WORKSHEET

A5.7 Dijkstra's shortest path algorithm

Algorithms and complexity · A level · OCR H446 2.3.1, AQA 7517 4.3.6.1, Eduqas A500QS 1.3 · about 30 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

Tracing Dijkstra's algorithm in a table, why negative weights break it, a priority queue version, its efficiency and applications.

- The algorithm
- A trace
- Why it works, and when it does not
- In Python
- Efficiency
- Where it is used

Questions

5 marks in all

1. In Dijkstra's algorithm, which vertex is visited next? [1 mark]

- ☐ **A** The unvisited vertex with the smallest distance so far
- ☐ **B** The neighbour joined by the lightest edge
- ☐ **C** The next vertex in alphabetical order
- ☐ **D** The vertex with the most edges

2. Why can Dijkstra's algorithm give a wrong answer if a graph has a negative edge weight? [1 mark]

- ☐ **A** A vertex's distance is fixed when it is visited, but a later negative edge could make a shorter route to it
- ☐ **B** It cannot store negative numbers
- ☐ **C** It loops forever
- ☐ **D** It only works on directed graphs

3. In the lesson's graph (A-B 4, A-C 2, B-C 1, B-D 5, C-D 8, C-E 10, D-E 2, D-F 6, E-F 3), what is the shortest distance from A to E? [1 mark]

4. What does this print? It reads a route back from a table of previous vertices. [1 mark]

```
previous = {"A": None, "B": "C", "C": "A", "D": "B", "E": "D"}
path = []
v = "E"
while v is not None:
    path.append(v)
    v = previous[v]
print("-".join(reversed(path)))
```

5. Which are applications of shortest path algorithms such as Dijkstra's? [1 mark]

Tick every answer that is true.

- ☐ **A** Satellite navigation route planning
- ☐ **B** Routing packets between routers (for example OSPF)
- ☐ **C** Planning a robot's route between waypoints
- ☐ **D** Compressing an image

The task: the shortest route table

`roads` in the starter is the graph above as a dictionary of dictionaries (an adjacency list with weights). Write `dijkstra(graph, start)` that returns two dictionaries, the shortest distance to each vertex and the previous vertex on that route (`None` for the start). Run it from `A` and print one line per vertex, in alphabetical order, in exactly this form, `<vertex>: <distance> from <previous vertex>`, for example: `X: 12 from Y` with `A: 0 from -` for the start. Then follow the previous vertices back from `G` and print the route and its length: `route A to G: <vertices joined with ->, cost <n>` such as `route A to G: A-X-Y-G, cost 20`. The program must find the route: do not type it.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

roads = {
    "A": {"B": 7, "C": 3},
    "B": {"A": 7, "C": 2, "D": 4},
    "C": {"A": 3, "B": 2, "D": 8, "E": 6},
    "D": {"B": 4, "C": 8, "E": 1, "F": 5},
    "E": {"C": 6, "D": 1, "F": 9, "G": 12},
    "F": {"D": 5, "E": 9, "G": 2},
    "G": {"E": 12, "F": 2},
}
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/a5-7-dijkstras-shortest-path/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Trace the task's graph by hand in a table before you run your program. Where did you have to change an entry?
2. Make the road from D to F one-way (D to F only). Does the answer change? Why not?
3. Change `dijkstra` so it stops as soon as the destination is visited. How many vertices does it visit on the way to D?