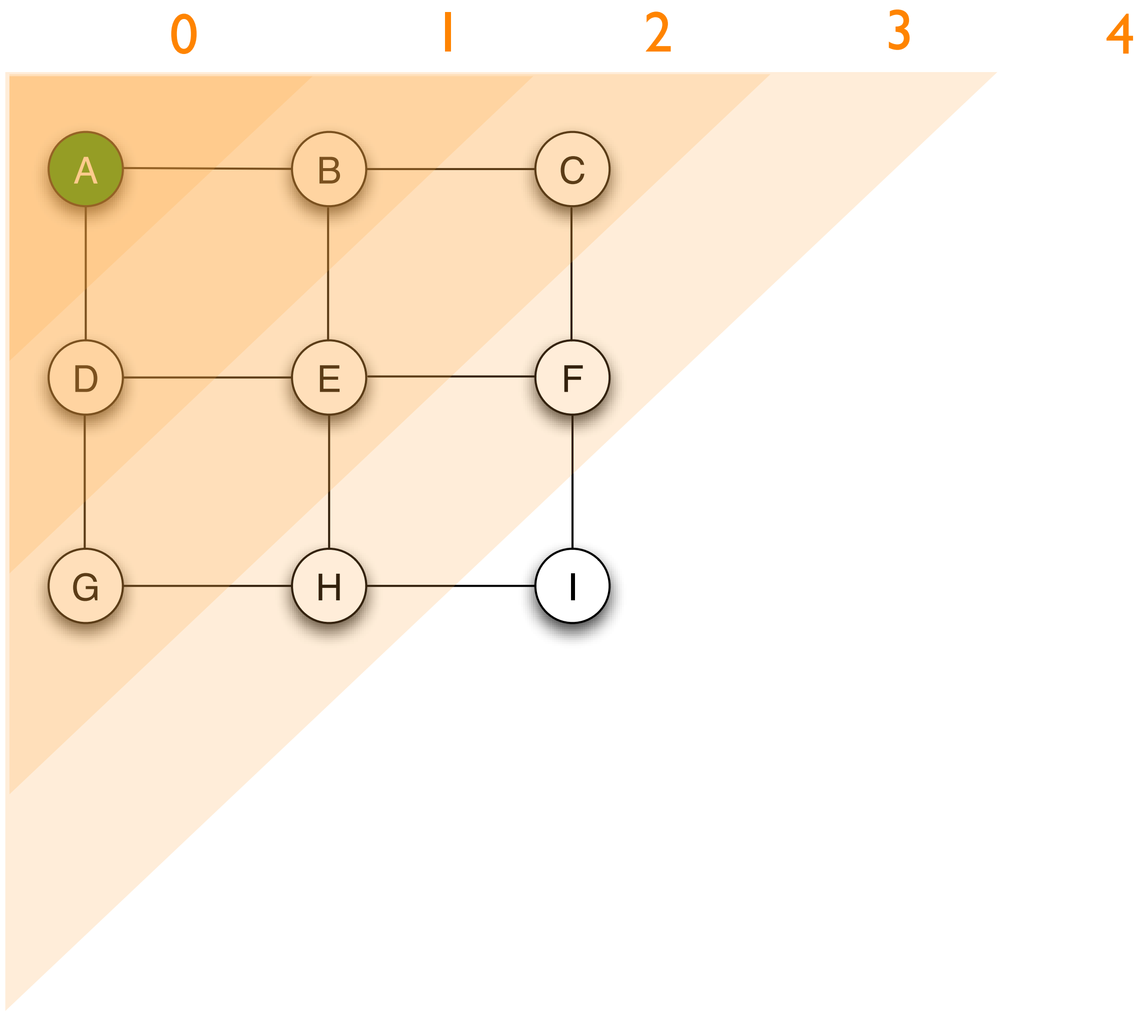
Decision making is key to robot intelligence.

Planning. Planning enables the robot to decide on a sequence of actions and reach a specified goal.

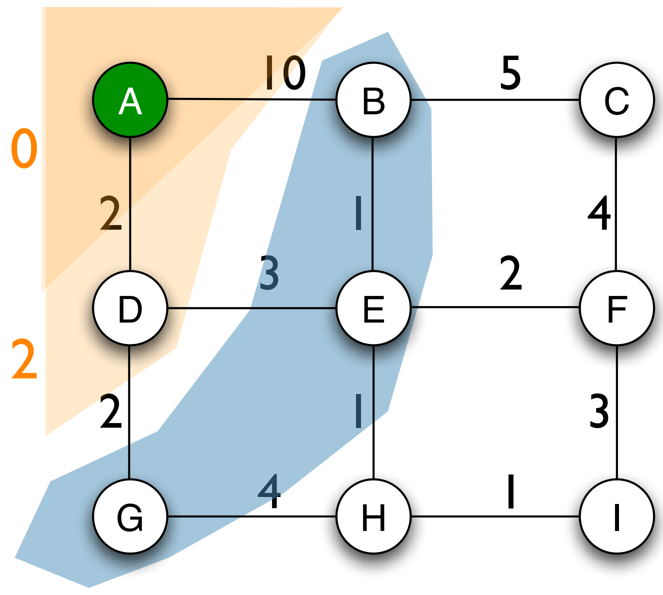
Planning in a graph. Find a shortest path from the start node I to the goal node A in a weighted graph.



To solve the problem, we simplify the graph and consider a graph with unit edge weights. Imagine that water originates from a source at A and flows along all edges at the same rate. When the "wavefront" reaches a node v for the first time, it provides the minimum time to reach v or equivalently, the minimum distance to v. This is sometimes called the wavefront propagation algorithm.



Now, let us go back to finding the shortest path in a weighted graph. By treating the edge weight as the distance between two nodes, we can apply exactly the same wavefront propagation idea, though the wavefront now looks a little jagged because of different edge weights.

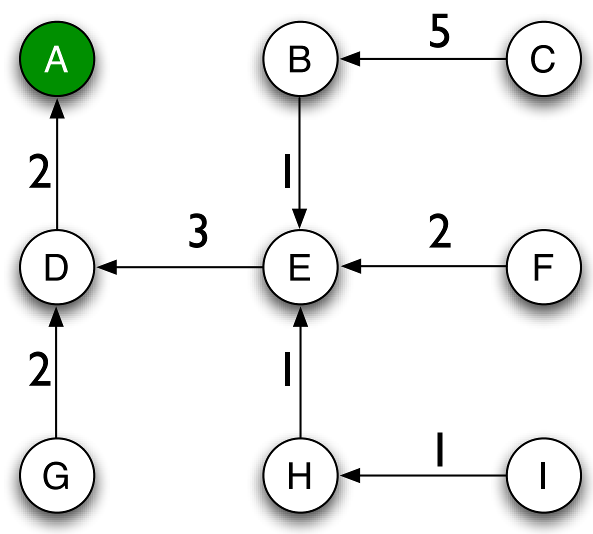


Dijkstra’s algorithm

```
Input
  A : adjacency list representation of a graph
  weight : edge weights
  n : the number of nodes in the input graph
  g : a given goal node
Output
  pred : a predecessor list encoding a shortest path tree

Dijkstra(A, weight, n, g)
for i = 1 to n
  dist[i] = $\infty$
pred[g] = NULL
dist[g] = 0
h.init(dist)          // a priority queue
for i=1 to n
  v = h.del()
  minDist = dist[v]
  for each vertex u in A[v]
    if minDist + weight(u,v) < dist[u] then
      pred[u] = v
      h.decrease(u, minDist+weight(u,v))
return pred
```

Dijkstra’s algorithm find the short path from a start nodes to all other nodes in the graph. The solution is a shortest-path tree.



Computational efficiency. The running time of Dijkstra's algorithm depends on the data structure for representing the graph and the priority queue. Suppose that the graph is represented as an adjacent list and the priority queue is implemented as the usual binary heap.

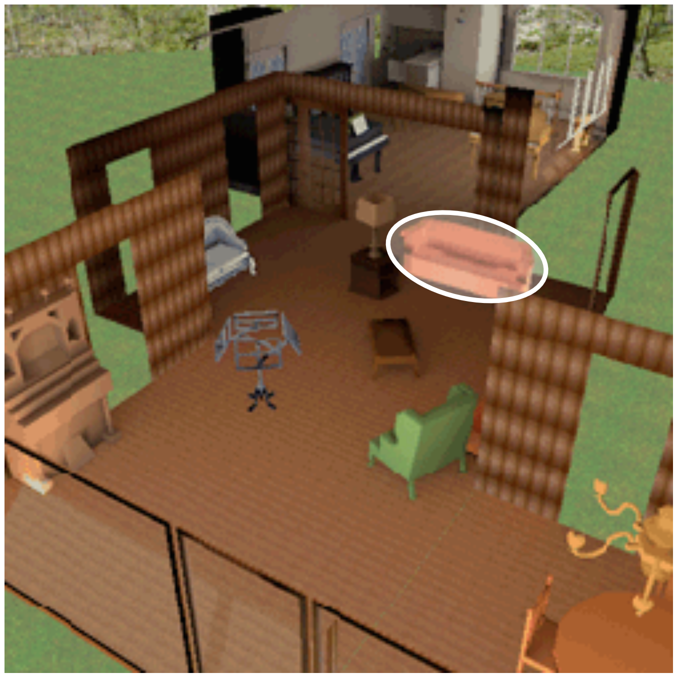
- The outer for loop (line 6) is executed once for every node of the input graph G.
- The inner for loop (line 9) is executed once for every edge of G.
- Heap deletion and decrease-key both take time $O(\lg n)$.

The total running time is $O((n+m) \lg n)$, where n and m are the number of nodes and edges of G, respectively. If G is connected, then $m > n$ and $O((n+m) \lg n) = O(m\lg n)$.

Question. If a graph is given, then we can apply the Dijkstra's algorithm or any other graph search algorithm to find a (shortest) path. How do we get the graph?

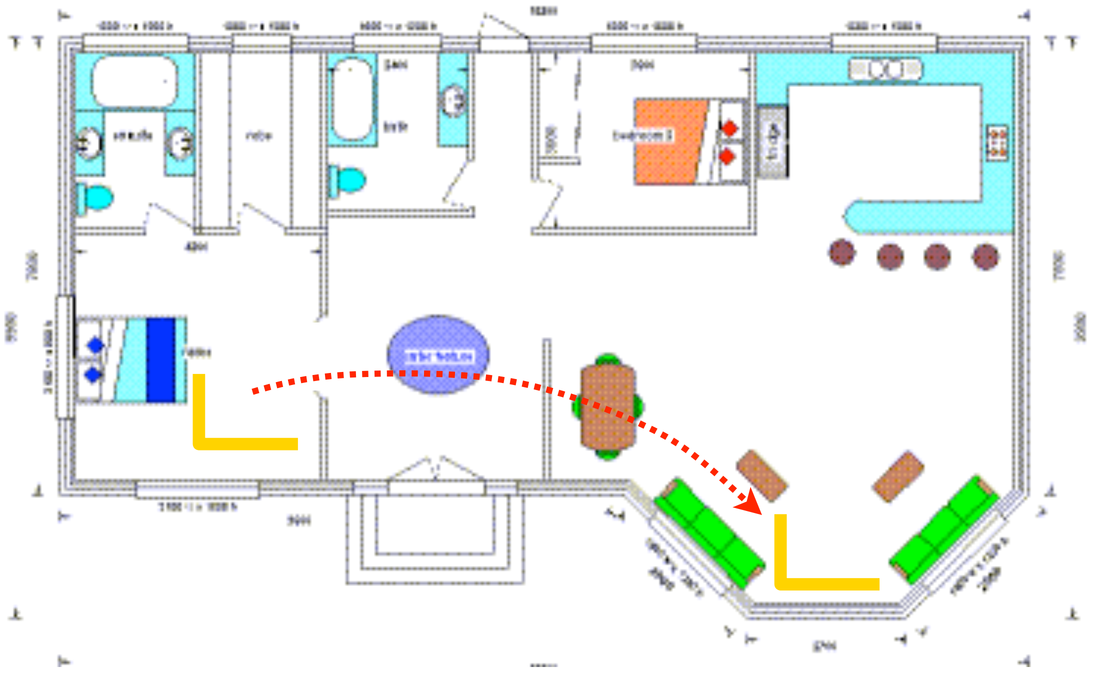
Constructing the graph is a key part of robot motion planning, and there are many ways to do it.

The piano mover's problem. In this problem, we want to move a rigid body in 3D space from an initial to a final pose without colliding with obstacles.



Again we first consider a simplified problem in 2D space.

Can the big L-shaped couch get through?

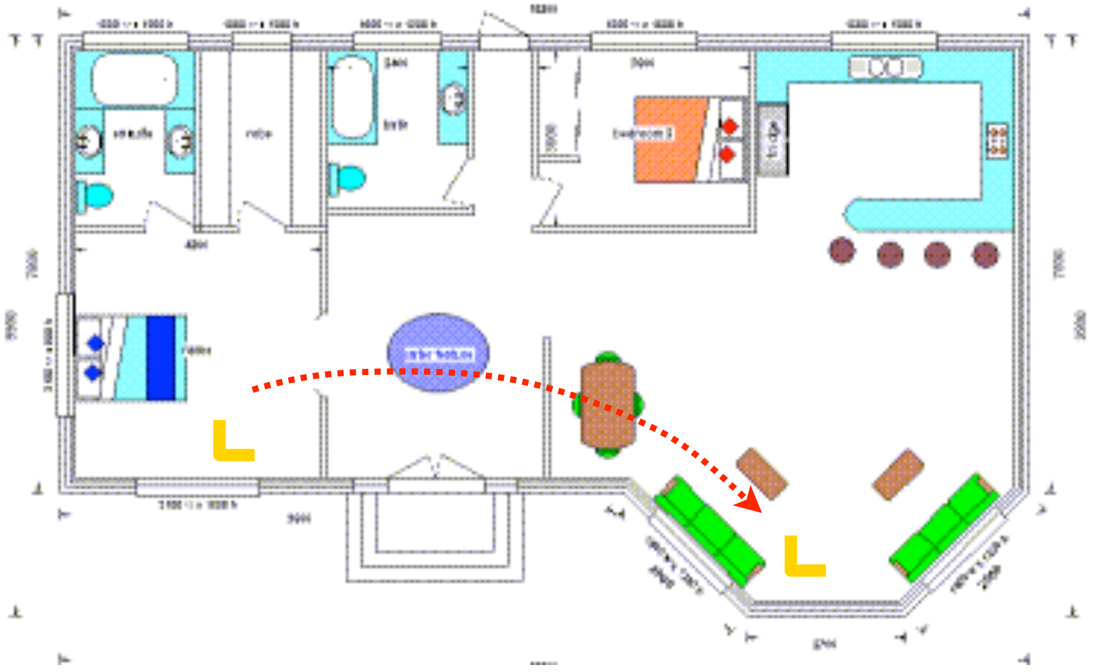


Maybe, it's not so obvious.

Simplification 1:

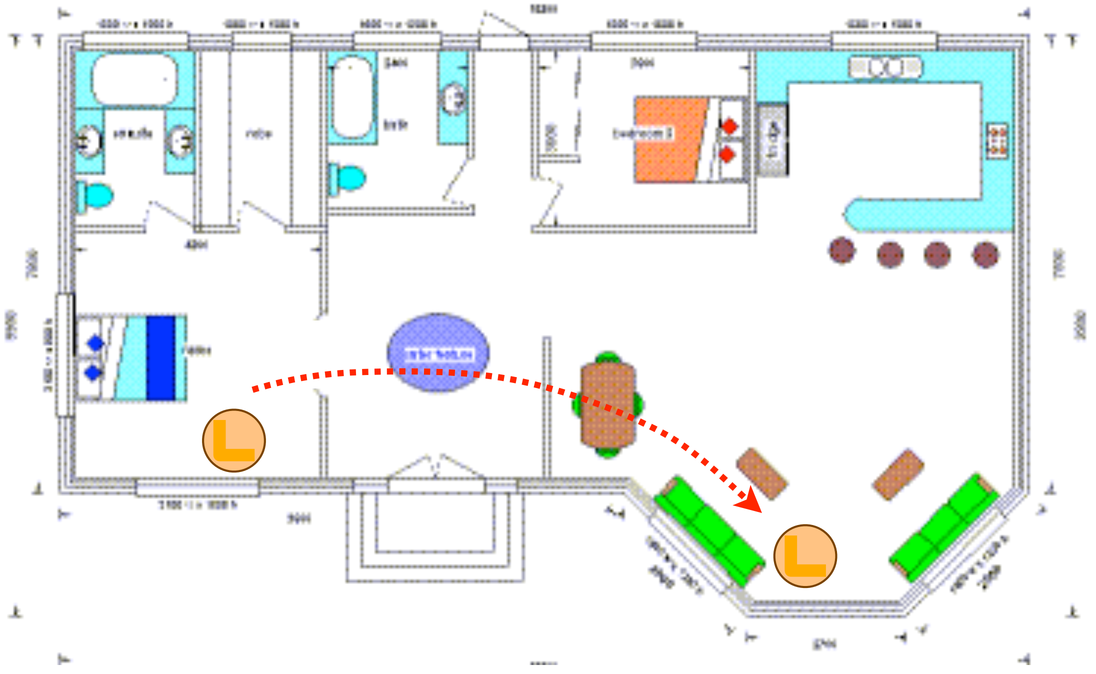
3D ➔ 2D

Can the small L-shaped couch get through?



If we say yes, why?

Can the small L-shaped couch get through?



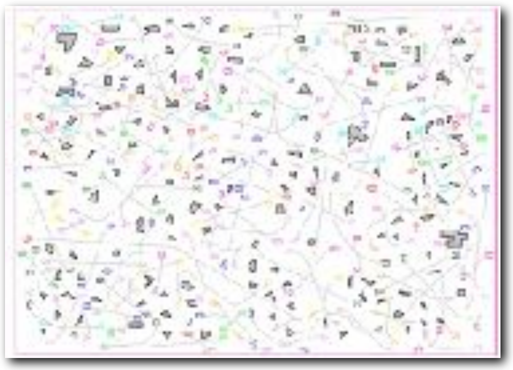
Simplification 2:

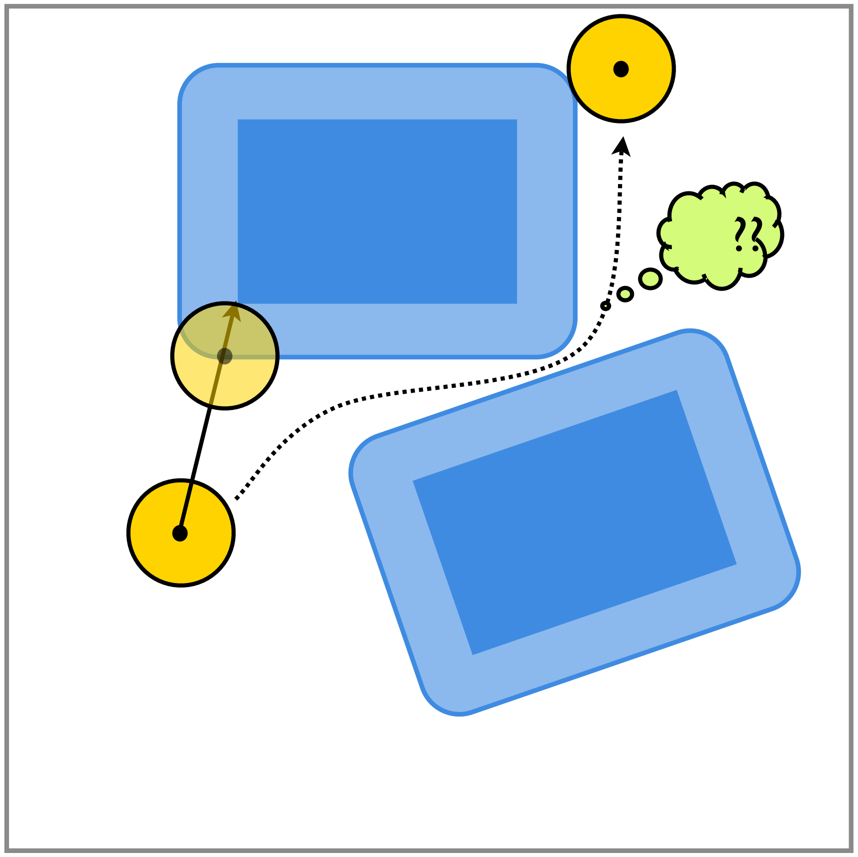
arbitrary object ➔ disc

Yes. Enclose the L-shaped couch within a disc. The diameter of the disc is no greater than the most narrow gap, e.g., the door, in this space.

How do we check whether a disc fits through all gaps?

We restate the problem. Expand each obstacle by the radius of the disc, and shrink the disc to a point. To find a collision-free path for the disc is equivalent to find a path not intersecting the expanded obstacles. Is there a collision-free **path** between two **points**?





Restate the problem:

disc ➔ point

Question. We have made two simplifications: 3D ➔ 2D and arbitrary object ➔ disk. As a result, we only need to plan the motion of a disk in a 2D environment. Because of the simplification, we have lost the completeness of the solution: a solution path may exist for the original problem, while no solution path exists for the simplified problem. How does each simplification cause incompleteness? Give examples.

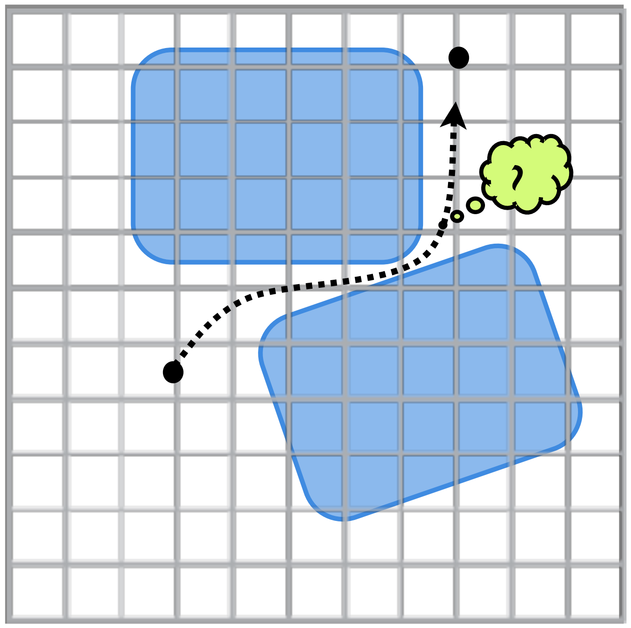
Consider a narrow doorway whose width is less than the disc radius. The L-shaped couch may get through by rotating, but the disc may fail to get through.

Question. After restating the problem, we only need to plan the path of a point. The restatement causes no further loss in the completeness of the solution. Why?

In the restated problem, there is a path not intersecting the expanded obstacles if and only if there is collision-free motion for the disc.

Cell-decomposition. How do we find a path between two point? In a cell-decomposition graph,

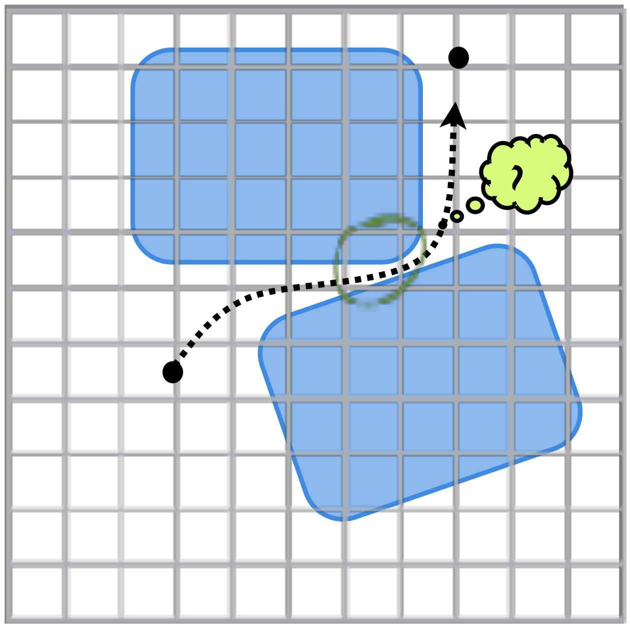
- A grid cell is free if it is not occupied by obstacles. A grid cell is occupied if it is fully or partially occupied by obstacles.
- Each node represents a free grid cell.
- There is an edge between two nodes if the two corresponding grid cells are adjacent.



Algorithm sketch.

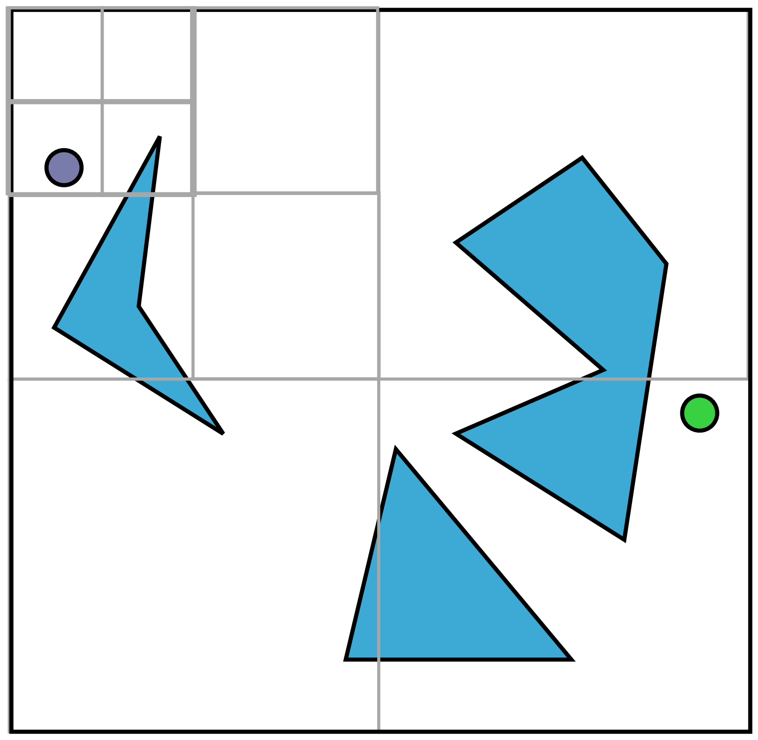
- Discretize the 2D space with a grid.
- Create the cell decomposition graph.
- Search the graph.

Question. The basic cell decomposition method is in general not complete. Why?

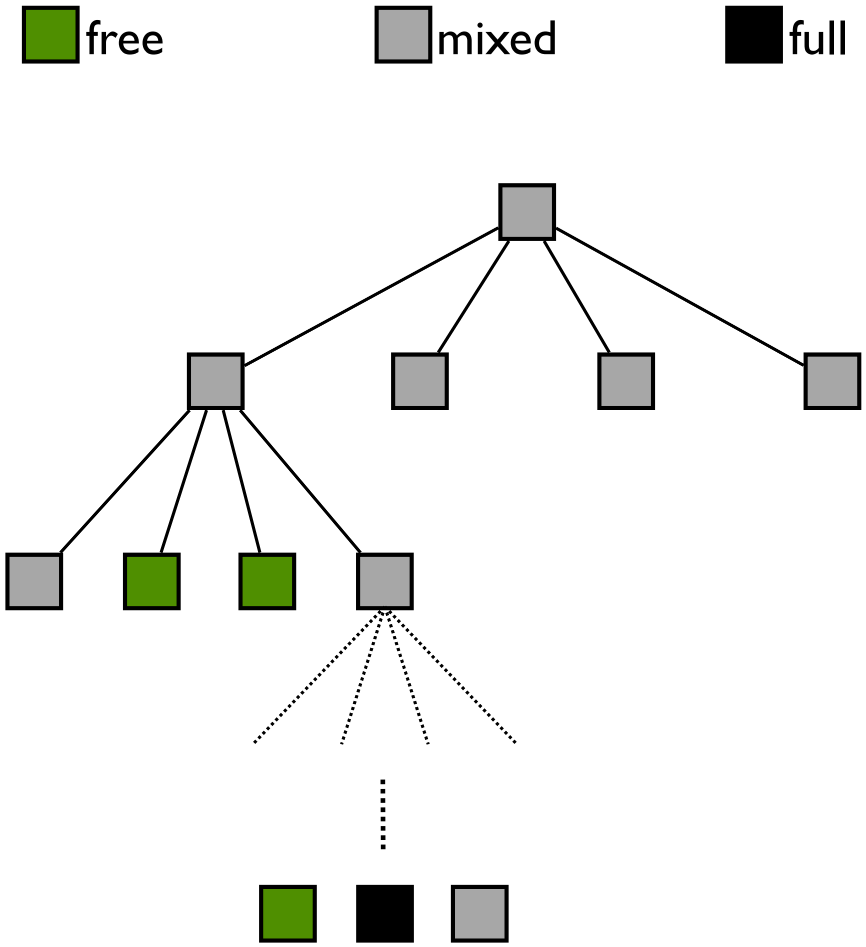


Consider the cell marked in green. Is it free according to the definition?

Hierarchical cell decomposition. Instead of discretizing the space with a fixed-size grid, we decompose the space hierarchically. A grid cell is free if it is not occupied by obstacles. A grid cell is full if it is fully occupied by obstacles. A grid cell is mixed if it is partially occupied by obstacles.



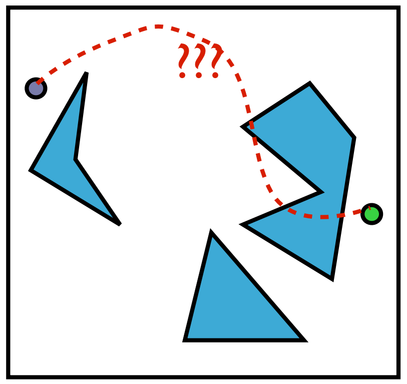
quad-tree

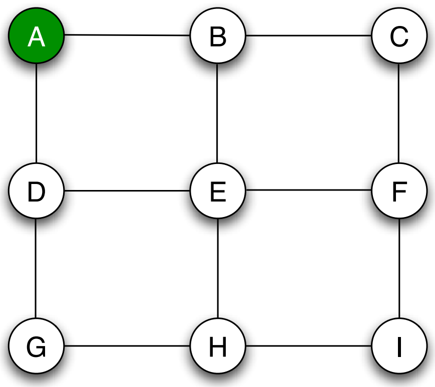


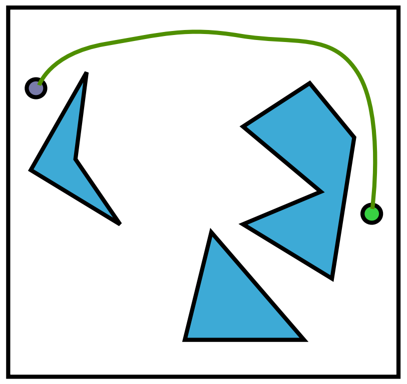
Algorithm sketch.

- Decompose the space into an initial collection of grid cells.
- Search for a sequence of mixed and free cells that connect the start and the goal.
- Decompose the mixed cells until a sequence of free cells is found between the start and the goal. Why or why not?

Summary. To plan collision-free motion for a robot, we go through two main steps:







Create a graph

cell decomposition

visibility graph

...

Search a graph

Dijkstra's

A*

best-first

...

What are the assumptions?

- The robot knows its precise location all the time.
- The robot executes the required actions exactly.

Under these assumption, we have developed a primitive, but fully working system for robot navigation.

Question. Would this system work for the following robots?

Localization.

- GPS
- Visual recognization
- …

Uncertainties.

- Bayesian filtering
- (Partially observable) Markov decision process
- …