

Topic 5 · Sheet A: Questions

Divide and conquer: merge sort and recurrences

Suggested time

About 75 minutes

Total marks

73

Questions

12

Builds on

Topics 1 to 4

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 divide, conquer and combine steps, the MERGE subroutine, merge sort, and how to solve a recurrence by unrolling it or drawing its recursion tree. The Master Theorem is not needed.

Merge. $\text{MERGE}(B[1..k], C[1..t])$ fills $D[1..k+t]$ one entry at a time. While both arrays have entries left, it compares $B[i] \leq C[j]$ and copies the smaller (taking $B[i]$ on a tie). Once one array is used up, it copies the rest of the other. Merge sort splits $A[1..n]$ into $A[1..\lfloor n/2 \rfloor]$ and $A[\lfloor n/2 \rfloor + 1..n]$.

1

4 marks

Run MERGE on $B = [1, 4, 7]$ and $C = [2, 3, 9]$.

- (a) Write down D . [1]
- (b) How many iterations does the main loop make? [1]
- (c) How many comparisons $B[i] \leq C[j]$ does it make? [1]
- (d) What is the running time of MERGE on arrays of sizes k and t ? [1]

2

4 marks

- (a) How many comparisons does MERGE make on $B = [1, 2, 3]$ and $C = [4, 5, 6]$? [1]
- (b) How many comparisons does it make on $B = [1, 3, 5]$ and $C = [2, 4, 6]$? [1]
- (c) What is the smallest possible number of comparisons for sizes k and t ? [1]
- (d) What is the largest possible number of comparisons for sizes k and t ? [1]

3

6 marks

Run merge sort on $A = [5, 2, 4, 7, 1, 3, 2, 6]$.

- (a) Show how the array is split, down to single entries. [2]
- (b) Write down the result of every call to MERGE, level by level. [3]
- (c) How many calls to MERGE are made in total? [1]

4

6 marks

- (a) Write down the recurrence for the running time $T(n)$ of merge sort, keeping the floors. [2]
- (b) Write the simplified recurrence used when n is a power of 2. [1]
- (c) Take $T(1) = c$ and $T(n) = 2T(n/2) + cn$. Compute $T(2)$, $T(4)$ and $T(8)$. [2]
- (d) Guess a formula for $T(n)$ when n is a power of 2. [1]

5

6 marks

Let $T(1) = c$ and $T(n) = 2T(n/2) + cn$ for n a power of 2.

- (a) Unroll the recurrence to show that $T(n) = 2^k T(n/2^k) + kcn$ for every k with $2^k \leq n$. [3]
- (b) Choose k to find an exact formula for $T(n)$. [2]
- (c) Give $T(n)$ in Θ notation. [1]

6

6 marks

Draw the recursion tree for $T(n) = 2T(n/2) + n$, with n a power of 2.

- (a) How many calls are there at depth i , and what size is each? [2]
- (b) How much non-recursive work is done at depth i in total? [1]
- (c) How many levels does the tree have? [1]
- (d) Hence find the total work. [2]

7

6 marks

How the problem is split matters.

- (a) A version of merge sort splits into three equal parts and merges in linear time, so $T(n) = 3T(n/3) + n$. Use a recursion tree to find $T(n)$ in Θ notation. [3]
- (b) A bad version splits off just one entry each time, so $T(n) = T(n-1) + T(1) + n$ with $T(1) = 1$. Find $T(n)$ in Θ notation. [3]

8

6 marks

- (a) State the invariant for MERGE at the start of iteration r of its main loop. [2]
- (b) Check it at the start of iteration $r = 4$ in your run from question 1. [2]
- (c) Explain why the invariant shows that MERGE returns the correct array. [2]

9

6 marks

MAXDC($A[i..j]$) returns $A[i]$ if $i = j$. Otherwise it sets $m = \lfloor (i+j)/2 \rfloor$ and returns the larger of MAXDC($A[i..m]$) and MAXDC($A[m+1..j]$).

- (a) Write a recurrence for its running time on n entries. [2]
- (b) Explain why it makes exactly $n - 1$ comparisons. [2]

- (c) Give its running time in Θ notation. [1]
- (d) Is it faster than a simple left-to-right scan? [1]

10

7 marks

Suppose every call to MERGE on a total of m entries makes $m - 1$ comparisons, its worst case. Then the number of comparisons $W(n)$ made by merge sort, for n a power of 2, satisfies $W(1) = 0$ and $W(n) = 2W(n/2) + n - 1$.

- (a) Compute $W(2)$, $W(4)$ and $W(8)$. [2]
- (b) Prove by induction that $W(n) = n \log_2 n - n + 1$. [3]
- (c) Compare $W(1024)$ with the number of comparisons Bubblesort makes when $n = 1024$. [2]

11

8 marks

Exam style. Recursion trees decide which part of a divide and conquer algorithm matters. Take n a power of 2.

- (a) An algorithm makes two half-size calls and combines in $\Theta(n^2)$ time: $T(n) = 2T(n/2) + n^2$. Find the total work at depth i of its recursion tree. [2]
- (b) Hence show that $T(n) \leq 2n^2$, and give $T(n)$ in Θ notation. [3]
- (c) Another algorithm combines in constant time: $T(n) = 2T(n/2) + 1$ with $T(1) = 1$. Find the total work and give $T(n)$ in Θ notation. [3]

12

8 marks

Exam style. When n is not a power of 2, merge sort splits unevenly.

- (a) Run merge sort on $[5, 1, 4, 2, 3]$, showing every split and every merge. [2]
- (b) Suppose $T(n)$ is increasing and $T(2^\lambda) = \Theta(2^\lambda \lambda)$. For $2^{\lambda-1} \leq n \leq 2^\lambda$, use $T(2^{\lambda-1}) \leq T(n) \leq T(2^\lambda)$ to show that $T(n) = \Theta(n \log n)$. [4]
- (c) How many levels of splitting does merge sort use for $n = 5$? Compare this with $\lceil \log_2 5 \rceil$. [2]

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