

TEST 3

Student Number: _____

Email Address: _____

Last (Family) Name(s): _____

First (Given) Name(s): _____

- You must obtain at least 40% of this test in order to pass the course.
- This test is worth 20% of your final mark.
- Answer each question directly on the test paper, in the space provided. Use the reverse side of the pages for rough work. If you need more space for one of your solutions, use the reverse side of a page and *indicate clearly the part of your work that should be marked*.
- **20% rule:** If you are unable to answer (part of) a question you will get 20% of the marks for the (part of the) question if you write “I don’t know” and *nothing else* for that part/question.

The Master Theorem

If for some $a, b, d > 0$:

$$T(n) = aT(n/b) + \mathcal{O}(n^d)$$

then

$$T(n) = \begin{cases} \mathcal{O}(n^d) & \text{if } a < b^d \\ \mathcal{O}(n^d \log(n)) & \text{if } a = b^d \\ \mathcal{O}(n^{\log_b(a)}) & \text{if } a > b^d \end{cases}$$

Question 1 (out of 12)	
Question 2 (out of 13)	
Question 3 (out of 13)	
Question 4 (out of 12)	
Total (out of 50)	

Question 1 [12]

Recall that if $a_{n-1} \dots a_0$ is the binary representation of a nonnegative integer A ($a_i \in \{0,1\}$ for $0 \leq i \leq n-1$), then

$$A = \sum_{i=0}^{n-1} a_i 2^i$$

Given two nonnegative integers in binary,

$$A = a_{n-1} \dots a_0 \quad \text{and} \quad B = b_{n-1} \dots b_0$$

the Integer Multiplication Problem is to compute the product $A \cdot B$.

The so-called “school algorithm” for this problem runs in time $\Theta(n^2)$. Here we are interested in divide-and-conquer approaches to solve this problem. Throughout this question, let n be a power of 2.

Let

$$\begin{aligned} A_1 &= a_{n-1} \dots a_{\frac{n}{2}}, & A_0 &= a_{\frac{n}{2}-1} \dots a_0, \\ B_1 &= b_{n-1} \dots b_{\frac{n}{2}}, & B_0 &= b_{\frac{n}{2}-1} \dots b_0 \end{aligned}$$

then

$$A = A_1 2^{\frac{n}{2}} + A_0 \quad \text{and} \quad B = B_1 2^{\frac{n}{2}} + B_0$$

So

$$A \cdot B = (A_1 \cdot B_1) 2^n + (A_1 \cdot B_0 + A_0 \cdot B_1) 2^{\frac{n}{2}} + A_0 \cdot B_0 \tag{1}$$

Note that given nonnegative integers N and k in binary, $N \cdot 2^k$ can be computed in time $\Theta(|N| + k)$, where $|N|$ is the *length* of the binary representation of N .

a [2] Consider the following algorithm

Multiply1(A, B)

1. If $n = 1$ Return $a_0 \cdot b_0$
2. Else
3. Compute A_1, A_0, B_1, B_0
4. $C \leftarrow \text{Multiply1}(A_1, B_1)$
5. $D_1 \leftarrow \text{Multiply1}(A_1, B_0)$
6. $D_2 \leftarrow \text{Multiply1}(A_0, B_1)$
7. $E \leftarrow \text{Multiply1}(A_0, B_0)$
8. Return $(C 2^n + (D_1 + D_2) 2^{\frac{n}{2}} + E)$.
9. End If

What is the running time of **Multiply1**? Clearly specify how you apply the Master Theorem.

b [7] Observe that

$$A_1 \cdot B_0 + A_0 \cdot B_1 = (A_1 + A_0) \cdot (B_1 + B_0) - (A_1 \cdot B_1 + A_0 \cdot B_0)$$

Using this relation, give the pseudo-code for a divide-and-conquer algorithm `Multiply2` that runs asymptotically faster than `Multiply1`.

c [3] What is the running time of `Multiply2`? Clearly specify how you apply the Master Theorem.

Question 2 [13]

A palindrome is a string whose reverse is the same string. For example,

abbczcbba

is a palindrome. Also, every string of length 1 is a palindrome.

We can partition every string into substrings each of which is a palindrome: a string of length n can be partitioned into n substrings of length 1 each. However, there might be other ways that use smaller number of partitions. For example, if the given string is already a palindrome, then it can be partitioned into just one palindrome—itsself. For another example, the string *ababbbabbababa* (of length 14) can be partitioned into 6 palindromes as

aba.b.bbabb.a.b.aba

The Palindrome Partition (PP) problem is to partition a given string into the smallest number of palindromes. Here the input string is presented as an array of length n : $S[1 \dots n]$.

This question asks for a dynamic-programming algorithm that solve the PP problem.

a [3] Give an array that can be used to solve this problem. Precisely describe the size of the array and the meaning of the elements in the array. (Hint: the problem size is the length of the given string.)

b [3] Give the initialization and the recurrence for computing the elements of the array.

c [2] Give an algorithm in pseudo-code for computing the values in the array.

d [5] Give an algorithm in pseudo-code for computing an optimal partition of the input string. Your algorithm should output an array B of length n , where $B[i] \in \{0, 1\}$ for $1 \leq i \leq n$, and $B[i] = 1$ if and only if $S[i]$ is the last character of a palindrome in the optimal partition.

For example, an optimal partition of the string $a b a b b b a b b a b a b a$ (above) is

B : 1 0 0 1 0 0 0 0 1 0 0 0 0 1

S : $a . b a b . b b a b b . a b a b a$

Question 3 [13]

Consider a set of mobile computing *clients* in a certain town who each need to be connected to one of several possible *base stations*. There are n clients C_1, C_2, \dots, C_n , with the position of each client specified by its (x, y) coordinates in the plane. There are also k base stations B_1, B_2, \dots, B_k ; the position of each of these is specified by its (x, y) coordinates as well.

For each client, we wish to connect it to exactly one of the base stations. Our choice of connections is constrained in the following ways:

- There is a *range parameter* r : A client can only be connected to a base station that is within distance r .
- There is also a *load parameter* L : no more than L clients can be connected to any single base station.

The Client-Station problem is, given the positions of a set of clients and a set of base stations, to find a set of *maximum* number of clients that can be connected simultaneously to the stations.

In this question you are asked to design an algorithm for the above problem using flow network.

a [3] Construct a flow network that can be used to solve this problem. Precisely specify the nodes, the edges (with directions) and the capacities in the network.

b [2] Name an algorithm to find the maximum flow in the network. (You do *not* have to give pseudo-code for the algorithm.) What is the running time of the algorithm? Give your answer using \mathcal{O} notation.

c [3] Using the maximum flow returned by the algorithm in part **b**), show how to compute *a set of maximum number of clients* that can be connected simultaneously to the stations.

d [5] What is the relationship between the maximum flow and the maximum number of clients that can be connected simultaneously to the stations? Prove. [Clearly state the facts that you need about the output of the algorithm you gave in part **b**). You are *not* required to prove these facts.]

Question 4 [12]

Given an undirected graph, the 2-COLOUR problem is to decide whether the graph is “2-colourable”, that is, whether there is a way of colouring the vertices with two colours Red and Blue such that no edge joins two vertices of the same colour.

a [10] Give an algorithm in pseudo-code that solves 2-COLOUR. Your algorithm should output NO if the graph is not 2-colourable, and output an array *Colour* if the graph is 2-colourable, where *Colour*[*v*] is the colour of vertex *v*.

(Answer to **4a**) continues here.)

b [2] What is the running time of your algorithm ? Give your answer using Θ notation.