WORKSHEET

# A6.7 Limits of computation: tractable and intractable problems

Theory of computation · A level · OCR H446 2.3.1, AQA 7517 4.4.4.4 · about 25 min

Name _____ Class _____ Date _____

## What this lesson is about

How complexity and hardware limit computation, tractable and intractable problems, and a heuristic route for the robot.

- Two things limit computation
- How fast things grow
- Tractable and intractable
- The travelling robot
- Heuristics

## Questions

**6 marks in all**

**1.** What makes a problem intractable? [1 mark]

- ☐ **A** It cannot be solved by any algorithm
- ☐ **B** It has no known polynomial-time solution
- ☐ **C** It needs more than one processor
- ☐ **D** Its algorithm uses recursion

**2.** Which time complexities count as tractable? [1 mark]

Tick every answer that is true.

- ☐ **A** $O(n^2)$
- ☐ **B** $O(2^n)$
- ☐ **C** $O(n \log n)$
- ☐ **D** $O(n!)$
- ☐ **E** $O(1)$

**3.** What is a heuristic? [1 mark]

- ☐ **A** An algorithm guaranteed to find the best answer
- ☐ **B** A rule of thumb that finds a good enough answer in reasonable time, without a guarantee it is the best
- ☐ **C** A way of proving a problem is non-computable
- ☐ **D** A faster processor

**4.** A brute-force route planner tries every order of visiting n places, n! orders. How many orders does it try for 6 places? [1 mark]

_____

**5.** An $O(2^n)$ algorithm can handle n = 40 in an hour. A new computer is twice as fast. What is the largest n it can handle in an hour? [1 mark]

- ☐ **A** 41
- ☐ **B** 80
- ☐ **C** 60
- ☐ **D** 1600

**6.** What does this program print? [1 mark]

```
import math
for n in [4, 8, 12]:
    print(n, n ** 2, 2 ** n, math.factorial(n))
```

## The task: the delivery route

Plan and drive a route to all five drop points, using the heuristic, and compare it with the best route. - The drop points are in `drops`: a dictionary from a letter to an `(x, y)` position on the mat in cm. The robot starts at `START`. Distances are straight lines: use `math.dist`. - `drive_to(x, y)` is written for you: it slides the robot to that mat position. - Write `tour_length(order)`: `order` is a sequence of letters; return the total distance from `START` through each point in that order, without coming back. - Write `nearest_neighbour()`: return a list of the five letters in the order the nearest neighbour heuristic visits them, starting from `START`. - Print `nearest neighbour:` then the letters separated by single spaces, a space, then the length to 1 decimal place and `cm`. For example `nearest neighbour: A B C D E 123.4 cm`. - Find the shortest order by brute force over every order (`itertools.permutations`) and print it the same way, starting `best:`. - Then drive the **nearest neighbour** route, visiting the points in that order. Do not type any orders or lengths in; work them out.

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

import math
from itertools import permutations

START = (50, 10)
drops = {"A": (80, 45), "B": (65, 70), "C": (85, 85), "D": (20, 20), "E": (55, 35)}

def wrapped(h):
    return (h + 180) % 360 - 180

def drive_to(x, y, speed=70):
    # slide to the mat point (x, y), holding heading 0, until within 3 cm
    while True:
        px, py = position()
        dx, dy = x - START[0] - px, y - START[1] - py
        if math.hypot(dx, dy) < 3:
            break
        a = math.radians(wrapped(math.degrees(math.atan2(dx, dy)) - heading()))
        drive(speed * math.cos(a), speed * math.sin(a), wrapped(0 - heading()) * 3)
        wait(0.1)
    stop()
```

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

**Do it on the robot**

www.bugbotlab.com/learn/a6-7-tractable-and-intractable/

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

## Challenges

1. How many orders would brute force try for 10 drop points? Time your program with 8 points.
2. Improve the nearest neighbour route: try swapping every pair of stops and keep any swap that shortens it. Is the result always the best route?
3. Explain why a computer a million times faster still could not use brute force for 30 drop points.