

Topic 3 · Sheet A: Worked answers and mark scheme

Analysing the runtime of an algorithm

Suggested time

About 75 minutes

Total marks

73

Questions

12

Builds on

Topics 1 to 2

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 Tracing ZEROPAIR

4 marks

(a) $3 + (-3) = 0$ at once: **1**. (b) no pair works, so all $\binom{4}{2}$: **6**. (c) $i = 1$ tries $j = 2, 3, 4$ (three sums), then $i = 2, j = 3$ gives 0: **4**. (d) $\frac{n(n-1)}{2}$.

B1 each part correct

4

Why this question The same algorithm takes very different times on inputs of the same size. That is why we need a single measure, the worst case.

2 Finding the maximum

4 marks

(a) $n - 1$, one for each $i = 2, \dots, n$.

(b) Increasing: every new entry is bigger, so $n - 1$ **times**. Decreasing: **never**.

(c) $\Theta(n)$.

B1 (a) 1

B1 (b) increasing 1

B1 (b) decreasing 1

B1 (c) 1

Why this question The number of comparisons never changes, even though the number of updates does. The comparisons are what fix the running time here.

3 Counting with constants

6 marks

- (a) Line 1: **once**. Line 2: n **times**. Lines 3 and 4: n^2 **times** each. Line 5: **once**.
- (b) $f(n) = c_1 + c_2n + (c_3 + c_4)n^2 + c_5$.
- (c) $f(n)$ is a quadratic in n with positive leading coefficient $c_3 + c_4$, so it is $\Theta(n^2)$ for any positive constants.

B1	(a) lines 1 and 5	1
B1	(a) line 2	1
B1	(a) lines 3 and 4	1
M1	(b) multiplying counts by costs	1
A1	(b)	1
E1	(c)	1

Why this question This is the whole idea of asymptotic running time: the constants depend on the machine and the language, but $\Theta(n^2)$ does not.

4 Best case and worst case

6 marks

- (a) Any array with $A[1] = \alpha$.
- (b) k .
- (c) n .
- (d) $\Theta(n)$.
- (e) The running time is defined as the **maximum over all inputs of size n** . Some inputs need n comparisons, so the worst case is $\Theta(n)$, whatever happens on easy inputs.

B1	(a)	1
B1	(b)	1
B1	(c)	1
B1	(d)	1
E2	(e) worst case is the maximum over inputs, and some inputs need $\Theta(n)$	2

Common mistake Giving the best case as the running time. "Runtime" in this module always means the worst case.

5 Three nested loops

6 marks

- (a) The triples $i < j < k$ from $\{1, 2, 3, 4\}$: **4**.
- (b) Line 4 runs once for each triple $i < j < k$ chosen from $\{1, \dots, n\}$, so the count is $\binom{n}{3} = \frac{n(n-1)(n-2)}{6}$.
- (c) $\Theta(n^3)$.
- (d) For $n \geq 4$, $n-1 \geq \frac{n}{2}$ and $n-2 \geq \frac{n}{2}$, so the count is at least $\frac{n \cdot \frac{n}{2} \cdot \frac{n}{2}}{6} = \frac{n^3}{24}$.

B1	(a)	1
M1	(b) one step for each triple $i < j < k$	1
M1	(b) counting the triples (or summing the loops)	1
A1	(b)	1
B1	(c)	1
B1	(d)	1

Why this question You can also add up the loops: $\sum_i \sum_{j>i} (n-j)$. Recognising the count as “choose 3” is much quicker.

6 Checking for repeats

6 marks

- (a) Arrays with **all entries different**: no **return NO** happens, so every pair is compared: $\frac{n(n-1)}{2}$ comparisons.
- (b) Any array with $A[1] = A[2]$.
- (c) $\Theta(n^2)$.
- (d) For example $A[1] = A[n]$ **with all other entries different**. With $i = 1$ it compares $A[1]$ with $A[2], \dots, A[n]$ and stops at $j = n$: $n-1$ comparisons.

B1	(a) all distinct	1
B1	(a) count	1
B1	(b)	1
B1	(c)	1
B1	(d) an input	1
E1	(d) checking it gives $n-1$	1

Why this question For algorithms that can stop early, the worst inputs are the ones where the “bad event” never happens. Here that is an array with no repeats.

7 A doubling loop

6 marks

- (a) $n = 1$: $i = 1$, so **1**. $n = 8$: $i = 1, 2, 4, 8$, so **4**. $n = 10$: also $i = 1, 2, 4, 8$, so **4**.
- (b) After t iterations $i = 2^t$. The body runs for $i = 2^0, \dots, 2^{\lfloor \log_2 n \rfloor}$: $\lfloor \log_2 n \rfloor + 1$ times.
- (c) $\Theta(\log n)$.
- (d) It runs $\lfloor \log_3 n \rfloor + 1$ times, which is **still** $\Theta(\log n)$, since logarithms to different bases differ by a constant factor.

B1	(a) $n = 1$ and $n = 8$	1
B1	(a) $n = 10$	1
M1	(b) $i = 2^t$ after t iterations	1
A1	(b)	1
B1	(c)	1
B1	(d)	1

Common mistake Writing $\log_2 n$ iterations. That is not even an integer when $n = 10$. Check small cases: the answer is $\lfloor \log_2 n \rfloor + 1$.

8 Steps of size i

6 marks

- (a) j takes the values $i, 2i, 3i, \dots \leq n$, so the body runs $\lfloor n/i \rfloor$ times. $i = 1$: n . $i = 2$: $\lfloor n/2 \rfloor$.
 $i = n$: 1.
- (b) $\sum_{i=1}^n \lfloor n/i \rfloor$.
- (c) This lies between $n \sum \frac{1}{i} - n$ and $n \sum \frac{1}{i}$, and $\sum_{i=1}^n \frac{1}{i} = \Theta(\log n)$. So the running time is $\Theta(n \log n)$.

B1	(a) each count	3
B1	(b)	1
M1	(c) comparing with the harmonic sum	1
A1	(c)	1

Why this question This loop shape is common: process every multiple of every i . It costs only a factor $\log n$ more than a single pass.

9 Prefix sums

6 marks

- (a) $B[k]$ takes $k - 1$ additions, so the total is $\sum_{k=1}^n (k - 1) = \frac{n(n-1)}{2}$: $\Theta(n^2)$.
 (b) For example:

```

1  B[1] ← A[1]
2  for k = 2, ..., n do B[k] ← B[k - 1] + A[k]
3  return B
  
```

- (c) The loop runs $n - 1$ times with one addition each time, so the total time is $\Theta(n)$.
 (d) About 500 000 additions for the first, and 999 for the second.

M1	(a) the sum	1
A1	(a)	1
M1	(b) reusing $B[k - 1]$	1
A1	(b) correct pseudocode	1
E1	(c)	1
B1	(d)	1

Why this question Reusing work you have already done is the simplest way to beat a quadratic algorithm.

10 Two indices on a sorted array

7 marks

- (a) $(1, 6)$: $-7 + 6 < 0$, so $\ell = 2$. $(2, 6)$: $-3 + 6 > 0$, so $r = 5$. $(2, 5)$: $-3 + 3 = 0$. **Pairs $(1, 6)$, $(2, 6)$, $(2, 5)$; output YES.**
 (b) Each iteration that does not return either increases ℓ or decreases r , so $r - \ell$ drops by 1. It starts at $n - 1$ and the loop stops when it reaches 0: **at most $n - 1$ iterations, $\Theta(n)$ time.**
 (c) SORTEDZEROPAIR: at most 999 sums. ZEROPAIR: up to $\binom{1000}{2} = 499\,500$.
 (d) If $A[\ell] + A[r] < 0$, sortedness means $A[\ell]$ plus *any* entry up to $A[r]$ is also negative, so $A[\ell]$ **can be discarded safely**. Without sorting this reasoning fails.

B1	(a) the pairs	1
B1	(a) output	1
E1	(b) $r - \ell$ decreases each time	1
B1	(b) $\Theta(n)$	1
B1	(c) both numbers	2
E1	(d)	1

Common mistake Arguing that “the loop could go on for a long time”. Find a quantity that drops every iteration; here it is $r - \ell$.

11 Exam style: what does it count?

8 marks

- (a) For each position i , it counts the earlier entries $A[j]$ (with $j < i$) that are larger than $A[i]$. So c is **the number of pairs $j < i$ with $A[j] > A[i]$** , the pairs that are out of order. ($j = i$ never counts.)
- (b) $\sum_{i=1}^n i = \frac{n(n+1)}{2}$.
- (c) $\Theta(n^2)$.
- (d) **Strictly decreasing arrays**: every pair $j < i$ counts, so $c = \frac{n(n-1)}{2}$.
- (e) **No**: there is no early **return**, so it always makes exactly $\frac{n(n+1)}{2}$ comparisons.

M1	(a) counting earlier larger entries	1
A1	(a) pairs out of order	1
M1	(b)	1
A1	(b)	1
B1	(c)	1
B1	(d) inputs	1
B1	(d) value	1
E1	(e)	1

Why this question Pairs that are out of order are called inversions. They measure how far an array is from sorted, and they come back when you study sorting.

12 Exam style: blocks one after another

8 marks

- (a) Block 1: n . Block 2: j takes $1, 2, 4, \dots$ while $j < n$, which is $\lceil \log_2 n \rceil$ values, so $n \lceil \log_2 n \rceil$. Block 3: $\lfloor \sqrt{n} \rfloor^2 \leq n$.
- (b) $\Theta(n \log n)$, decided by **block 2**. The others are $O(n)$.
- (c) The inner loop then runs about $\frac{n}{2}$ times, so block 2 costs about $\frac{n^2}{2}$ and the total becomes $\Theta(n^2)$.
- (d) Each block's count is multiplied by a fixed positive constant, and constant factors do not change a Θ class.

B1	(a) blocks 1 and 3	1
M1	(a) block 2 inner loop is logarithmic	1
A1	(a) block 2	1
B1	(b) $\Theta(n \log n)$	1
B1	(b) block 2	1
M1	(c) inner loop now linear	1
A1	(c)	1
E1	(d)	1

Why this question When blocks run one after another, their times add, and the largest one wins. When loops are nested, their counts multiply.

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