WORKSHEET

A2.9 Computational methods

Recursion and computational thinking · A level · OCR H446 2.2.2, AQA 7517 4.4.1.1, Eduqas A500QS 1.3 · about 25 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

Problem recognition, divide and conquer, backtracking, heuristics, performance modelling, data mining and visualisation.

- Is it a problem a computer can solve?
- Divide and conquer
- Backtracking
- Heuristics
- Performance modelling
- Data mining and visualisation

Questions

6 marks in all

1. What is a heuristic? [1 mark]

- ☐ **A** A rule of thumb that finds a good enough answer quickly, without guaranteeing the best
- ☐ **B** An algorithm that always finds the optimal answer
- ☐ **C** A way of storing results to avoid recalculating them
- ☐ **D** A diagram of a solution

2. Which method builds a solution step by step and, on reaching a dead end, undoes the last choice and tries another? [1 mark]

- ☐ **A** Backtracking
- ☐ **B** Pipelining
- ☐ **C** Data mining
- ☐ **D** Visualisation

3. A brute force program checks every order of visiting 8 places. How many orders is that? [1 mark]

_____

4. A supermarket searches millions of receipts and finds that people who buy barbecue charcoal often buy burgers. Which method is this? [1 mark]

- ☐ **A** Data mining
- ☐ **B** Divide and conquer
- ☐ **C** Backtracking
- ☐ **D** Heuristics

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

```
def biggest(items, lo, hi):
    if lo == hi:
        return items[lo]
    mid = (lo + hi) // 2
    a = biggest(items, lo, mid)
    b = biggest(items, mid + 1, hi)
    return a if a > b else b

print(biggest([12, 47, 5, 33, 29], 0, 4))
```

6. Which features make a problem suitable for solving by computational methods? [1 mark]

Tick every answer that is true.

- ☐ **A** It can be stated precisely, with a clear test of a correct answer
- ☐ **B** It can be solved by a finite sequence of steps
- ☐ **C** An algorithm can solve it in a reasonable time for the sizes that matter
- ☐ **D** It depends on personal taste

The task: mining the run log

The starter holds a log of 12 runs, each a dictionary with the keys `"robot"`, `"speed"` (a whole number) and `"bumps"` (how many times it hit something). Look for a pattern between speed and bumps. - Find the different speeds that appear in the log, from the data (do not type them into your code), and sort them from smallest to largest. - For each speed, print `speed <speed>: <runs> runs, <mean> bumps per run`, with the mean to 2 decimal places, for example `speed 40: 4 runs, 0.25 bumps per run`. - Finally print `most bumps: speed <speed>`, the speed with the highest mean.

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

runs = [
    {"robot": "Ada", "speed": 40, "bumps": 0}, {"robot": "Bolt", "speed": 60, "bumps": 1},
    {"robot": "Cog", "speed": 80, "bumps": 2}, {"robot": "Ada", "speed": 40, "bumps": 1},
    {"robot": "Bolt", "speed": 60, "bumps": 0}, {"robot": "Cog", "speed": 80, "bumps": 3},
    {"robot": "Ada", "speed": 40, "bumps": 0}, {"robot": "Bolt", "speed": 60, "bumps": 1},
    {"robot": "Cog", "speed": 80, "bumps": 2}, {"robot": "Ada", "speed": 40, "bumps": 0},
    {"robot": "Bolt", "speed": 60, "bumps": 2}, {"robot": "Cog", "speed": 80, "bumps": 1},
]
```

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

Do it on the robot

www.bugbotlab.com/learn/a2-9-computational-methods/

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

Challenges

1. Try all 24 orders of the four stops. What is the shortest route, and how much longer is the heuristic's?
2. Add a fifth run speed to the log. Did your mining code need changing? If it did, it was not really finding the speeds from the data.
3. Change `place(4)` to `place(8)`. Add a counter for how many times a choice is undone, and explain what it measures.