OCR GCSE (J277)

Module L: Standard algorithms

Workbooklet · 7 parts · 146 marks

Contents

Part	Topic	Marks
L1	Linear search	21
L2	Binary search	21
L3	Bubble sort	21
L4	Merge sort	20
L5	Insertion sort	20
L6	Comparing the search and sort algorithms	22
L7	Totalling, counting, maximum, minimum and average	21
	Total	**146**

MODULE L STANDARD ALGORITHMS · L1

Linear search

Original practice questions · 21 marks · ~25 minutes · OCR J277 spec 2.1.3

Name _____ Date _____

Instructions. Answer all questions in the spaces provided. The marks for each part are shown in brackets [ ]. The total for this worksheet is **21 marks**. Code is written in OCR Exam Reference Language. Arrays start at index 0. All codes on this sheet are made up.

1

Total: 4 marks

(a) Tick (✓) **one** box in each row to show whether each statement about a linear search is true or false. **[4]**

Statement	True	False
A linear search starts by checking the first item in the list.		
The data must be sorted before a linear search can be used.		
A linear search can stop as soon as the item it is looking for is found.		
A linear search checks the middle item of the list first.		

2

Total: 3 marks

(a) Describe the main steps of a linear search. **[3]**

3

Total: 4 marks

A warehouse robot stores the shelf codes it can reach in the array `shelves`. The codes are in no particular order.

Index	0	1	2	3	4	5	6	7
shelves	P41	C08	R17	D55	C08	H23	M90	B12

(a) The robot uses a linear search that stops when the item is found to look for `"D55"`. List, in order, every shelf code that is compared with `"D55"`. **[2]**

(b) State the index at which `"D55"` is found. **[1]**

(c) The robot then uses a linear search that stops when the item is found to look for `"C08"`. State the index at which `"C08"` is found. **[1]**

4 **Total: 7 marks**

A program checks whether a sensor ID typed in by an engineer is stored in the array `sensors`.

```
01 sensors = ["S4", "S9", "S2", "S7", "S2"]
02 target = input("Enter sensor ID")
03 found = False
04 i = 0
05 while i < 5 AND found == False
06     if sensors[i] == target then
07         found = True
08     else
09         i = i + 1
10     endif
11 endwhile
12 if found == True then
13     print("Found at index " + str(i))
14 else
15     print("Not found")
16 endif
```

(a) Identify the standard searching algorithm used in this program. **[1]**

(b) Identify the line number where an item in the array is compared with the target. **[1]**

(c) State the purpose of the variable `found` in this program. **[1]**

(d) The engineer enters `S2`. Complete the trace table for the program. You may not need to use all the rows. **[3]**

Line number	i	found	Output

(e) State what the program outputs if the engineer enters `S5`. **[1]**

5 **Total: 3 marks**

A delivery company stores 50 000 parcel records in no particular order. A linear search is used to find the record for one parcel.

(a) State the greatest number of records the linear search might need to check. **[1]**

(b) Describe the **two** situations in which the linear search would need to check this many records. **[2]**

1

2

Total for this worksheet: 21 marks

MODULE L STANDARD ALGORITHMS · L2

Binary search

Original practice questions · 21 marks · ~25 minutes · OCR J277 spec 2.1.3

Name _____ Date _____

Instructions. Answer all questions in the spaces provided. The marks for each part are shown in brackets [ ]. The total for this worksheet is **21 marks**. Code on this sheet is written in OCR Exam Reference Language. You do **not** need to remember the code for a binary search.

1

Total: 4 marks

A garden centre's computer system holds three lists.

(a) Tick (✓) **one** box in each row to show whether a binary search could be used on the list **as it is now**. **[3]**

List	Yes	No
`4, 9, 15, 22, 31, 48`		
`"Ash", "Elm", "Fir", "Oak", "Pine", "Yew"`		
`12, 5, 18, 3, 27, 30`		

(b) Give the condition that a list must meet before a binary search can be used on it. **[1]**

2 **Total: 7 marks**

A warehouse robot keeps the codes of the parcels on its shelf in a sorted list:

```
103   118   125   140   152   167   171   186   194   205   219
```

When a list, or the part of a list still being searched, has an even number of items, the robot takes the item to the **left** of the centre as the middle item.

(a) Show how a binary search finds parcel **205** in the list. At each stage, give the middle item that is checked and the items that are discarded. **[4]**

(b) The robot then uses a binary search to look for parcel **130**, which is not in the list.

(i) Give the parcel codes that are checked, in the order they are checked. **[2]**

(ii) State how the binary search shows that parcel 130 is not in the list. **[1]**

3 **Total: 8 marks**

A sensor company stores the serial numbers of its devices in a sorted array called `codes`. The function `findCode()` looks for a serial number in the array. It returns the index where the number is stored, or -1. The array is indexed from 0.

```
01 function findCode(codes, target)
02     low = 0
03     high = codes.length - 1
04     while low <= high
05         mid = (low + high) DIV 2
06         if codes[mid] == target then
07             return mid
08         elseif codes[mid] < target then
09             low = mid + 1
10         else
11             high = mid - 1
12         endif
13     endwhile
14     return -1
15 endfunction
```

(a) Identify the searching algorithm used in `findCode()`. Give **one** reason from the code for your answer. **[2]**

Algorithm:

Reason:

(b) The array `codes` holds these values:

Index	0	1	2	3	4	5	6	7
Value	12	19	23	37	41	56	64	78

Complete the trace table for the function call `findCode(codes, 23)`. You may not need to use all the rows. **[4]**

low	high	mid	Value returned

(c) State the purpose of line 09. **[1]**

(d) State what it means when the function returns -1 at line 14. **[1]**

4 **Total: 2 marks**

A city library's system stores 100 000 book records, sorted by book number.

(a) Explain why a binary search would usually find a book record using far fewer comparisons than a linear search. **[2]**

Total for this worksheet: 21 marks

MODULE L STANDARD ALGORITHMS · L3

Bubble sort

Original practice questions · 21 marks · ~25 minutes · OCR J277 spec 2.1.3 (with 2.1.2 trace tables)

Name _____ Date _____

Instructions. Answer all questions in the spaces provided. The marks for each part are shown in brackets [ ]. The total for this worksheet is **21 marks**. All code is written in OCR Exam Reference Language. Arrays start at index 0.

1

Total: 4 marks

(a) Tick (✓) **one** box in each row to show whether each statement about a bubble sort into ascending order is true or false. **[4]**

Statement	True	False
It compares items that are next to each other in the list.		
It can only be used if the data is already partly in order.		
At the end of the first pass, the largest value is in its final position.		
It splits the list into sub-lists that each hold one item.		

2

Total: 5 marks

A café records the number of orders it takes in each hour of one morning:

```
18  7  25  12  9  30
```

The manager uses a bubble sort to put the numbers into ascending order. The bubble sort stops when a pass makes no swaps.

(a) Complete the table to show the order of the numbers after each of the first three passes. **[3]**

	Numbers					
Start	18	7	25	12	9	30
After pass 1						
After pass 2						
After pass 3						

(b) The numbers are in order after pass 3. Explain why the bubble sort still carries out a fourth pass. **[2]**

3 **Total: 5 marks**

A market stall keeps the prices of its items in the array `prices`. The algorithm below puts the prices in order.

```
01 do
02     swapped = false
03     for i = 0 to prices.length - 2
04         if prices[i] > prices[i + 1] then
05             temp = prices[i]
06             prices[i] = prices[i + 1]
07             prices[i + 1] = temp
08             swapped = true
09         endif
10     next i
11 until swapped == false
```

(a) Give **two** features of this code that show it is a bubble sort. **[2]**

1

2

(b) Describe the purpose of the variable `swapped` in this algorithm. **[2]**

(c) The stall owner wants the prices sorted into **descending** order instead. Rewrite line 04 so that the algorithm does this. **[1]**

04

4 **Total: 7 marks**

A weather station stores four wind speed readings. The code below carries out **one pass** of a bubble sort and then outputs a value.

```
01 array speeds = [14, 9, 21, 6]
02 swaps = 0
03 for i = 0 to 2
04     if speeds[i] > speeds[i + 1] then
05         temp = speeds[i]
06         speeds[i] = speeds[i + 1]
07         speeds[i + 1] = temp
08         swaps = swaps + 1
09     endif
10 next i
11 print(swaps)
```

(a) Complete the trace table for this code. You may not need to use all the rows. **[4]**

i	temp	swaps	Output

(b) State the contents of the array `speeds` after the code has finished. **[1]**

(c) Explain why lines 05 to 07 use the variable `temp` to swap two items. **[2]**

Total for this worksheet: 21 marks

MODULE L STANDARD ALGORITHMS · L4

Merge sort

Original practice questions · 20 marks · ~25 minutes · OCR J277 spec 2.1.3

Name _____ Date _____

Instructions. Answer all questions in the spaces provided. The marks for each part are shown in brackets [ ]. The total for this worksheet is **20 marks**.

1

Total: 3 marks

(a) Tick (✓) **one** box in each row to show whether each statement about a merge sort is true or false. **[3]**

Statement	True	False
A list that contains only one item is already sorted.		
A merge sort can only be used if the data is already in order.		
When two lists are merged, they are joined end to end and the new list is then sorted.		

2

Total: 2 marks

(a) A merge sort has two main stages. Describe the **two** main stages of a merge sort. **[2]**

1

2

3 **Total: 4 marks**

A parcel depot weighs eight parcels before loading a van. The heaviest parcels must be loaded first. The weights, in kg, are stored in this order:

9	27	3	14	21	5	11	18

(a) Show the steps that a merge sort would take to put these weights into **descending** order (largest first). **[4]**

4 **Total: 4 marks**

A wildlife trust stores the names of animals it has recorded. Part way through a merge sort, these two lists are both in alphabetical order and are about to be merged into one list in alphabetical order.

Left list	Badger	Heron	Otter	Wren
Right list	Crane	Fox	Stoat	Vole

(a) Complete the table to show the next **three** comparisons made during the merge, and the item that is moved into the new list after each one. The first comparison has been done for you. **[3]**

Comparison	Items compared	Item moved into the new list
1	*Badger and Crane*	*Badger*
2		
3		
4		

(b) Give the new list when the merge is complete. **[1]**

5 **Total: 5 marks**

A programmer at the parcel depot writes this function. It is given two lists, `left` and `right`, which are each already in ascending order. Arrays are indexed from 0.

```
01 function combine(left, right)
02     array result[left.length + right.length]
03     i = 0
04     j = 0
05     k = 0
06     while i < left.length AND j < right.length
07         if left[i] <= right[j] then
08             result[k] = left[i]
09             i = i + 1
10         else
11             result[k] = right[j]
12             j = j + 1
13         endif
14         k = k + 1
15     endwhile
16     while i < left.length
17         result[k] = left[i]
18         i = i + 1
19         k = k + 1
20     endwhile
21     while j < right.length
22         result[k] = right[j]
23         j = j + 1
24         k = k + 1
25     endwhile
26     return result
27 endfunction
```

(a) Identify the sorting algorithm that this function is part of. **[1]**

(b) Describe the purpose of lines 16 to 25. **[2]**

(c) The function is called with `left = [4, 9, 15]` and `right = [2, 10]`.

(i) State the contents of `result` when the function returns. **[1]**

(ii) State how many times line 07 is run during this call. **[1]**

Answer: .....

6 **Total: 2 marks**

At the end of each day the depot sorts a list of about 50 000 parcel codes. It could use a bubble sort or a merge sort.

(a) Give **one** advantage of using a merge sort instead of a bubble sort for this list. **[1]**

(b) Give **one** disadvantage of using a merge sort instead of a bubble sort. **[1]**

Total for this worksheet: 20 marks

MODULE L STANDARD ALGORITHMS · L5

Insertion sort

Original practice questions · 20 marks · ~25 minutes · OCR J277 spec 2.1.3

Name _____ Date _____

Instructions. Answer all questions in the spaces provided. The marks for each part are shown in brackets [ ]. The total for this worksheet is **20 marks**. Code is written in OCR Exam Reference Language. Arrays start at index 0.

1

Total: 4 marks

(a) While an insertion sort is running, the list is made up of two parts. State what the two parts are. **[1]**

(b) Describe the main steps of an insertion sort that puts a list into ascending order. **[3]**

2

Total: 4 marks

A delivery depot weighs five parcels. The weights, in kg, are stored in a list in the order they arrive.

(a) Complete the table to show the list after each item is inserted when an insertion sort puts the weights into ascending order. The starting list has been done for you. **[4]**

	List				
Start	31	18	42	7	25
After insertion 1					
After insertion 2					
After insertion 3					
After insertion 4					

3 **Total: 4 marks**

A garden centre's stock program uses an insertion sort to put plant names into alphabetical order. Part of the way through the sort, the list is:

```
Aster   Fern   Mint   Iris   Basil   Tulip
```

The first three items (Aster, Fern, Mint) are in the sorted part.

(a) State the next item that will be inserted into the sorted part. **[1]**

(b) Describe how the item you gave in part (a) is moved into its correct position. **[2]**

(c) The sort continues. Write the list after **Basil** has been inserted into the sorted part. **[1]**

4 **Total: 8 marks**

A sports centre records the times, in seconds, that four members take to swim one length. The algorithm below puts the times into ascending order.

```
01 times = [48, 35, 52, 29]
02 for i = 1 to times.length - 1
03     current = times[i]
04     j = i - 1
05     while j >= 0 AND times[j] > current
06         times[j + 1] = times[j]
07         j = j - 1
08     endwhile
09     times[j + 1] = current
10 next i
```

(a) Identify the sorting algorithm used in this code. **[1]**

(b) Identify the line number where `current` is placed into its correct position in the sorted part. **[1]**

(c) Describe what lines 05 to 08 do. **[2]**

(d) Complete the trace table to show the values of `i`, `current` and `j` as the algorithm runs. Start a new row each time a value changes. You may not need to use all the rows. **[4]**

i	current	j

Total for this worksheet: 20 marks

MODULE L STANDARD ALGORITHMS · L6

Comparing the search and sort algorithms

Original practice questions · 22 marks · ~25 minutes · OCR J277 spec 2.1.3

Name _____ Date _____

Instructions. Answer all questions in the spaces provided. The marks for each part are shown in brackets [ ]. The total for this worksheet is **22 marks**.

1

Total: 3 marks

(a) Tick (✓) **one** box in each row to show whether each statement is true for a **linear search** only, a **binary search** only, or **both**. **[3]**

Statement	Linear search only	Binary search only	Both
It can search a list that has not been sorted			
It compares the target with the middle item of the items that remain			
It can search a list that is in ascending order			

2

Total: 3 marks

(a) Tick (✓) **one** box in each row to show which sorting algorithm each statement describes. **[3]**

Statement	Bubble sort	Insertion sort	Merge sort
Of these three, it usually needs the most extra memory, because it creates new lists as it works			
It can stop early if it completes a pass through the list without making any swaps			
It takes each item in turn and inserts it into the correct place among the items before it			

3

Total: 3 marks

A parcel depot's system uses the function `findCode()` to look for a locker number (`target`) in the array `codes`. It is written in OCR Exam Reference Language.

```
01 function findCode(codes, target)
02     low = 0
03     high = codes.length - 1
04     while low <= high
05         mid = (low + high) DIV 2
06         if codes[mid] == target then
07             return mid
08         elseif codes[mid] < target then
09             low = mid + 1
10         else
11             high = mid - 1
12         endif
13     endwhile
14     return -1
15 endfunction
```

(a) Identify the searching algorithm used in `findCode()`. **[1]**

(b) State the pre-requisite that the array `codes` must meet for this algorithm to work. **[1]**

(c) Identify the line number of the line that works out the position of the middle item. **[1]**

4 **Total: 6 marks**

The array `codes` holds the 15 locker numbers that are in use, in ascending order. The first item is at position 0.

```
3   8   11   15   19   24   27   31   36   40   44   52   58   63   70
```

(a) A **linear search** is used to look for the locker number 58. State how many items the linear search checks to find 58, including 58 itself. **[1]**

(b) The function `findCode()` from Question 3 is used to look for 58 instead. Show the steps of the binary search, giving each item that is compared with 58 and the items that are discarded. **[3]**

(c) The depot then looks for the locker number 8. Explain why a linear search would find 8 using fewer comparisons than the binary search in `findCode()`. **[2]**

5 **Total: 2 marks**

A museum stores the badge numbers of the 25 visitors who are inside the building, in the order they arrived. The list is not sorted, and badge numbers are added and removed all day.

(a) Give **two** reasons why a linear search is more suitable than a binary search for checking whether a badge number is in this list. **[2]**

1

2

6 **Total: 5 marks**

(a) A weather service sorts about 2 million temperature readings every night. The sort runs on a large server that has plenty of spare memory. The service uses a merge sort. **[2]**
Explain why a merge sort is a suitable choice for this task.

(b) A small battery-powered sensor has very little memory. Every minute it sorts its last 20 readings, which are nearly always already in order. **[3]**
Recommend a sorting algorithm for the sensor. Justify your choice.

Algorithm

Total for this worksheet: 22 marks

MODULE L STANDARD ALGORITHMS · L7

Totalling, counting, maximum, minimum and average

Original practice questions · 21 marks · ~25 minutes · OCR J277 spec 2.1.2, 2.2.1, 2.2.3

Name _____ Date _____

Instructions. Answer all questions in the spaces provided. The marks for each part are shown in brackets [ ]. The total for this worksheet is **21 marks**. All code on this sheet is written in OCR Exam Reference Language. Arrays are indexed from 0.

1

Total: 3 marks

A café stores the number of cakes it sold each month in the array `sales`. The array holds **12** values. This algorithm should calculate the mean number of cakes sold per month.

```
01 total = ..........
02 for i = 0 to sales.length - 1
03     total = total .......... sales[i]
04 next i
05 average = total / ..........
06 print(average)
```

(a) Complete the algorithm by filling in the **three** gaps. Write your answers below. **[3]**

Line 01 .....

Line 03 .....

Line 05 .....

2

Total: 6 marks

A weather station stores the rainfall, in mm, for five days in the array `rain`. This program finds the wettest day's rainfall and counts the dry days (days with 0 mm).

```
01 wettest = rain[0]
02 dryDays = 0
03 for day = 1 to 4
04     if rain[day] > wettest then
05         wettest = rain[day]
06     endif
07     if rain[day] == 0 then
08         dryDays = dryDays + 1
09     endif
10 next day
11 print(wettest)
12 print(dryDays)
```

(a) Complete the trace table when `rain = [4, 12, 0, 15, 9]` **[4]**

You may not need to use all the rows.

day	wettest	dryDays	Output

(b) In a different week, `rain = [0, 3, 0, 7, 2]`. There were two dry days, but the program outputs `1` for `dryDays`. **[2]**

Explain why the program gives the wrong number of dry days, and refine line 03 so that it counts correctly.

Refined line 03 .....

3 **Total: 4 marks**

A plant nursery measures 30 seedlings. Their heights, in cm, are stored in the array `heights` (indexes 0 to 29).

(a) Write an algorithm that finds and outputs the **smallest** height in the array. **[4]**
Do **not** use a built-in function such as `min()`.
You must use either: OCR Exam Reference Language, or a high-level programming language that you have studied.

4 **Total: 2 marks**

A freezer records 24 temperatures, in °C, in the array `temps`. Every temperature is below 0 °C, for example `-18`, `-21` and `-16`. A programmer starts the search for the highest temperature like this:

```
01 highest = 0
02 for t = 0 to 23
03     if temps[t] > highest then
04         highest = temps[t]
05     endif
06 next t
07 print(highest)
```

(a) Explain why this program will **not** output the highest temperature in the array. **[1]**

(b) Refine line 01 so that the program works for any set of temperatures. **[1]**

Line 01 .....

5 **Total: 6 marks**

A bicycle repair workshop wants a program to summarise each day's work.

(a) Write a program that: **[6]**

- asks the user how many repairs were completed today
- asks the user to enter the time, in minutes, of each repair
- outputs the total time of all the repairs
- outputs how many repairs took **more than** 45 minutes
- outputs the mean (average) time of a repair.

You must use either: OCR Exam Reference Language, or a high-level programming language that you have studied.

Total for this worksheet: 21 marks