WORKSHEET

A4.8 Project: plan the route

Trees and graphs · A level · OCR H446 1.4.2, AQA 7517 4.2.4.1, Eduqas A500QS 1.1 · about 35 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

Model the mat as a graph, find the shortest route with breadth-first search, and drive it.

- The brief
- The plan
- Step 1: from grid to graph
- Step 2: search, then read the route
- Step 3: drive it
- Testing

Questions

5 marks in all

1. The robot must reach a square in the fewest moves, and every move costs the same. Why use breadth-first rather than depth-first search? [1 mark]

- ☐ **A** Breadth-first finds a shortest path in an unweighted graph; depth-first may not
- ☐ **B** Depth-first cannot find any route at all
- ☐ **C** Breadth-first uses less memory
- ☐ **D** Depth-first only works on trees

2. Why is an adjacency list a good choice for a grid of squares? [1 mark]

- ☐ **A** Each square has at most four neighbours, so the graph is sparse
- ☐ **B** A grid is a dense graph
- ☐ **C** An adjacency matrix cannot store a grid
- ☐ **D** Adjacency lists answer 'are these two squares joined?' in one look

3. A 5 by 5 grid of squares has no blocked squares. Each square is joined to the squares directly above, below, left and right of it. How many edges does the graph have? [1 mark]

_____

4. What does this program print? [1 mark]

```
from collections import deque
GRAPH = {"S": ["A", "B"], "A": ["S", "C"], "B": ["S", "D"], "C": ["A", "G"], "D": ["B"],
"G": ["C"]}
parent = {"S": None}
queue = deque(["S"])
while queue:
    v = queue.popleft()
    for n in GRAPH[v]:
        if n not in parent:
            parent[n] = v
            queue.append(n)
route = []
v = "G"
while v is not None:
    route.append(v)
    v = parent[v]
print(" ".join(reversed(route)))
```

5. After a breadth-first search records each vertex's parent, what shape do the parent links form? [1 mark]

- ☐ **A** A tree rooted at the start vertex
- ☐ **B** A cycle through every vertex
- ☐ **C** A complete graph
- ☐ **D** A queue

The task: plan the route

`GRID` is the mat from the brief, with row 0 at the far end of the mat and column 0 on the left, and `NAMES` gives the letters in reading order. The robot starts in the centre of square A facing up the mat, and square centres are 25 cm apart. Build an adjacency list for the free squares. Print `vertices:` and the number of free squares, then `edges:` and the number of edges. Find the route from A to O with the fewest moves, using a breadth-first search that takes squares from the front of a queue with `popleft()` or `pop(0)`. Print `route:` followed by the letters of the squares from A to O separated by single spaces, and then `moves:` followed by the number of moves. Finally drive the route square by square, without touching a blocked square, finishing in O.

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

from collections import deque

GRID = ["....",
        ".##.",
        "..#.",
        "#..."]
NAMES = "ABCDEFGHIJKLMNOP"

graph = {}
```

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

Do it on the robot
www.bugbotlab.com/learn/a4-8-project-plan-the-route/
The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Make the program print `no route` instead of crashing when the goal cannot be reached. Test it by blocking N and P.
2. Suppose a sideways move takes the robot twice as long as a forward or backward move. What does "shortest" mean now, and why can breadth-first search no longer find it?
3. Replace the breadth-first search with a depth-first one and print the route it finds. How many moves does it take, and does that depend on the order you try the neighbours in?