Module 3: Mission Planning in Driving Environments

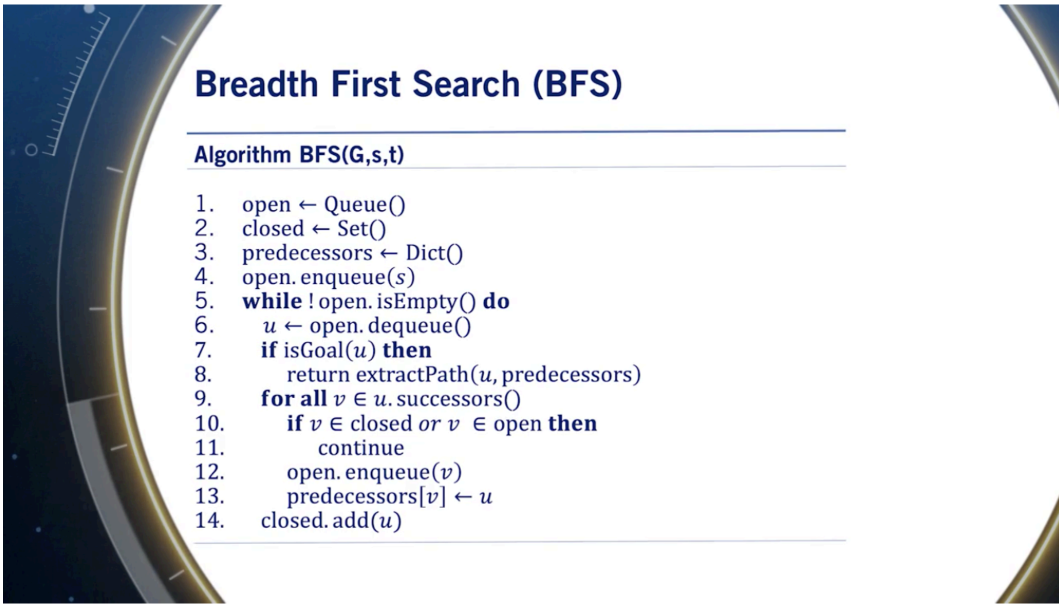
Lesson 1: Creating a Road Network Graph

単語
- Closure: The closure of a road or border is the blocking of it in order to prevent people from using it.

内容
- The mathematical concept of a graph.
- Use a directed graph to represent a road network.
- Implement Breadth-First Search.

Mission Planning
- The mission planner will focus on aspects of the road network when planning, such as speed limits and road lengths, traffic flow rates and road closures.

Graphs
- Since our graph formulation is currently unweighted, a good candidate algorithm is the Breadth-First Search or BFS.



Breadth First Search (BFS)
- At a high level, BFS can be thought of as iterating through all of the vertices in the graph but doing so in a manner such that all adjacent vertices are evaluated first before proceeding deeper into the graph.
- extractPathする時、途中$predecessors[u]$に複数あっても、どのpredecessorを経由しても必ずsに行ける?
  - そうだろう。sから探索するから。しかし複数のpredecessorの可能性はなさそう。なぜなら、$vertex$がopen queueに入れられる時predecessorをつけるでしょう。つまり一回だけつけるでしょう。
  - でもloopがあったら? ($if\ \ v \in closed\ \ or\ \ v \in open\ \ then\ \ continue$はこれを防ぐ)

- つまりgraphにloopがあるかもしれないが、searchはtreeの形になる。
- This prevents us from getting stuck in cycles during the graph search.

- A dictionary is an unordered set of key-value pairs and for each node in the closed set, stores a predecessor vertex that will identify momentarily.
- Because we use a queue to store open vertices, we ensure that all adjacent vertices at the current depth in the search are processed before proceeding deeper into the graph.

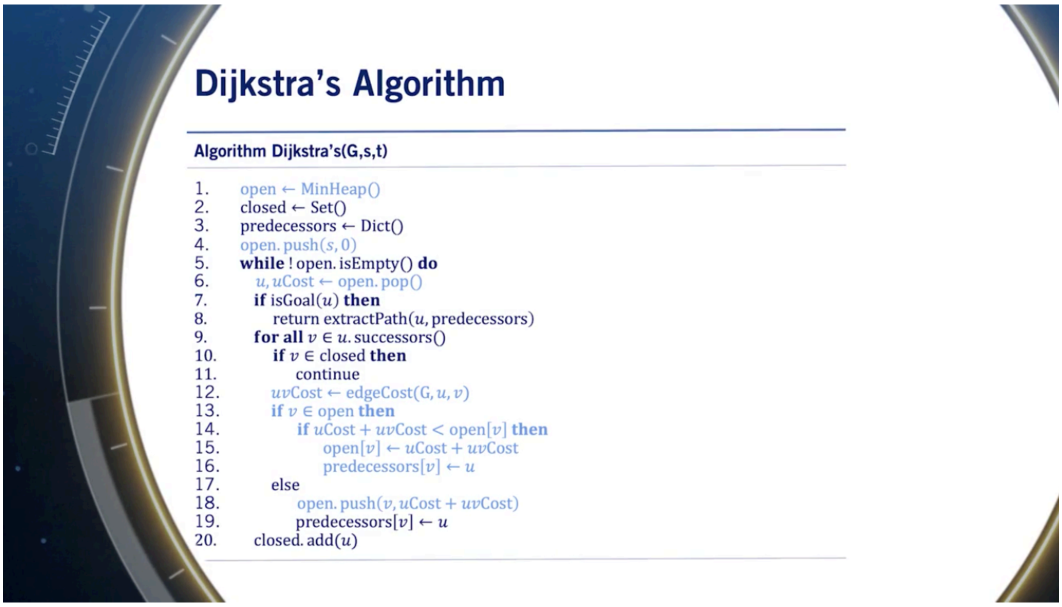
Lesson 2: Dijkstra's Shortest Path Search

内容

- The difference between weighted and unweighted graphs.
- Recognize the value of weighted graphs to the mission planning problem.
- Be able to implement Dijkstra's algorithm in a mission planning context to find the shortest path to a destination in a graph.

Weighted Graph

- The units of the weights are arbitrary, as long as they are common to all edges.



Dijkstra's Algorithm（大事、先生の説明が凄い）

- The main difference is in the order we process the vertices.
- A MinHeap is a data structure that stores keys and values, and sort the keys in terms of their associated values from smallest to largest.
- In our case, the values of each key vertex in the graph will correspond to the distance it takes to reach that vertex, along the shortest path to that vertex we've found so far.
  - In this sense, Dijkstra's algorithm processes vertices with a lower accumulated cost before other ones.
  - Thus unlike BFS, a vertex that was added later in the search can be processed before when that was added earlier, so long as its accumulated cost is lower.
  - 処理順番はaccumulated costです。挿入順番じゃない。
- One interesting case however, is if we find a new path to a vertex that is already in the open heap but has not been processed yet.
  - In this case, we have to check if the newly found path to this vertex is cheaper than the older path.
- Once we process the goal vertex, we must have necessarily processed all possible predecessor vertices of the goal node.

        - Since a predecessor must have accumulated distance less than or equal to the goal vertex.
        - この保証は処理順番によるんだ。つまりもしあるnodeが本当に最短パスに位置したら、このnodeはgoal nodeの処理前必ず既に処理された。このnodeまでのcostがgoal nodeまでのcostより絶対低いから。つまり先に処理される。
        - Since all predecessor vertices have been processed, we will have found the shortest path to the goal vertex.

Search on a Map
- Example - map of Berkeley, California.
    - 2097 vertices.
    - 5740 edges.
- Example - map of New York City, New York.
    - 54,837 vertices.
    - 140,497 edges.
- Vertices of the graph correspond to intersections, and the edges correspond to road segments.
- Dijkstra's algorithm is quite efficient, which allows it to scale really well to real-world problems such as these two maps.
- While Dijkstra's is an efficient algorithm, we can leverage certain heuristics to make it even faster in practice.

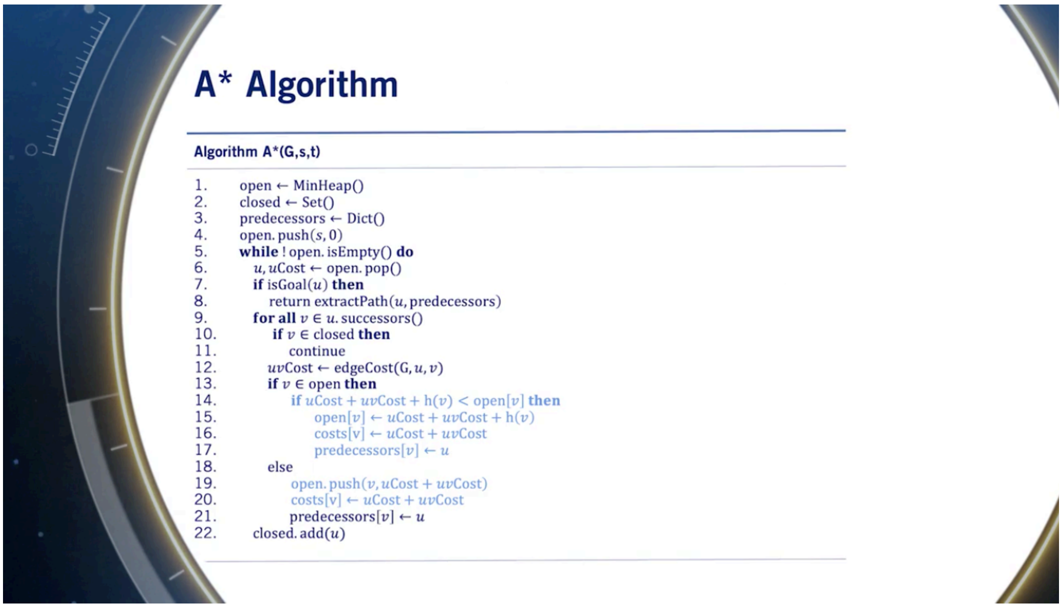
Lesson 3: A* Shortest Path Search

内容
- What admissible heuristics are in the context of graph search.
- How to use the Euclidean heuristic to improve our mission planning speed in practice.
- Implement the A* search algorithm, leveraging the Euclidean heuristic.
- How to apply A* search to variants on the mission planning problem involving time instead of distance.

Recall: Dijkstra's for Weighted Graph
- Dijkstra's algorithm required us to search almost all of the edges present in the graph, even though only a few of them were actually useful for constructing the optimal path.

Euclidean Heuristic
- Search Heuristicの意味：In this context, a search heuristic is an estimate of the remaining cost to reach the destination vertex from any given vertex in the graph.
- Exploits structure of the problem.
- Fast to calculate.
- Straight-line distance between two vertices is a useful estimate of true distance along the graph.
    - このestimateが可能な理由は、graphは地図だから。
    - $h(v) = \|t - v\|$.
- This estimate is always an underestimate of the true distance to reach the goal.（大事）
    - This is an important requirement for A* search, and heuristics that satisfy this requirement are called admissible heuristics.

A* Algorithm
- The main difference between Dijkstra's algorithm and A* is that instead of using the accumulated cost, we use the accumulated cost plus $h(v)$ (つまりestimated total cost to the goal), as the value we push onto the MinHeap.
    - In this sense, A* biases the search towards vertices that are likely to be part of the optimal path according to our search heuristic.
    - $accumulated\ cost + h(v)$の意味は、もし始点からvまで最短パスにあったら、goalまでの最低コストだよ。
    - この最低コストが小さい方が最短パスになりそうだよ、というバイアス。



間違っている、修正は下記

- Since we are storing a heuristic based total cost in the MinHeap, we also need to keep track of the true cost of each vertex as well, which we store in the $cost$ structure.
- An interesting thing to note is that if we take our heuristic to be zero for all vertices which is still an admissible heuristic, we then end up with Dijkstra's algorithm.
- スライドのアルゴリズムは間違っている。
  - 6行目：uCostは$costs[v]$から取るべき！なぜなら$open$にはhが入るので、本当のコストじゃない!
    - そういえば$costs$ dictionaryの初期化も欠いている。
  - 14行目：スライドの判定もいいけど、$if\ uCost + uvCost < costs[v]\ then$の方がいいでしょう。
  - 19行目：$open.push(v, uCost + uvCost + h(v))$.
  - 参考ページ:
    - https://en.wikipedia.org/wiki/A*_search_algorithm
    - http://theory.stanford.edu/~amitp/GameProgramming/ImplementationNotes.html

Extension to Other Factors
- Traffic, speed limits, and weather affect mission planning.
- Time rather than distance is better at capturing these factors.
- Replace distance edge weights with time estimates.
- use $\dfrac{h(v)}{v_{max}}$ as the new heuristic.

単語
- Decimal place: the position of a digit after the decimal point, each successive position to the right having a denominator of an increased power of ten.