

Topic 6 · Sheet A: Worked answers and mark scheme

The Master Theorem and binary search

Suggested time

About 75 minutes

Total marks

73

Questions

12

Builds on

Topics 1 to 5

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 Reading off a, b, d

4 marks

(a) $a = 4, b = 2, d = 1$. (b) $a = 1, b = 3, d = 2$. (c) $a = 7, b = 2, d = 2$. (d) $a = 2, b = 4, d = \frac{1}{2}$.

B1 each part correct

4

Common mistake In (d), forgetting that $\sqrt{n} = n^{1/2}$, so $d = \frac{1}{2}$. The exponent d does not have to be a whole number.

2 Choosing the case

4 marks

(a) $\frac{4}{2} = 2 > 1$: $\Theta(n^{\log_2 4}) = \Theta(n^2)$. (b) $\frac{1}{9} < 1$: $\Theta(n^2)$. (c) $\frac{7}{4} > 1$: $\Theta(n^{\log_2 7})$, about $n^{2.81}$. (d) $\frac{2}{4^{1/2}} = 1$: $\Theta(\sqrt{n} \log n)$.

B1 each part correct

4

Why this question Always compute a/b^d first. Below 1 the top level wins, equal to 1 every level ties, above 1 the leaves win.

3 Three Master Theorem solutions

6 marks

(a) $a = 8, b^d = 4$, so $a > b^d$: $\Theta(n^{\log_2 8}) = \Theta(n^3)$.

(b) $a = 9, b^d = 9$, so $a = b^d$: $\Theta(n^2 \log n)$.

(c) $a = 3, b^d = 4$, so $a < b^d$: $\Theta(n)$.

M1	(a) comparing a with b^d	1
A1	(a)	1
M1	(b)	1
A1	(b)	1
M1	(c)	1
A1	(c)	1

Common mistake In (a), answering $\Theta(n^2)$ because of the n^2 term. When $a > b^d$ the answer does not involve d at all.

4 Where the theorem comes from

6 marks

(a) a^i calls, each of size n/b^i .

(b) Each call costs $(n/b^i)^d$, so the level costs $a^i \cdot \frac{n^d}{b^{id}} = n^d \left(\frac{a}{b^d}\right)^i$.

(c) $h = \log_b n$, since $n/b^h = 1$.

(d) If $a = b^d$, every one of the $h + 1 = \log_b n + 1$ levels costs exactly n^d , so the total is $n^d(\log_b n + 1)$.

B1	(a) calls	1
B1	(a) size	1
M1	(b) calls times cost per call	1
A1	(b)	1
B1	(c)	1
E1	(d)	1

Why this question The whole theorem is one geometric sum with ratio a/b^d . Knowing this, you never need to memorise the cases.

5 Tracing binary search

6 marks

- (a) $(1, 10)$, $m = 5$: $16 < 23$. $(6, 10)$, $m = 8$: $56 > 23$. $(6, 7)$, $m = 6$: $A[6] = 23$. **Returns 6.**
- (b) $(1, 10)$, $m = 5$; $(6, 10)$, $m = 8$; $(6, 7)$, $m = 6$: $23 < 50$; then $(7, 7)$: $A[7] = 38 \neq 50$. **Returns NO.**
- (c) **3**: 16, 56 and 23.

M1	(a) halving correctly	1
A1	(a) calls	1
A1	(a) output	1
B1	(b) calls	1
B1	(b) output	1
B1	(c)	1

Common mistake Recursing on (i, m) instead of $(i, m - 1)$. Position m has already been checked, so it is left out.

6 Binary search is logarithmic

6 marks

- (a) $T(n) = T(n/2) + \Theta(1)$.
- (b) $a = 1$, $b = 2$, $d = 0$, so $a/b^d = 1$: $\Theta(\log n)$.
- (c) The range halves each call, from 1000 down to 1: about $\log_2 1000 \approx 10$, so **about 10 calls**.
- (d) **1000**.

B1	(a)	1
M1	(b)	1
A1	(b)	1
M1	(c) halving	1
A1	(c)	1
B1	(d)	1

Why this question $d = 0$ because $\Theta(1) = \Theta(n^0)$. The middle case with $d = 0$ gives just $\log n$.

7 Multiplying large numbers

6 marks

- (a) $T(n) = 4T(n/2) + \Theta(n)$. $4 > 2$, so $\Theta(n^{\log_2 4}) = \Theta(n^2)$.
- (b) $T(n) = 3T(n/2) + \Theta(n)$. $3 > 2$, so $\Theta(n^{\log_2 3})$, about $n^{1.585}$.
- (c) $1024^2 = 1\,048\,576$, while $1024^{\log_2 3} = 3^{10} = 59\,049$, about 18 times smaller.

B1	(a) recurrence	1
B1	(a) solution	1
B1	(b) recurrence	1
B1	(b) solution	1
M1	(c) $1024^{\log_2 3} = 3^{10}$	1
A1	(c)	1

Why this question Saving one recursive call out of four changes the exponent, which matters far more than any constant factor.

8 When the theorem does not apply

6 marks

- (a) The size goes down by *subtracting* 1, not dividing by $b > 1$. Unrolling, $T(n) = T(1) + 2 + 3 + \dots + n$, which is $\Theta(n^2)$.
- (b) $n \log_2 n$ is not $\Theta(n^d)$ for any constant d . With $n = 2^k$, depth i has 2^i calls of size 2^{k-i} , each costing $2^{k-i}(k-i)$, so the level costs $n(k-i)$. Summing, $n(k + (k-1) + \dots + 1) = \frac{nk(k+1)}{2}$, so $T(n) = \Theta(n \log^2 n)$.
- (c) The number of subproblems 2^n **is not a constant**.

E1	(a) reason	1
A1	(a)	1
E1	(b) reason	1
M1	(b) cost per level	1
A1	(b)	1
E1	(c)	1

Common mistake In (b), forcing the theorem with $d = 1$ and getting $\Theta(n \log n)$. The extra $\log n$ on every level gives an extra factor of $\log n$ overall.

9 Binary search for a square root

6 marks

- (a) $m = 25$: $hi = 24$. $m = 12$: $hi = 11$. $m = 6$: $lo = 6$. $m = 9$: $hi = 8$. $m = 7$: $lo = 7$. $m = 8$: $hi = 7$. Now $lo = hi$: **returns 7**.
- (b) Each iteration roughly halves $hi - lo$, from n down to 0, so there are about $\log_2 n$ iterations: $\Theta(\log n)$.
- (c) Throughout, $lo^2 \leq n$ and every value above hi has square greater than n . When $lo = hi$, lo is **the largest integer with square at most n** , which is $\lfloor \sqrt{n} \rfloor$.

M1	(a) first steps correct	1
A1	(a) full trace	1
A1	(a) output	1
M1	(b) range halves	1
A1	(b)	1
E1	(c)	1

Why this question Binary search works on any yes/no question that changes answer only once along a sorted range, not just on arrays.

10 Comparing three algorithms

7 marks

- (a) A : $5 > 4$, so $\Theta(n^{\log_2 5})$, about $n^{2.32}$. B : $2 < 8$, so $\Theta(n^3)$. C : $16 = 16$, so $\Theta(n^2 \log n)$.
- (b) **C, then A, then B**, since $n^2 \log n = o(n^{2.32})$ and $n^{2.32} = o(n^3)$.
- (c) Need $a < 2^2$: $0 < a < 4$. At $a = 4$ it becomes $\Theta(n^2 \log n)$.

B1	(a) each solution	3
M1	(b) comparing the growth rates	1
A1	(b)	1
M1	(c) $a < b^d$	1
A1	(c)	1

Why this question Algorithm B makes the fewest recursive calls, yet it is the slowest. The expensive combine step dominates everything else.

11 Exam style: ternary search

8 marks

- (a) Two comparisons, then one call on a third: $T(n) = T(n/3) + \Theta(1)$.
- (b) $a = 1$, $b = 3$, $d = 0$, $a/b^d = 1$: $\Theta(\log n)$.

- (c) Ternary: about $2 \log_3 n = \frac{2}{\log_2 3} \log_2 n \approx 1.26 \log_2 n$. Binary: about $\log_2 n$. So ternary search makes about 26% more comparisons.
- (d) **Binary search**: same $\Theta(\log n)$, but fewer comparisons.

M1	(a)	1
A1	(a)	1
M1	(b)	1
A1	(b)	1
M1	(c) change of base	1
A1	(c) ternary count	1
A1	(c) comparison	1
E1	(d)	1

Why this question Two algorithms in the same Θ class can still differ by a constant factor. When it matters, count the operations exactly.

12 Exam style: two cases from scratch

8 marks

- (a) $\frac{3}{2^2} = \frac{3}{4}$.
- (b) Depth i costs $n^2(\frac{3}{4})^i$. Summing over all depths, $\sum_{i \geq 0} n^2(\frac{3}{4})^i = \frac{n^2}{1 - \frac{3}{4}} = 4n^2$, an upper bound for the finite sum.
- (c) The top level alone costs n^2 , so $n^2 \leq T(n) \leq 4n^2$: $\Theta(n^2)$.
- (d) With $n = 2^k$, depth $i < k$ costs $4^i \cdot \frac{n}{2^i} = n 2^i$, and the $4^k = n^2$ leaves cost n^2 . Total $n(2^k - 1) + n^2 = 2n^2 - n$, which is $\Theta(n^2)$. And $n^{\log_2 4} = n^2$, which agrees.

B1	(a)	1
M1	(b) cost per level	1
M1	(b) geometric sum	1
A1	(b)	1
A1	(c)	1
M1	(d) cost per level and leaves	1
A1	(d) total	1
A1	(d) agreement	1

Why this question In (d), half of all the work happens at the leaves: there are n^2 of them. That is the case $a > b^d$.

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