ANSWERS

F5.9 Merge sort and comparing algorithms

Algorithms · GCSE · OCR J277 2.1.3, AQA 8525 3.1.2, Edexcel 1CP2 1.2.7 · about 20 min

What this lesson is about

Splitting and merging, and choosing between the searches and sorts.

- Merging two sorted lists
- Splitting and merging
- Comparing the sorts
- Choosing an algorithm

Questions

5 marks in all

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

```
left, right = [2, 8], [3, 4, 20]
result = []
i = j = 0
while i < len(left) and j < len(right):
    if left[i] <= right[j]:
        result.append(left[i]); i = i + 1
    else:
        result.append(right[j]); j = j + 1
print(result + left[i:] + right[j:])
```

Answer:

```
[2, 3, 4, 8, 20]
```

Merging compares only the front items, then adds what is left over.

2. Put the stages of merge sort in order. [1 mark]

Number the lines 1 to 4 to put them in the right order.

___ `Keep splitting until every part has one item`

___ `Merge pairs of parts into sorted parts`

___ `Keep merging until one sorted list is left`

___ `Split the list in half`

Answer:

```
Split the list in half
Keep splitting until every part has one item
Merge pairs of parts into sorted parts
Keep merging until one sorted list is left
```

Divide first, then conquer by merging.

3. Why is merge sort usually faster than bubble sort on large lists? [1 mark]

- ☐ **A** It needs far fewer comparisons as the list grows
- ☐ **B** It uses no memory
- ☐ **C** It only works on sorted lists
- ☐ **D** It skips half the items

> **Answer:** A. Bubble sort's comparisons grow with n times n; merge sort's grow much more slowly.

4. What is a disadvantage of merge sort compared with bubble sort? [1 mark]

- ☐ **A** It uses more memory, building new lists as it merges
- ☐ **B** It is slower on large lists
- ☐ **C** It cannot sort numbers
- ☐ **D** It needs the list already sorted

> **Answer:** A. Bubble sort works inside the original list; merge sort needs extra space.

5. A school has 2 million sorted records and searches them thousands of times a day. Which search should it use? [1 mark]

- ☐ **A** Binary search
- ☐ **B** Linear search
- ☐ **C** Bubble sort
- ☐ **D** Insertion sort

> **Answer:** A. The data is sorted and large, and searched often: binary search's speed wins.

The task: merge two tunes

Two robots each played a sorted tune: `left = [262, 330, 440, 523]` and `right = [294, 349, 392, 494]`. Write `merge(left, right)` yourself (no `sort` or `sorted`) and use it to make one sorted tune. Print `merged: <the list>` and play every note for 0.2 seconds.

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

left = [262, 330, 440, 523]
right = [294, 349, 392, 494]
```

The hint students can ask for: Walk both lists at once with a position in each. Take the smaller of the two items in front of you and move that position on. When one list runs out, the rest of the other follows.

A solution

```
from bugbot import *
connect()
left = [262, 330, 440, 523]
right = [294, 349, 392, 494]

def merge(left, right):
    result = []
    i = 0
    j = 0
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i = i + 1
        else:
            result.append(right[j])
            j = j + 1
    return result + left[i:] + right[j:]

tune = merge(left, right)
print("merged:", tune)
for note in tune:
    tone(note, 0.2)
```

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