National University of Singapore

School of Computing

CS3230 - Design and Analysis of Algorithms

Midterm Test

(Semester 1 AY2026/27)

Time Allowed: 120 minutes

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains TWO (2) sections.
   It comprises TWENTY-ONE (21) printed pages, including this page.
3. This is an **Open Book** Assessment.
   You cannot use any electronic device except one calculator.
4. For Section A, use the boxes on page 15 (use 2B pencil).
   For Section B, answer **ALL** questions within the **boxed space**.
   If you leave the boxed space blank, you will get automatic free 0.5 mark for that box.
   However, if you write at least a single character and it is totally wrong, you will get 0 mark.
   Canceling your (incomplete) answer can be done by crossing the entire box with a big **X**.
   Update one day before the midterm test: Specific for this paper (no guarantee for final assessment format), we still give partial marks of up to half the marks of the box for certain partial solutions.
   You can use either pen or pencil. Just make sure that you write **legibly**!
5. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
   Read all the questions first! Some questions might be easier than they appear.
6. You can assume that all **logarithms are in base** 2.
7. The total marks of this paper is 60 marks.
   It will then be scaled to 30% of the course weightage.

A MCQs ($20 \times 1.5 = 30$ marks)

1. Rank the following functions in increasing order of growth.
   No two functions have the same order of growth.

   - $f_1(n) = n\sqrt{n}$
   - $f_2(n) = n \log \log n$
   - $f_3(n) = \log^{1/2} n$
   - $f_4(n) = n^{1.464}$
   - $f_5(n) = \log \log n$

   a). $f_1(n) < f_2(n) < f_3(n) < f_4(n) < f_5(n)$
   b). $f_5(n) < f_3(n) < f_2(n) < f_4(n) < f_1(n)$
   c). $f_3(n) < f_5(n) < f_2(n) < f_4(n) < f_1(n)$
   d). $f_5(n) < f_3(n) < f_2(n) < f_1(n) < f_4(n)$
   e). $f_5(n) < f_3(n) < f_4(n) < f_2(n) < f_1(n)$

2. Which of the following statements is **TRUE**?

   a). $2^{\lfloor \log n \rfloor} \in o\left(2^{\log n}\right)$.
   b). $2^{\lfloor \log^2 n \rfloor} \in o\left(2^{\log^2 n}\right)$.
   c). $\lfloor \log n \rfloor^{100} \in o\left((\log n)^{100}\right)$.
   d). $(\log \log n)^{\lfloor \log \log n \rfloor} \in o\left((\log \log n)^{\log \log n}\right)$.
   e). None of the above

3. You are asked to solve a computational problem in $o(n^4)$ time.
   Any algorithm for this problem requires $\Omega(n^3)$ time.
   Which listed running-time bound is consistent with both constraints?

   a). $\Theta(n^5)$
   b). $\Theta(n^4)$
   c). $O(n^3)$
   d). $o(n^3)$
   e). $\Theta(n^2 \log^{\frac{1}{2}} n)$

4. Four out of the five options below fall under the same case of Master Theorem.
   Pick the odd one out.

   a). $T(n) = 2 \cdot T(n/2) + n$
   b). $T(n) = 2 \cdot T(n/\sqrt{2}) + n^2$
   c). $T(n) = T(n/2) + 1$

d). $T(n) = T(2n/3) + 1$

e). $T(n) = T(n/2) + n$

5. Which of the following recurrences has solution $T(n) \in \Theta(n^2 \log n)$?

   a). $T(n) = 4 \cdot T(n/2) + n^2$

   b). $T(n) = 4 \cdot T(n/2) + n$

   c). $T(n) = 3 \cdot T(n/2) + n^2$

   d). $T(n) = 8 \cdot T(n/2) + n^2$

   e). None of the above

6. Let n be a power of 2. Every arithmetic operation takes constant time, both recursive calls are executed, and there is no memoization.

```
Solve(n):
    if n = 1: return
    for i = 1 to n:
        j = 1
        while j < n:
            j = 2*j
    Solve(n/2)
    Solve(n/2)
```

   Let $T(n)$ be the running time. Which statement is **TRUE**?

   a). $T(n) \in O(n \log n)$

   b). $T(n) \in \Theta\big(n \log^2 n\big)$

   c). $T(n) \in o\big(n \log^2 n\big)$

   d). $T(n) \in \Omega(n^2)$

   e). None of the above.

7. An array is *bitonic* if its elements first strictly increase and then strictly decrease. You are given two bitonic arrays A and B, each containing n distinct elements. All $2n$ elements are distinct. Which of the following is the fastest asymptotic running time for merging A and B and obtaining an array of all the $2n$ elements in sorted order?

   a). $O(\log n)$

   b). $O(n)$

   c). $O(n \log n)$

   d). $O(n^{3/2})$

   e). $O(n^2)$

8. Let T be a binary tree with n nodes and depth d. Each node contains a distinct numerical value. A node is a **local minimum** if its value is smaller than the values of all of its neighboring nodes (left/right child, and parent).

   Consider the following algorithm for finding ANY local minima. Start at the root. If the current node is a local minimum, return it. Otherwise, move to any child whose value is smaller than the current node and repeat.

   Which of the following statements is **TRUE**?

   a). The algorithm is correct and runs in $O(d)$ time in the worst case. In particular, it runs in $O(\log n)$ time when the tree is balanced.

   b). The algorithm is incorrect, because a node even smaller than the root and both its children may exist in the other subtree; therefore both subtrees must be searched.

   c). The algorithm is correct but requires $\Theta(n)$ time in the worst case, since the local minimum may occur at any of the n nodes.

   d). The algorithm is correct and runs in $O(\log n)$ time for every binary tree, because each node has at most two children.

   e). The algorithm is incorrect as every node is not examined in the worst case, and a local minimum cannot be certified without comparing it against all other nodes.

9. Consider the following algorithm for computing a^n, where n is a positive integer.

$$
\begin{aligned}
&\textsc{Power}(a, n):\\
&\quad \text{If } n \le 2, \text{ return } a^n\\
&\quad x = \textsc{Power}(a, \lfloor n/3 \rfloor)\\
&\quad \text{If } n \bmod 3 = 0, \text{ return } x^3\\
&\quad \text{If } n \bmod 3 = 1, \text{ return } x^3 a\\
&\quad \text{If } n \bmod 3 = 2, \text{ return } x^3 a^2
\end{aligned}
$$

   Assume that each arithmetic operation takes constant time.
   What is the worst-case running time of the algorithm?

   a). $\Theta(n)$

   b). $\Theta(n^{1/3})$

   c). $\Theta(\log^3 n)$

   d). $\Theta(\log n)$

   e). None of the above.

10. You are given access to a sorted, strictly increasing array A, but you are not told its length. You may query $A[i]$ for any positive integer i.
    If i is exceeds the length of the array, the query returns $+\infty$.

    Given a value x, suppose that $x = A[p]$ for some unknown position p.
    Which of the following is the fastest way to find p in the worst case?

a). Query $A[1], A[2], A[4], A[8], \ldots$ until you find the first 2^j such that $A[2^j] \geq x$, and then search the interval $2^{j-1}, \ldots, 2^j$ sequentially. This takes $O(p)$ time.

b). Use binary search directly on A, which takes $O(\log p)$ time even though the length of the array is unknown.

c). Query $A[1], A[2], A[4], A[8], \ldots$ until you find the first 2^j such that $A[2^j] \geq x$, and then binary-search between 2^{j-1} and 2^j. This takes $O(\log p)$ time.

d). Examine $A[1], A[2], \ldots$ in order until x is found, which takes $O(p)$ time and is asymptotically optimal because the array length is unknown.

e). It is impossible to guarantee finding p without first determining the length of A.

11. An array contains n distinct keys, where n is even. You are promised that

$$A[2i-1] < A[2i] \qquad \text{for every } i = 1, \ldots, n/2;$$

thus each adjacent pair is already sorted. No ordering between different pairs is promised.

Among algorithms that use only comparisons between keys, what is the tight asymptotic bound on the minimum possible worst-case number of comparisons needed to sort the whole array?

a). $\Theta(n)$

b). $\Theta(n \log \log n)$

c). $\Theta(n^2)$

d). $\Theta(n \log n)$

e). None of the above.

12. You are given n visually identical balls. Exactly one ball is heavier than normal and exactly one different ball is lighter than normal. All the remaining balls have the same weight.

You may use a balance scale. In each weighing, you may place any collection of balls on the left pan and any collection on the right pan. A weighing has three possible outcomes: *left heavier*, *balanced*, or *right heavier*. Your goal is to **identify both the heavy ball and the light ball.**

Using a decision-tree argument, which of the following is the highest valid lower bound on the worst-case number of weighings required by any algorithm?

a). $\lceil \log_3 n \rceil$

b). $\lceil \log_3 \binom{n}{2} \rceil$

c). $\lceil \log_3 \big(n(n-1)\big) \rceil$

d). $\lceil \log_2 \big(n(n-1)\big) \rceil$

e). $n - 1$

13. You are running the registration desk at a conference. Attendees arrive one at a time throughout the day, and you must give away exactly one door prize. The lucky winner must be announced the moment registration closes, but you have no idea how many people (n) will show up.

So you use the following strategy. You keep a single sticky note on your desk. When the first attendee registers, you write their name down. When the i-th ($i \geq 2$) attendee registers, you cross out the name and write theirs instead, with probability $\frac{1}{i}$ (by rolling an i-sided die and update the name on the sticky note when seeing a 1). Otherwise, you leave the sticky note alone.

When registration closes, the name on the sticky note wins and is immediately announced. What is the probability distribution of the winner?

a). The attendee who arrived *later* has higher chance

b). The attendee who arrived *earlier* has higher chance

c). Uniform: each attendee has probability $\frac{1}{n}$ to be selected as winner

d). Uniform, but only if n is known in advance

e). Attendee i has probability $\frac{1}{i}$ of winning

14. An array $A[1..n]$ has a *majority element* if some value occurs strictly more than $n/2$ times. Assume that n is a positive power of 2.
Consider the following procedure MAJORITY(A), where n is the array size.
The special marker NONE is not a possible array value.

i). If $n = 1$, return $A[1]$.

ii). Set $x \leftarrow$ MAJORITY($A[1..n/2]$) and $y \leftarrow$ MAJORITY($A[n/2 + 1..n]$).

iii). If $x = y$, return x.

iv). Otherwise, scan A and count the occurrences of each of x and y that is not NONE. If one occurs strictly more than $n/2$ times, return that value. Otherwise, return NONE.

The algorithm is *correct* if it returns the majority element whenever one exists, and NONE otherwise. Which of the following statements is **TRUE**?

a). The algorithm is correct and has worst-case running time $\Theta(n)$

b). The algorithm is correct and has worst-case running time $\Theta(n \log n)$

c). The algorithm is incorrect and has worst-case running time $\Theta(n)$

d). The algorithm is incorrect and has worst-case running time $\Theta(n \log n)$

e). None of the above

15. Consider a randomized version of Quicksort based on this pivot selection rule: choose two distinct elements uniformly at random from the current array and use the smaller of the two as the pivot. What is the expected running time for sorting n distinct elements using this algorithm?

a). $\Theta(n^2)$

b). $\Theta(n \log n)$

c). $\Theta(n \log^2 n)$

d). $\Theta(n^{3/2})$

e). None of the above.

16. Each of n distinct balls is placed independently and uniformly at random into one of n bins. Let X be the number of 4-element subsets of balls whose 4 balls are placed in the same bin. What is $\mathbb{E}[X]$?

    a). $\Theta(n^3)$

    b). $\Theta(n)$

    c). $\Theta(n^{-1})$

    d). $\Theta(n^{-3})$

    e). None of the above

17. Randomized Quicksort sorts five distinct keys $x_1 < x_2 < x_3 < x_4 < x_5$. At each recursive call, the pivot is chosen uniformly from that subarray using fresh randomness and compared with every other key in that subarray.

    What is the probability that x_1 is compared with *both* x_3 and x_5 during the entire execution?

    a) 1/3

    b) 4/15

    c) 2/5

    d) 2/3

    e) None of the above.

18. Given the following DP 0/1-Knapsack Table shown in Table 1 where the rows go from $i = 0$ to $i = n$ and the columns go from $j < 0$, $j = 1$, all the way until $j = W$:

	$j<0$	0	1	2	3	4	5	6	7	8
$i=0$	$-\infty$	0	0	0	0	0	0	0	0	0
1	$-\infty$	0	5	5	5	5	5	5	5	5
2	$-\infty$	0	5	9	14	14	14	14	14	14
3	$-\infty$	0	5	9	14	14	14	14	14	14
4	$-\infty$	0	5	9	14	14	17	22	22	22
5	$-\infty$	0	5	9	14	14	17	22	22	22
6	$-\infty$	0	5	9	14	14	17	22	22	22

Table 1: DP table with 7 rows and 10 columns

Which of the following statements is **FALSE** for the particular 0/1-Knapsack case?

a) There are $n = 6$ items being considered

b) The largest knapsack size is $W = 8$

c) The optimal solution is 22

d) If the last two items (item 5 and 6 are removed), the optimal solution remains 22

e) If the knapsack size W is reduced to 5, the optimal solution remains 22

19. You are given a string s with n lowercase ['a'..'z'] characters. In one step, you can insert any character at any index of the string. Return the minimum number of steps to make s palindrome[1]. For example, if $s =$ "banana", we can append 'b' to make palindrome "bananab".

    Let $dp(i, j)$ be the minimum number of steps (inserted characters) to make substring $s[i..j]$ a palindrome and is called with $dp(0, n-1)$. Which of the options below is **NOT** part of the Dynamic Programming formulation $dp(i, j)$ for this problem?

    a). $dp(i, j) = 0$ if $i = j$

    b). $dp(i, j) = 0$ if $i > j$

    c). $dp(i, j) = dp(i+1, j-1)$ if $s[i] = s[j]$

    d). $dp(i, j) = 1 + \min(dp(i+1, j), dp(i, j-1))$ if $s[i] \neq s[j]$

    e). $dp(i, j) = 2 + dp(i+1, j-1)$ if $s[i] \neq s[j]$

20. An *independent set* of a graph is a set of vertices such that no two vertices in the set are adjacent.

    Let T be a tree rooted at r. For each vertex v, let $I(v)$ and $O(v)$ denote the maximum size of an independent set in the subtree rooted at v, subject to the conditions that v is included and excluded, respectively. Let $\mathcal{C}(v)$ be the set of children of v.

    Which of the following recurrences is correct?

    a). $I(v) = 1 + \sum_{u \in \mathcal{C}(v)} O(u), \qquad O(v) = \sum_{u \in \mathcal{C}(v)} \max\{I(u), O(u)\}$

    b). $I(v) = 1 + \sum_{u \in \mathcal{C}(v)} \max\{I(u), O(u)\}, \qquad O(v) = \sum_{u \in \mathcal{C}(v)} O(u)$

    c). $I(v) = \sum_{u \in \mathcal{C}(v)} O(u), \qquad O(v) = 1 + \sum_{u \in \mathcal{C}(v)} \max\{I(u), O(u)\}$

    d). $I(v) = 1 + \max_{u \in \mathcal{C}(v)} O(u), \qquad O(v) = \max_{u \in \mathcal{C}(v)} \max\{I(u), O(u)\}$

    e). None of the above.

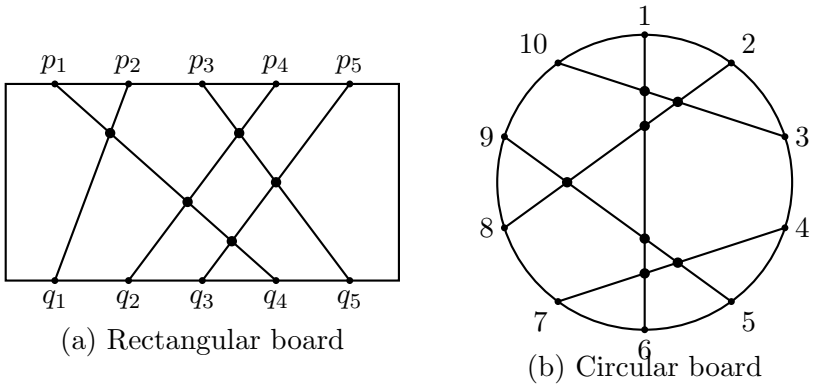
[1] A palindrome string is one that reads the same backward as well as forward.

B Essays (30 marks)

B.1 Routing Conflicts on Circuit Boards (10 marks)

A hardware company is designing circuit boards with prescribed pairwise connections between electrical terminals. During the initial design phase, each required connection is represented by a straight trace joining its two terminals. Two traces on the same circuit-board layer cannot cross: a crossing would electrically connect the two traces. Thus, **every pair of crossing traces represents a routing conflict** that must be resolved later by rerouting a trace or moving part of it to another layer. The engineers would like to quickly estimate how complicated a proposed layout is by **counting the number of routing conflicts.**

Assume throughout that no three traces cross at the same interior point.



(a) Rectangular board (b) Circular board

Figure 1: Examples of preliminary straight-line circuit routing. The company knows all the terminal points and the pairs that need to be connected to each other. Dots inside the boards represent routing conflicts, that is, intersections between pairs of traces. The rectangular board in (a) has **5 routing conflicts**, while the circular board in (b) has **7 routing conflicts**.

B.1.1 Terminals along Parallel Rows (6 marks)

A rectangular circuit board has n terminals along its top edge and n terminals along its bottom edge. Each top terminal is connected to exactly one bottom terminal and every bottom terminal is used exactly once. Figure 1 shows an example of such a layout.

To formalize simply, assume that the board is aligned with standard 2-D axes, the top edge is along the line $y = 1$, and the bottom edge is along $y = 0$. All the terminal points are specified by their x co-ordinates. Points on the top edge are denoted by p_i for all $i \in [n]$, and points on the bottom line by q_i for all $i \in [n]$. Pairs of terminals that must be connected are given as a list of ordered pairs $\mathcal{C} = [c_i = (p_j, q_k) : i, j, k \in [n]]$.

Design a divide-and-conquer algorithm that counts the total number of pairs of crossing traces in $O(n \log n)$ time or faster. Explain why your algorithm is correct and analyze its running time.

[Hint: you can assume that all the terminals on one edge, say the bottom one, are indices of an array, and the terminals connected to each index are the values stored in the array at that index. A

routing conflict between two traces, say (p_a, q_b) and (p_c, q_d), then indicates that if the indices satisfy $q_b \leq q_d$, the elements satisfy the inverted relation $p_a \geq p_c$.]

[Hint 2: does merge sort help you here?]

B.1.2 Terminals along a Circle (4 marks)

The company is now designing a circular circuit board. It has $2n$ terminals arranged around its boundary and the design specifies n pairs of terminals that must be connected. Each terminal belongs to exactly one pair. Figure 1 shows an example of such a layout.

Design a divide-and-conquer algorithm for this layout that counts the total number of routing conflicts. Your algorithm should run in $O(n \log^2 n)$ time or faster. Explain why your algorithm is correct and analyze its running time.

[Hint: Can you 'cut' the circle and see how part (a) can help?]

B.2 Randomized Matching (10 marks)

Let $G = (S \cup T, E)$ be a simple undirected bipartite graph. The vertices in S represent *students*, and the vertices in T represent *internship jobs*. An edge $\{s, t\} \in E$ indicates that student s is considering job t.

For a vertex v, let $N(v)$ denote its set of neighbors, and let $d(v) = |N(v)|$ denote its degree. In particular, $N(s)$ is the set of jobs that student s is considering.

A *matching* is a set of edges no two of which share an endpoint. Consider the following randomized procedure:

1. Every student s with $N(s) \neq \emptyset$ chooses exactly one job $t \in N(s)$ uniformly at random and submits an application for it. The students make their choices independently.

2. After all students submit their applications, for every job receiving at least one application, the employer selects exactly one applicant and makes that student an offer. If there are multiple applicants, the selection may be arbitrary.

3. The output matching M has an edge $\{s, t\}$ exactly when student s receives an offer for job t.

These edges form a matching because every student applies for at most one job and every job offers a position to at most one student.

After the procedure, a student is called *free* if they receive no offer, and a job is called *free* if no offer is made for it. An edge is called a *free edge* if both of its endpoints are free.

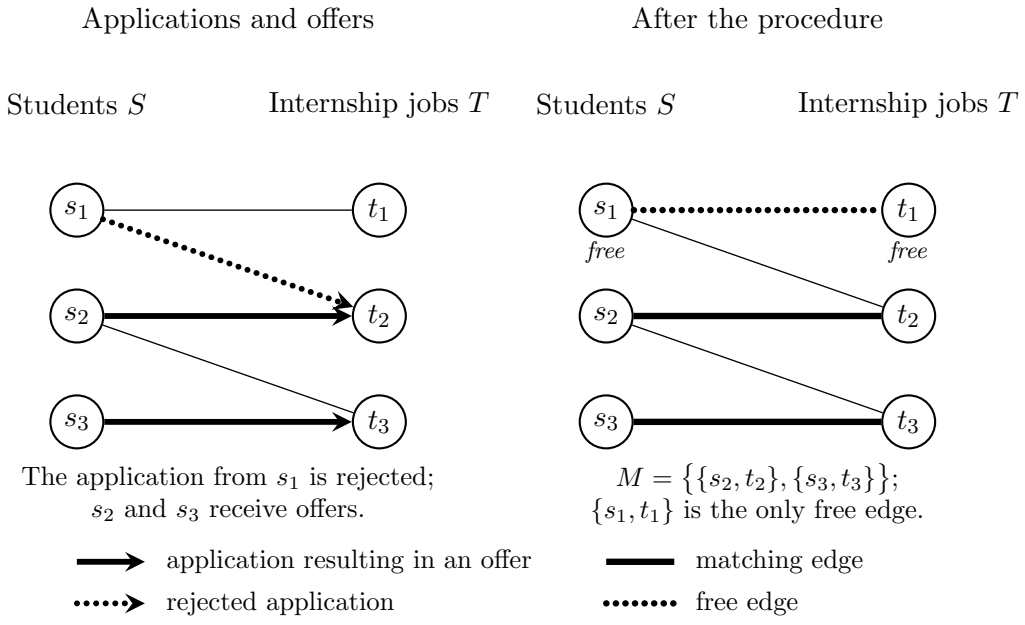


Figure 2: An execution of the randomized matching procedure: $S = \{s_1, s_2, s_3\}$, $T = \{t_1, t_2, t_3\}$, and $E = \big\{\{s_1, t_1\}, \{s_1, t_2\}, \{s_2, t_2\}, \{s_2, t_3\}, \{s_3, t_3\}\big\}$. Students s_1 and s_2 both apply for job t_2, while student s_3 applies for job t_3. Student s_2 receives an offer for job t_2, and student s_3 receives an offer for job t_3. Student s_1 receives no offer, and no offer is made for job t_1. Thus, $\{s_1, t_1\}$ is the only free edge.

B.2.1 Conditional probability of receiving an offer (3 marks)

Fix a student s with $d(s) > 0$. Prove that, conditioning on the event that at least half of the jobs in $N(s)$ receive no application from any other student, the probability that s receives an offer is at least $1/2$. More specifically, let

$$J = \{t \in N(s) : t \text{ receives an application from some student other than } s\}.$$

Assuming that the conditioning event has positive probability, prove that

$$\Pr\left[s \text{ receives an offer} \,\middle|\, |J| \leq \frac{d(s)}{2}\right] \geq \frac{1}{2}.$$

B.2.2 Free edges incident to one student (4 marks)

For every student s, let F_s be the number of free edges incident to s after the procedure. Prove that

$$\mathbb{E}[F_s] \leq d(s)/2.$$

B.2.3 The total number of free edges (3 marks)

Let $F \subseteq E$ be the set of free edges after the procedure. Prove that

$$\mathbb{E}[|F|] \leq |E|/2.$$

B.3 Prompt Triage (10 marks)

An LLM serving endpoint receives n prompts, numbered $0, 1, \ldots, n-1$ in arrival order. Exactly one prompt arrives per second: prompt i arrives at second i. Prompt i is described by two integers: its *reward* r_i ($1 \le r_i \le 10^5$), the value of answering it, and its *length* ℓ_i ($1 \le \ell_i \le 10^5$), the number of output tokens a complete answer requires. For this problem, you will be given a 2D integer array `prompts` of size $n \times 2$ upfront where `prompts[i] = [`r_i`, `ℓ_i`]`. Let $n =$ `prompts.length`. You will be asked later in Section B.3.3 to analyze in terms of n.

The model generates exactly one token per second. While it is generating, it cannot read incoming prompts: any prompt that arrives during generation is *dropped* and earns nothing.

Whenever a prompt i arrives and the model is idle, you must choose one of the two actions:

1. **Answer it.** You earn r_i. The model is busy for the next ℓ_i seconds, so prompts $i{+}1, i{+}2, \ldots, i{+}\ell_i$ are dropped (indices beyond $n-1$ refer to prompts that do not exist and are simply ignored). The model is idle again when prompt $i + \ell_i + 1$ arrives, if it exists.
2. **Discard it.** You earn nothing. Discarding takes no time, so the model is idle when prompt $i + 1$ arrives.

Return the maximum total reward that your model can earn.

Example 1: `prompts = [[4,1],[3,2],[1,1],[3,3]]`, return `7`.
Here are the optimal actions to earn reward of `7`, second by second:

0. Answer prompt 0 (+4); the model is busy for 1 second, so prompt 1 is dropped.
1. *model is currently busy generating answer for prompt 0.*
2. Prompt 2 arrives while the model is idle: discard it (strategically).
3. Answer prompt 3 (+3); it would drop prompts 4, 5, and 6, which do not exist.

The total is $4 + 3 = 7$.
Answering prompt 2 after prompt 0 would drop prompt 3, for a total of only $4 + 1 = 5$.

Example 2: `prompts = [[12,4],[7,1],[2,1],[8,1],[3,1],[5,2]]`, return `20`.
Here are the optimal actions to earn reward of `20`, second by second:

0. Discard prompt 0.
1. Answer prompt 1 (+7), which drops prompt 2.
2. *model is currently busy generating answer for prompt 1.*
3. Answer prompt 3 (+8), which drops prompt 4.
4. *model is currently busy generating answer for prompt 3.*
5. Answer prompt 5 (+5); it would drop prompts 6 and 7, which do not exist.

The total is $7 + 8 + 5 = 20$.
Answering prompt 0 earns 12 but drops prompts 1 to 4; the best continuation is then prompt 5, for a total of only $12 + 5 = 17$.

B.3.1 Subtask 1: Test Your Understanding (2 marks)

For 1 mark each, what are the answers if:

1. `prompts = [[9,6],[8,5],[7,4],[6,3],[5,2],[4,1],[10,100000]]`
2. `prompts = [[6,2],[4,1],[4,1],[4,1],[7,2],[3,1],[4,1]]`

B.3.2 Subtask 2: Prove correctness/Provide a counterexample + a fix (6 marks)

A GenAI (the exact prompt, the actual model used, and the thinking budget are not specified) was tasked to solve this problem and it produced the following output:

This is a Dynamic Programming (DP) problem like 0/1-Knapsack. The n prompts occupy seconds $0, 1, \ldots, n-1$, so the model has a budget of n seconds in total. Answering prompt i uses $w_i = \min(\ell_i + 1,\ n - i)$ of them: one second to read it plus ℓ_i seconds of generation, capped at $n - i$ because nothing exists after second $n - 1$.
State. For $0 \le i \le n$ and $0 \le s \le n$, let $go(i, s)$ be the maximum reward obtainable from prompts $i, i+1, \ldots, n-1$ with s seconds of budget left.
Base case. $go(n, s) = 0$ for every s (no prompts left).
Recurrence ($0 \le i < n$):

$$go(i,s) = \begin{cases} go(i+1,\ s) & \text{if } w_i > s \\ \max\big(\ go(i+1,\ s),\ r_i + go(i+1,\ s - w_i)\ \big) & \text{if } w_i \le s \end{cases}$$

In the first case prompt i no longer fits, so it is discarded; in the second case we either discard it or answer it and pay w_i seconds. The answer is $go(0, n)$.

If this generated algorithm is correct, prove its correctness.
If the generated algorithm is incorrect, provide a counterexample and give a corrected DP algorithm with asymptotically optimal worst-case running time. Follow the explanation format of the generated algorithm above. Briefly justify its correctness.

B.3.3 Subtask 3: Analyze Time and Auxiliary Space Complexity (2 marks)

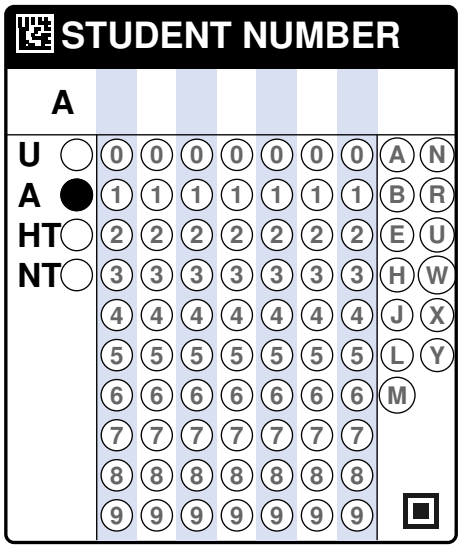
Give tight asymptotic bounds, using Θ notation, on both the worst-case running time and the worst-case auxiliary space usage of the algorithm you analyze below. The term auxiliary space implies that the $\Theta(n)$ space required to store the input `prompts` is not part of the analysis.

If you declare that the proposed algorithm in Section B.3.2 is correct, analyze that algorithm. Otherwise, analyze your corrected algorithm.

The grading of this section is **independent** of your answer in Section B.3.2.

The Answer Sheet

Write your Student Number in the box below using **(2B) pencil** and shade the corresponding bubbles.
Do NOT write your name.



Fill in your MCQ answers in the special MCQ answer box below for automatic grading.
We do not manually check your answer.
Shade your answer properly (use (2B) pencil, fully enclose the circle; select just ONE circle).

No	1	2	3	4	5	6	7	8	9	10
A	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
B	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
C	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
D	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
E	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

No	11	12	13	14	15	16	17	18	19	20
A	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
B	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
C	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
D	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
E	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

Box B.1.1. Terminals along Parallel Rows



Box B.1.2. Terminals along a Circle



Box B.2.1. Conditional probability of receiving an offer

Box B.2.2. Free edges incident to one student

Box B.2.2. continued (if you need more writing space)

Box B.2.3. The total number of free edges

Box B.3.1. Test Your Understanding; Write TWO positive integers, the return value of:

1. `prompts = [[9,6],[8,5],[7,4],[6,3],[5,2],[4,1],[10,100000]]`, and
2. `prompts = [[6,2],[4,1],[4,1],[4,1],[7,2],[3,1],[4,1]]`

Box B.3.2. Prove correctness/Provide a counterexample + a fix

PS: This question has partial marks scheme, i.e., [0 (wrong), 0.5 (blank), [undisclosed], 6 (correct)]

Box B.3.2. continued (if you need more writing space)

Box B.3.3. Analyze Time **and Auxiliary Space** Complexity

– END OF PAPER; All the Best –