COMPSCI 320 2026 Midterm Test

Put your answers in the space below the questions. Write clearly and *show all your work*!

Marks for each question are shown below and just before each answer area.

This 60 minute test is worth 10% of your final grade for the course.

Question #:	1	2	3	4	Total
Possible marks:	9	9	10	12	40
Awarded marks:					

1. **Asymptotic notation**

   (a) State the definition of $f(n) = O(g(n))$. Your definition should explicitly state the constants that must exist and the condition that must hold for sufficiently large n. ***(2 marks)***

   There exist constants $c > 0$ and $n_0 \geq 0$ such that $0 \leq f(n) \leq c\,g(n)$ for all $n \geq n_0$.

   (b) Using the definition of Big-O, prove that

   $$7n^2 + 5n + 4 = O(n^2).$$

   Your proof must explicitly give suitable values for c and n_0. ***(3 marks)***

   For $n \geq 1$, we have $5n \leq 5n^2$ and $4 \leq 4n^2$. Hence

   $$0 \leq 7n^2 + 5n + 4 \leq 7n^2 + 5n^2 + 4n^2 = 16n^2.$$

   Thus the definition holds with $c = 16$ and $n_0 = 1$, so $7n^2 + 5n + 4 = O(n^2)$.

   (c) Suppose

   $$f(n) = g(n)h(n) + e(n) + j(n),$$

   where all four functions are non-negative for sufficiently large n. You know that

   $$g(n) = \Theta(n), \qquad h(n) = \Theta(\log n), \qquad e(n) = O(n), \qquad j(n) = O(n \log n).$$

   Give the tightest asymptotic bound on $f(n)$ that follows from this information. Justify your answer using the standard properties of asymptotic notation. ***(4 marks)***

   Since $g(n) = \Theta(n)$ and $h(n) = \Theta(\log n)$,

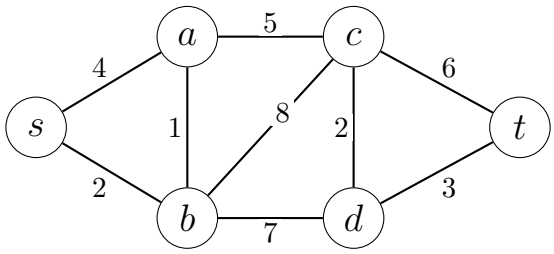
   $$g(n)h(n) = \Theta(n \log n).$$

   Also $e(n) = O(n) = O(n \log n)$ and $j(n) = O(n \log n)$, so the whole sum is $O(n \log n)$. Because the functions are eventually non-negative and the term $g(n)h(n)$ is already $\Omega(n \log n)$, the whole sum is also $\Omega(n \log n)$. Therefore

   $$\boxed{f(n) = \Theta(n \log n)}.$$

2. **Dijkstra's algorithm**

Consider the following undirected weighted graph. All edge weights are positive. Run Dijkstra's algorithm starting from vertex s.



(a) Give the order in which vertices become permanent, and the distance associated with each vertex when it becomes permanent. Show enough intermediate working to make your execution of the algorithm clear. ***(5 marks)***

The permanent order and final distances are

$$(s, 0),\ (b, 2),\ (a, 3),\ (c, 8),\ (d, 9),\ (t, 12).$$

Key relaxations: from s, $d(a) = 4, d(b) = 2$; from b, $d(a)$ drops to 3 and $d(d) = 9$; from a, $d(c) = 8$; from c, $d(t) = 14$; from d, $d(t)$ drops to 12.

(b) What is a shortest path from s to t, and what is its total length? ***(2 marks)***

A shortest path is $s \to b \to d \to t$, with length $2 + 7 + 3 = 12$.

(c) Give the edges of the shortest-path tree produced by Dijkstra's algorithm. ***(2 marks)***

$$\{(s, b), (b, a), (a, c), (b, d), (d, t)\}.$$

3. **When greedy goes wrong**

A greedy rule can appear reasonable without always producing an optimal solution. For each part below, demonstrate why the proposed greedy rule can fail.

A counterexample must show both:

- the solution produced by the greedy rule; and
- a strictly better feasible solution.

You do *not* need to prove that your counterexample is the smallest possible.

(a) **Coin changing.** Consider coin denominations $\{1, 3, 4\}$ and suppose we wish to make an amount of 6. The cashier's greedy algorithm repeatedly takes the largest coin that does not exceed the remaining amount. Show that this greedy algorithm does not produce a minimum-cardinality set of coins. ***(5 marks)***

Greedy takes $4 + 1 + 1$, using 3 coins. The feasible solution $3 + 3$ uses only 2 coins, so greedy is not optimal.

(b) **0–1 Knapsack.** For the 0–1 Knapsack problem, consider the greedy rule:

Consider items in decreasing order of value-to-weight ratio, taking an item whenever it fits.

Construct an instance containing exactly three items for which this greedy rule is not optimal. Give the capacity, the weight and value of each item, the solution produced by the greedy rule, and a better solution. ***(5 marks)***

Let the capacity be $W = 6$, and take

$$(w_1, v_1) = (4, 6), \qquad (w_2, v_2) = (3, 4), \qquad (w_3, v_3) = (3, 4).$$

The densities are $1.5, 4/3, 4/3$, so greedy takes item 1 first. No other item then fits, giving value 6. Taking items 2 and 3 has total weight 6 and value 8, which is better.

4. **Divide and conquer**

Consider the following recursive algorithm. Assume for simplicity that n is always a power of 2.

function Recursive-Algorithm(array A)
$n \leftarrow |A|$
if $n = 1$ **then return** Base(A)
divide A into two arrays A_1, A_2 of size $n/2$
$X \leftarrow$ Recursive-Algorithm(A_1)
$Y \leftarrow$ Recursive-Algorithm(A_2)
return Combine(X, Y)

Suppose the running time of COMBINE, including any work needed to divide the input, on a problem of size n is $C(n)$.

(a) Write a recurrence $T(n)$ describing the running time of RECURSIVE-ALGORITHM. You may assume that the base case takes constant time. ***(3 marks)***

A suitable recurrence is

$$T(n) = \begin{cases} \Theta(1), & n = 1, \\ 2T(n/2) + C(n), & n > 1. \end{cases}$$

Equivalently, one may include an additional $\Theta(1)$ term in the second line.

(b) State the Master Theorem for recurrences of the form

$$T(n) = aT(n/b) + \Theta(n^c).$$

State all three cases. ***(3 marks)***

For constants $a \geq 1$, $b \geq 2$, $c \geq 0$:

$$T(n) = \begin{cases} \Theta(n^c), & c > \log_b a, \\ \Theta(n^c \log n), & c = \log_b a, \\ \Theta(n^{\log_b a}), & c < \log_b a. \end{cases}$$

assuming the usual constant-size base cases.

(c) Suppose now that

$$C(n) = \Theta(n).$$

Use the Master Theorem to determine a tight asymptotic bound on $T(n)$. Clearly identify a, b, c, and the applicable case of the theorem. ***(2 marks)***

Here $a = 2$, $b = 2$, and $c = 1$. Since $\log_b a = \log_2 2 = 1 = c$, this is the equal-exponent case. Therefore

$$\boxed{T(n) = \Theta(n \log n)}.$$

(d) Suppose an algorithm has running time

$$T(n) = \begin{cases} 1, & \text{if } n = 1, \\ T(n/2) + 1, & \text{if } n > 1. \end{cases}$$

Assume n is a power of 2, and prove by induction that $T(n) = \log_2 n + 1$.

(4 marks)

We prove the result by induction over powers of 2.

Base case: For $n = 1$,

$$T(1) = 1 = \log_2 1 + 1.$$

Induction hypothesis: Assume that

$$T(n/2) = \log_2(n/2) + 1.$$

Then

$$\begin{aligned} T(n) &= T(n/2) + 1 \\ &= \log_2(n/2) + 1 + 1 \\ &= \log_2 n - 1 + 2 \\ &= \log_2 n + 1. \end{aligned}$$

Therefore, $T(n) = \log_2 n + 1$ for every n that is a power of 2.