2.3 ALGORITHMS · 2.3.1(F)

Merge & quick sort — Mark scheme

30 marks · spec 2.3.1(f)

AO key: AO1 = knowledge · AO2 = application. Traces verified. Credit any valid equivalent.

Q	ANSWER	AO	MARKS
1(a)	Divide: split the list in half repeatedly down to single items (1); conquer: merge sorted sub-lists back together (1). (2)	AO1	2
1(b)	O(n log n). (1)	AO1	1
1(c)	It divides the problem into smaller sub-problems of the same type (sub-lists) (1); solves them and combines (merges) the results (1). (2)	AO1	2
1(d)	It is not in place — it needs extra memory for the sub-lists. (1)	AO1	1

Q	ANSWER	AO	MARKS
2(a)	Split: [8,3,5,1] → [8,3] & [5,1] → [8][3][5][1] (2); merge: [3,8] & [1,5] → [1,3,5,8] (2). (4)	AO2	4
2(b)	[1, 3, 5, 8]. (1)	AO2	1
2(c)	[1, 3, 5, 8]. (1)	AO2	1

Q	ANSWER	AO	MARKS
3(a)	Left (smaller): [2, 1] (1); right (larger): [6, 8] (1). (2)	AO2	2
3(b)	The pivot (4) is now in its final/correct sorted position (1); everything left is smaller and everything right is larger (1). (2)	AO2	2
3(c)	It recursively quick sorts the left part (1); and the right part, the same way (1). (2)	AO1	2

Q	ANSWER	AO	MARKS
4(a)	O(n log n). (1)	AO1	1
4(b)	$O(n^2)$ (1); when the pivot is consistently poorly chosen (e.g. smallest/largest each time, such as an already-sorted list) (1). (2)	AO1	2
4(c)	Any two: merge divides by position / quick by value (pivot) · merge always O(n log n) / quick can be $O(n^2)$ · merge needs extra memory / quick can be in place. (2)	AO1	2
4(d)	Quick sort. (1)	AO1	1

Q	ANSWER	AO	MARKS
5(a)	Merge/quick are O(n log n) but bubble is O(n^2) (1); for a million items that's ~20 million operations vs ~10^{12} (1); so the O(n log n) sorts are vastly faster on large data (1). (3)	AO2	3
5(b)	Quick sort (1); it can sort in place, using little extra memory, whereas merge sort needs extra lists (1). (2)	AO2	2
5(c)	3 ($\log_2 8$). (1)	AO2	1

Total for paper: 30 marks