

Topic 6 · Sheet A: Questions

The Master Theorem and binary search

Suggested time

About 75 minutes

Total marks

73

Questions

12

Builds on

Topics 1 to 5

How to use this sheet. Work on your own paper. The questions start gently and get harder; questions 11 and 12 are exam style. Marks are a guide to how much working is expected, roughly one minute per mark. You need the Master Theorem, the recursion tree behind it, and binary search.

Master Theorem. If $T(n) = aT(\lceil n/b \rceil) + \Theta(n^d)$ with constants $a > 0$, $b > 1$, $d \geq 0$, then $T(n) = \Theta(n^d)$ if $a < b^d$; $T(n) = \Theta(n^d \log n)$ if $a = b^d$; and $T(n) = \Theta(n^{\log_b a})$ if $a > b^d$.

1

4 marks

For each recurrence, write down a , b and d .

- (a) $T(n) = 4T(n/2) + \Theta(n)$ [1]
- (b) $T(n) = T(n/3) + \Theta(n^2)$ [1]
- (c) $T(n) = 7T(n/2) + \Theta(n^2)$ [1]
- (d) $T(n) = 2T(n/4) + \Theta(\sqrt{n})$ [1]

2

4 marks

For each recurrence in question 1, compute a/b^d and give the solution.

- (a) $T(n) = 4T(n/2) + \Theta(n)$ [1]
- (b) $T(n) = T(n/3) + \Theta(n^2)$ [1]
- (c) $T(n) = 7T(n/2) + \Theta(n^2)$ [1]
- (d) $T(n) = 2T(n/4) + \Theta(\sqrt{n})$ [1]

3

6 marks

Solve each recurrence with the Master Theorem, saying which case applies.

- (a) $T(n) = 8T(n/2) + \Theta(n^2)$ [2]
- (b) $T(n) = 9T(n/3) + \Theta(n^2)$ [2]
- (c) $T(n) = 3T(n/4) + \Theta(n)$ [2]

4

6 marks

Consider $T(n) = aT(n/b) + n^d$, ignoring rounding, with recursion stopping at size 1.

- (a) How many calls are there at depth i of the recursion tree, and what size is each? [2]
- (b) Show that the total non-recursive work at depth i is $n^d \left(\frac{a}{b^d}\right)^i$. [2]
- (c) What is the maximum depth h ? [1]
- (d) Explain in one sentence why $a = b^d$ gives an extra factor of $\log n$. [1]

5

6 marks

Here is `BINARYSEARCH( $A, i, j, \alpha$ )`, started as `BINARYSEARCH( $A, 1, n, \alpha$ )` on a sorted array.

```

1  if  $i > j$  then return NO
2  if  $i = j$  then return  $i$  if  $A[i] = \alpha$ , and NO otherwise
3   $m \leftarrow \lfloor (i + j) / 2 \rfloor$ 
4  if  $A[m] = \alpha$  then return  $m$ 
5  if  $A[m] > \alpha$  then return BINARYSEARCH( $A, i, m - 1, \alpha$ )
6  return BINARYSEARCH( $A, m + 1, j, \alpha$ )

```

Let $A = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]$.

- (a) Search for $\alpha = 23$. List the calls (i, j) with their m , and give the output. [3]
- (b) Search for $\alpha = 50$. List the calls and give the output. [2]
- (c) How many entries of A were compared with α in (a)? [1]

6

6 marks

- (a) Write the recurrence for the worst-case running time of binary search. [1]
- (b) Solve it with the Master Theorem, stating a , b and d . [2]
- (c) For $n = 1000$, about how many calls does binary search make in the worst case? [2]
- (d) How many comparisons does a linear scan make in the worst case for $n = 1000$? [1]

7

6 marks

To multiply two n -digit numbers, split each into two halves of $n/2$ digits.

- (a) The obvious method computes four products of half-size numbers and combines them in $\Theta(n)$ time. Write and solve its recurrence. [2]
- (b) A clever method needs only three half-size products, still with $\Theta(n)$ extra work. Write and solve its recurrence. [2]
- (c) For $n = 1024$, compare n^2 with $n^{\log_2 3}$. [2]

8

6 marks

The Master Theorem only applies to recurrences of its exact form.

- (a) Explain why it does not apply to $T(n) = T(n-1) + n$, and solve this recurrence by unrolling. [2]
- (b) Explain why it does not apply directly to $T(n) = 2T(n/2) + n \log_2 n$. Solve it using a recursion tree, for n a power of 2. [3]
- (c) Explain why it does not apply to $T(n) = 2^n T(n/2) + n$. [1]

9

6 marks

This algorithm finds $\lfloor \sqrt{n} \rfloor$ for an integer $n \geq 1$.

```

1  lo ← 0; hi ← n
2  while lo < hi do
3      m ← ⌈(lo + hi)/2⌉
4      if m2 ≤ n then lo ← m else hi ← m - 1
5  return lo

```

- (a) Trace it for $n = 50$, listing m , lo and hi after each iteration. [3]
- (b) Explain why the number of iterations is $\Theta(\log n)$. [2]
- (c) Explain why it returns $\lfloor \sqrt{n} \rfloor$. [1]

10

7 marks

Three algorithms solve the same problem, with running times

$$T_A(n) = 5T_A(n/2) + \Theta(n^2), \quad T_B(n) = 2T_B(n/2) + \Theta(n^3), \quad T_C(n) = 16T_C(n/4) + \Theta(n^2).$$

- (a) Solve each recurrence. [3]
- (b) Rank the algorithms from fastest to slowest for large n . [2]
- (c) For $T(n) = aT(n/2) + \Theta(n^2)$, for which values of a is $T(n) = \Theta(n^2)$? [2]

11

8 marks

Exam style. Ternary search on a sorted array compares α with the entries at about one third and two thirds of the way along, then recurses on the one third that can contain α .

- (a) Write a recurrence for its worst-case running time. [2]
- (b) Solve it with the Master Theorem. [2]
- (c) Ternary search makes up to 2 comparisons per level and binary search 1. Compare their worst-case numbers of comparisons for large n . [3]
- (d) Which would you use, and why? [1]

Exam style. Prove two cases of the Master Theorem directly with recursion trees, for n a power of 2.

- (a) For $T(n) = 3T(n/2) + n^2$, find a/b^d . [1]
- (b) Show that the total work is at most $4n^2$. [3]
- (c) Hence show that $T(n) = \Theta(n^2)$. [1]
- (d) For $T(n) = 4T(n/2) + n$ with $T(1) = 1$, sum the work over all levels and show that $T(n) = \Theta(n^2)$. Check that this agrees with $\Theta(n^{\log_b a})$. [3]

End of questions. Worked answers and the mark scheme are in the matching answers PDF.