

Topic 4 · Sheet A: Worked answers and mark scheme

Bubblesort: correctness and running time

Suggested time

About 75 minutes

Total marks

73

Questions

12

Builds on

Topics 1 to 3

How marks are awarded. Each mark has a code, as on most UK exam papers.

B A mark for a correct statement or value on its own. It does not depend on any other mark.

M A method mark, for a correct approach, even if the final answer is wrong.

A An accuracy mark, for the correct answer *after* a correct method. It cannot be awarded without the M mark before it.

E An explanation mark, for a clear reason in words. A correct reason that is worded differently scores in full.

For true or false questions, the mark is for the correct decision; where a reason is asked for, it must be the right reason.

1 One outer iteration

4 marks

- (a) $j = 5$: $1 < 3$, no swap. $j = 4$: $5 > 1$, swap to $[4, 2, 1, 5, 3]$. $j = 3$: $2 > 1$, swap to $[4, 1, 2, 5, 3]$.
 $j = 2$: $4 > 1$, swap. Result $[1, 4, 2, 5, 3]$.
- (b) **4**, one for each $j = 5, 4, 3, 2$.
- (c) **3**.

M1	(a) comparing from the right-hand end	1
A1	(a)	1
B1	(b)	1
B1	(c)	1

Common mistake Starting the inner loop from the left. In this version j runs *down* from n , so the smallest entry bubbles to the front.

2 Finishing the run

4 marks

- (a) $[1, 2, 4, 3, 5]$. (b) $[1, 2, 3, 4, 5]$. (c) $4 + 3 + 2 + 1 = 10$. (d) $3 + 2 + 1 + 0 = 6$.

B1 each part correct

4

Why this question The array is sorted after iteration 3, but the algorithm still runs iteration 4 and makes its comparison. Plain Bubblesort never stops early.

3 The invariant

6 marks

- (a) At the start of iteration i , $A[1] < A[2] < \dots < A[i-1]$, **and these are the $i-1$ smallest entries**: $A[i-1] < A[k]$ for every $k \geq i$.
- (b) At the start of $i = 3$ the array is $[1, 2, 4, 3, 5]$. Then $A[1] < A[2]$, since $1 < 2$, and 1, 2 are the two smallest values. **The invariant holds**.
- (c) After iteration $n-1$ (equivalently, at the start of a would-be iteration n), $A[1] < \dots < A[n-1]$ and $A[n-1] < A[n]$. So **the whole array is sorted**.

B1	(a) first $i-1$ entries sorted	1
B1	(a) and they are the smallest	1
M1	(b) checking both parts	1
A1	(b)	1
M1	(c) invariant with $i = n$	1
E1	(c) $A[n]$ is then the largest	1

Common mistake Stating only “the first $i-1$ entries are sorted”. That alone does not give a sorted array at the end; you also need them to be the *smallest* entries.

4 Counting comparisons

6 marks

- (a) j runs from n down to $i+1$: $n-i$.
- (b) $\sum_{i=1}^{n-1} (n-i) = (n-1) + \dots + 1 = \frac{n(n-1)}{2}$.
- (c) $\Theta(n^2)$.
- (d) $n = 10$: 45. $n = 100$: 4950.

B1	(a)	1
M1	(b) the sum	1
A1	(b)	1
B1	(c)	1
B1	(d) each value	2

Why this question Multiplying n by 10 multiplies the work by about 100. That is what $\Theta(n^2)$ means in practice.

5 Counting swaps

6 marks

- (a) **0**.
- (b) Every comparison finds a pair out of order, so $\frac{n(n-1)}{2}$ swaps.
- (c) The comparisons on line 3 happen **whatever the data**, and there are $\frac{n(n-1)}{2}$ of them.
- (d) Swapping $A[j-1]$ and $A[j]$ only changes the relative order of these two entries, which were an inversion and now are not. Every other pair keeps its relative order. So each swap removes **exactly one inversion**. The run ends sorted, with no inversions, so the number of swaps **equals the number of inversions** at the start.

B1	(a)	1
M1	(b) every comparison swaps	1
A1	(b)	1
E1	(c)	1
E1	(d) only the swapped pair changes order	1
E1	(d) sorted array has no inversions	1

Why this question A sorted input needs no swaps but still costs $\Theta(n^2)$. Comparisons and swaps measure different things.

6 The smallest entry

6 marks

- (a) Once j reaches the position of the minimum, every later comparison finds $A[j-1] > A[j]$, because the minimum is now at $A[j]$. So it is swapped one place left at every step, and **ends at position 1**.
- (b) It moves from position k to position 1, one place per swap: **$k-1$ swaps**.
- (c) In iteration i , $j \geq i+1$, so the entries compared are $A[j-1]$ and $A[j]$ with $j-1 \geq i$. **Positions $1, \dots, i-1$ are never touched**.

M1	(a) the minimum is reached by the inner loop	1
E1	(a) it then loses every comparison	1
M1	(b) moves one place per swap	1
A1	(b)	1
M1	(c) smallest index used is $j-1 \geq i$	1
A1	(c)	1

Why this question These two facts together are exactly the inductive step that proves the invariant.

7 Stopping early

6 marks

- (a) No swap means $A[j-1] < A[j]$ for every j from $i+1$ to n , so $A[i], \dots, A[n]$ are in order. With the invariant, the first $i-1$ entries are sorted and smaller than these, so **the whole array is already sorted**.
- (b) The first iteration makes $n-1$ comparisons, finds no swap, and stops. So the best case is $\Theta(n)$.
- (c) Still $\Theta(n^2)$. A **decreasing array** makes a swap in every iteration, so all $\frac{n(n-1)}{2}$ comparisons happen.

E1	(a) remaining part in order	1
E1	(a) combined with the invariant	1
M1	(b)	1
A1	(b)	1
B1	(c) $\Theta(n^2)$	1
B1	(c) an input	1

Common mistake Claiming the variant is now $\Theta(n)$. Stopping early improves the *best* case. The running time means the worst case, and that does not change.

8 The other direction

6 marks

- (a) $j = 1: 3 > 1$, swap to $[1, 3, 4, 2]$. $j = 2: 3 < 4$. $j = 3: 4 > 2$, swap. Result $[1, 3, 2, 4]$.
- (b) At the start of iteration i , **the last $i-1$ entries are sorted and are the $i-1$ largest**.
- (c) Iteration i makes $n-i$ comparisons, so the total is again $\frac{n(n-1)}{2}$.

M1	(a)	1
A1	(a)	1
B1	(b) sorted end part	1
B1	(b) largest entries	1
M1	(c)	1
A1	(c)	1

Why this question The two versions are mirror images. Writing the invariant for a new version is good practice for any loop you meet.

9 Trying every ordering

6 marks

- (a) $n!$.
- (b) In the worst case the sorted order is tried last: $n!(n-1)$ comparisons.
- (c) $10! \cdot 9 = 32\,659\,200$, against **45** for Bubblesort.
- (d) $n! = \omega(n^2)$.

B1	(a)	1
M1	(b)	1
A1	(b)	1
B1	(c) each number	2
B1	(d)	1

Why this question “Try everything” is always correct but usually useless. Even a simple quadratic algorithm is vastly better.

10 Repeated entries

7 marks

- (a) $i = 1$: $j = 4$ swaps to $[2, 1, 1, 2]$; $j = 3$: $1 > 1$ is false; $j = 2$ swaps. $[1, 2, 1, 2]$. $i = 2$: $[1, 1, 2, 2]$.
 $i = 3$: $[1, 1, 2, 2]$.
- (b) $2 + 1 + 0 = 3$ swaps. The inversions are the pairs of positions $(1, 2)$, $(1, 4)$ and $(3, 4)$: also **3**.
- (c) The test is $A[j-1] > A[j]$, which is false for equal entries. So **two equal entries are never swapped with each other**, and only adjacent entries move past each other.

M1	(a)	1
A1	(a) after $i = 1$	1
A1	(a) after $i = 2$ and $i = 3$	1
B1	(b) swaps	1
B1	(b) inversions	1
E1	(c) strict test	1
E1	(c) only adjacent swaps	1

Why this question A sort that keeps equal entries in their original order is called stable. It matters when you sort records by one field after sorting by another.

11 Exam style: a full trace

8 marks

- (a) After $i = 1$: $[1, 5, 3, 8, 2, 9]$. After $i = 2$: $[1, 2, 5, 3, 8, 9]$.
- (b) $5 + 4 + 3 + 2 + 1 = 15$.

- (c) 5 beats 3, 1, 2; 3 beats 1, 2; 8 beats 1, 2; 9 beats 2. That is $3 + 2 + 2 + 1 = 8$ inversions, so **8 swaps**.
- (d) $A[1] = 1$ and $A[2] = 2$. By the invariant, after two iterations the first two positions hold the two smallest entries in order. (Here 9 also happens to be in place, but the invariant does not promise that.)

M1	(a)	1
A1	(a) after $i = 1$	1
A1	(a) after $i = 2$	1
B1	(b)	1
M1	(c) counting inversions	1
A1	(c)	1
B1	(d)	1
E1	(d) via the invariant	1

Common mistake Counting the swaps by running the whole sort. Counting inversions is much faster, and it is a check on your trace.

12 Exam style: nearly sorted inputs

8 marks

- (a) P has inversions only with its last entry 1: $n - 1$. Q has inversions only with its first entry n : also $n - 1$.
- (b) In iteration 1 the entry 1 bubbles all the way to the front, and P is sorted. Iteration 2 makes no swap, so the variant stops. Comparisons: $(n - 1) + (n - 2) = 2n - 3$.
- (c) Each iteration moves n only one place to the right (only the comparison at $j = i + 1$ swaps). So every one of the $n - 1$ iterations swaps, the variant never stops early, and it makes all $\frac{n(n-1)}{2}$ comparisons.
- (d) Bubblesort moves a **small entry many places left in one iteration**, but a **large entry only one place right per iteration**. P needs a small entry to move left; Q needs a large one to move right.

B1	(a) each array	2
M1	(b) sorted after one iteration	1
A1	(b)	1
M1	(c) one swap per iteration	1
A1	(c)	1
E2	(d) the asymmetry between moving left and right	2

Why this question Both arrays are “one entry out of place”, yet one costs $\Theta(n)$ and the other $\Theta(n^2)$. The number of swaps is the same; the number of iterations is not.

Total: 73 marks. Spotted a mistake? Report it at unimixedtutor.com and we will fix it.