Dijkstra in Disney World

Elizabeth Qiu

May 23, 2022

Abstract

For my final math project, I may or may not have started the project on the day of the due date. I'm procrastinating, yet always trying to find the most efficient and easiest way to get things done. So I thought it would be fitting to write about finding the shortest path for X. I am going to write about Dijkstra's algorithm.

1 Introduction

Say I wanted to go from my home (point A) to Disney World (point B). It would be a shame for Google Maps to first take me to Niagara Falls, and then to Seattle, before directing me to Orlando, Florida. In fact, I think Google would be bankrupt by now if it offered that quality of a service.

But let's say that there's a huge traffic jam all throughout US Route 301 (Maryland to Florida), and now I have to be stuck in the long line of cars. If I take a slightly longer method using local roads, I may actually end up at Disney World faster than using the inefficient highway.

That is where Dijkstra's algorithm comes in handy! It is all about getting from point A to B using the fastest method.

2 The Basics

2.1 Credit where it's due

Dijkstra's algorithm was developed by Edsger Dijkstra. He was a prominent Dutch computer scientist, software engineer, systems scientist, and science essayist ~~and balloon artist~~.

2.2 Background

Unlike breadth-first search that looks out for the shortest path with the fewest segments, Dijkstra's algorithm sets out to find the shortest path with the lowest cost in terms of weights. Dijkstra's algorithm only works on weighted graphs, while breadth-first search is handy for unweighted graphs. For the algorithm to work, the edges must also be directed, and must not have negative weight values.

2.3 The algorithm

Dijkstra's algorithm is consisted of 4 iterable steps for a graph. Also, the implementation is very complicated, as least for me. So, I've used pseudo-code instead of a real language. ...

1. Find the "cheapest" node — the node that you can get to in the least amount of time.
2. Check if there's a cheaper path to the neighbors of the node. If so, update the costs.
3. Repeat steps 1-2 until you've gone through every node in the graph.
4. Add together the weights to find the value of the final (cheapest) path.

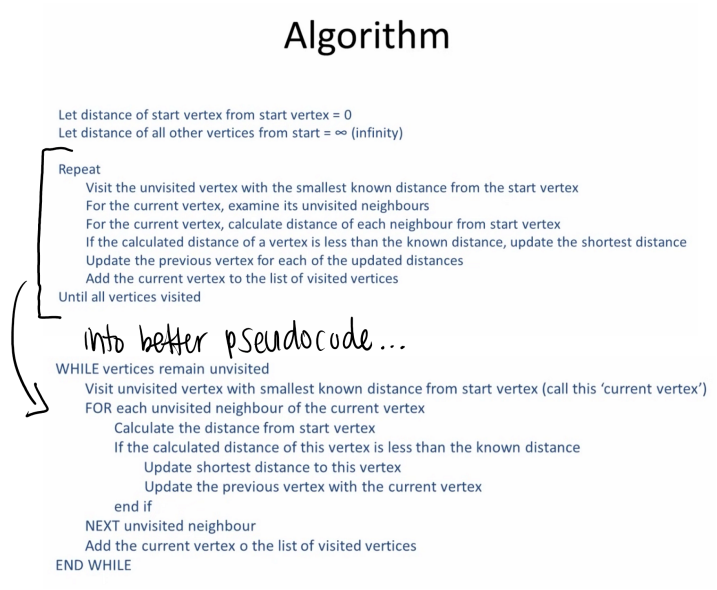

Figure 1: The algorithm described from words to pseudo-code. Credits: A YouTube video

2.4 Implementation

If you watch the video, this will make much more sense! We'll be taking a trip from Maryland (or DC) to Disney World in Florida. For Dijkstra's algorithm, we need 3 hash tables,

1. Graph
2. Costs
3. Parents

where a hash table is a data structure that stores key-value pairs with a hash function.

```
# dijkstra.python
# PLEASE NOTE: this will not actually run because i didn't completely define some stuff :)
# credits: Grokking Algorithms

# dictionary
parents = {}
parents["a"] = "start"
parents["b"] = "start"
parents["fin"] = None

# array
processed = []

node = find_lowest_cost_node(costs)
while node is not None:
    cost = costs[node]
    neighbors = graph[node]
    for n in neighbors.keys():
        new_cost = cost + neighbors[n]
        if costs[n] > new_cost:
            costs[n] = new_cost
            parents[n] = node
    processed.append(node)
    node = find_lowest_cost_node(costs)
```

Figure 2: This map was uploaded via the file-tree menu.

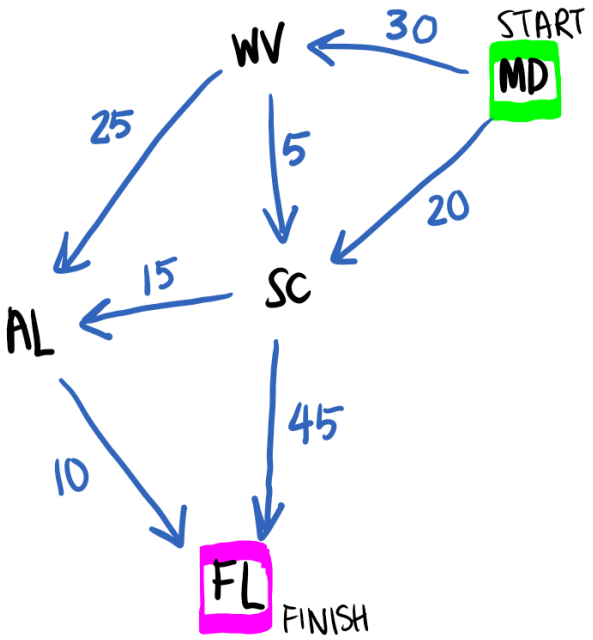

Figure 3: For calculation purposes, VA, NC, TN, and GA do not exist (sorry!) — this is what a graph of the map above may look like.

Figure 4: From the information on the graph, these are the 3 hash tables we have generated.

```
def find_lowest_cost_node(costs):
    lowest_cost = float("inf")
    lowest_cost_node = None
    for node in costs:
        cost = costs[node]
        if cost < lowest_cost and node not in processed:
            lowest_cost = cost
            lowest_cost_node = node
    return lowest_cost_node
```

2.5 Thank you!

Thanks for reading! I hope you got a little better idea of how Dijkstra's algorithm works.

References

[1] "Grokking Algorithms" by Aditya Y. Bhargava

[2] "Dijkstra's Shortest Path Algorithm", a YouTube video made by the Computer Science channel ...