ANSWERS

A5.1 Comparing algorithms

Algorithms and complexity · A level · OCR H446 2.3.1, AQA 7517 4.4.4.1, Eduqas A500QS 1.3 · about 25 min

BugBotLab

What this lesson is about

Time and space efficiency as functions of the size of the problem; linear, polynomial, exponential and logarithmic functions; permutations and n!.

- Why not use a stopwatch?
- Time and space
- The maths: four kinds of function
- Permutations

Questions

6 marks in all

1. Why are algorithms compared by counting operations as a function of n rather than by timing them? [1 mark]

- ☐ **A** A timing depends on the computer, the language and the data, but the growth in operations does not
- ☐ **B** Timing a program is not possible on modern computers
- ☐ **C** Counting operations always gives a smaller number
- ☐ **D** Timings are only accurate for sorting algorithms

> **Answer:** A. A stopwatch measures one run on one machine. Counting operations as n grows describes the algorithm itself.

2. Which of these is an exponential function of n? [1 mark]

- ☐ **A** 2^n
- ☐ **B** n^2
- ☐ **C** $2n$
- ☐ **D** $\log_2 n$

> **Answer:** A. In an exponential function n is the power. n^2 is polynomial, $2n$ is linear and $\log_2 n$ is logarithmic.

3. What is $\log_2 256$? [1 mark]

> **Answer:** 8. $2^8 = 256$, so 256 can be halved 8 times before it reaches 1.

4. In how many different orders can a robot visit 5 distinct checkpoints? [1 mark]

> **Answer:** 120. The number of permutations of 5 distinct objects is $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$.

5. Checking a list for repeats by storing every item seen in a set, instead of comparing every pair, makes which of these true? [1 mark]

Tick every answer that is true.

- ☐ **A** It makes fewer comparisons
- ☐ **B** It uses more memory
- ☐ **C** It uses less memory
- ☐ **D** It needs the list to be sorted

> **Answer:** A, B. It trades space for time: one check per item, but a set that can grow to n items.

6. What does this print? [1 mark]

```
n = 1
for k in range(1, 6):
    n = n * k
print(n, 2 ** 5, 5 ** 2)
```

> **Answer:**
>
> ```
> 120 32 25
> ```
>
> The loop works out 5! = 120, which is already bigger than $2^5 = 32$ and $5^2 = 25$.

The task: the growth table

Print a table of how four functions grow. For each n in 4, 8, 16, 32 and 64, print one line in exactly this form: `n=8 log2=3 square=64 exponential=256` where `log2` is $\log_2 n$ (a whole number here, because every n is a power of 2), `square` is n^2 and `exponential` is 2^n. Work out `log2` by counting how many times n can be halved (with `// 2`) before it reaches 1, not with the `math` module.

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

n = 8
print("n=" + str(n))
```

The hint students can ask for: Loop over the five values of n. For each one, copy n into another variable and keep halving the copy while it is bigger than 1, counting as you go. Build the line from the four numbers.

A solution

```
from bugbot import *
connect()

for n in [4, 8, 16, 32, 64]:
    halves = 0
    m = n
    while m > 1:
        m = m // 2
        halves = halves + 1
    print("n=" + str(n), "log2=" + str(halves), "square=" + str(n * n), "exponential=" +
str(2 ** n))
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.