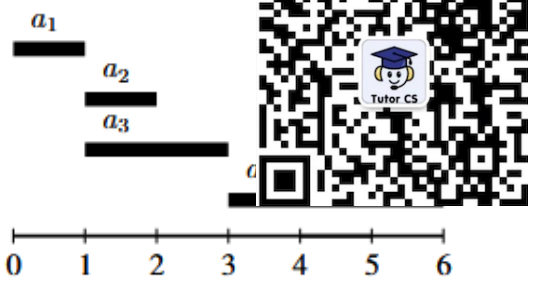
1. Please select the right answers.

(a) (6 points) We apply the dynamic programming algorithm we have seen in class to solve the weighted activity scheduling problem. An instance of this problem is show below (Figure A). The weight of an activity a_i is noted V_i and is equal to the length (or duration) of the activity. The predecessor of an activity a_i is noted $pred(a_i)$. We filled the dynamic programming table M below (Figure B). Please se

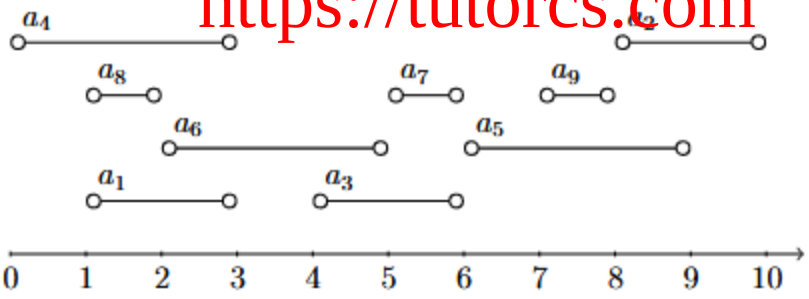


activity (a_i)	a_1	a_2	a_3	a_4
pred	-	a_1	a_1	a_2
$M[a_i]$	1	2	3	6
$V_i + M[pred(a_i)]$	1	2	3	6
$M[a_{i-1}]$	0	1	3	3

(A) Instance of the weighted activity scheduling problem (B) Dynamic programming table M

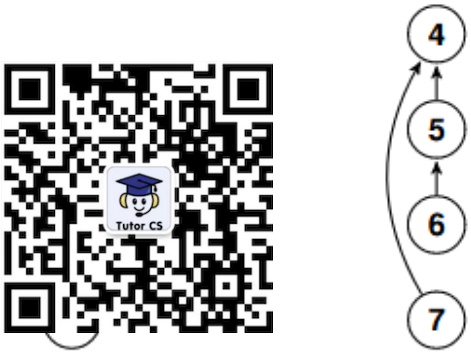
A. The table M does not contain errors.
B. The table M contains one error.
C. The table M contains two errors.
D. The table M contains three errors.
E. The table M contains four errors.

(b) (3 points) What is the longest series of activities (i.e. number of activities) that will be returned by the greedy algorithm for solving the scheduling problem (i.e. finding the maximal number of compatible activities) as seen in class? Please select all the sets that are right.



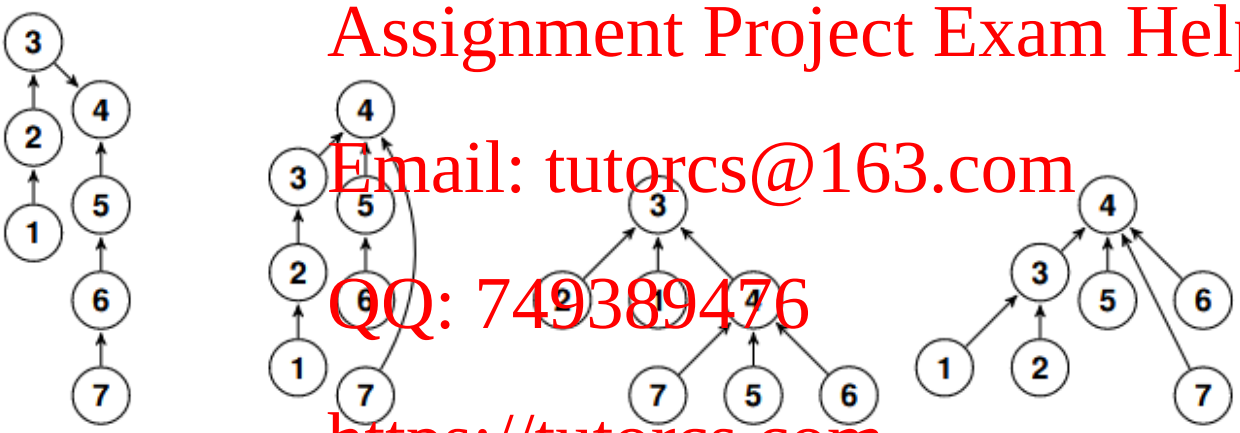
A. (a8, a6, a7, a5, a2)
B. (a8, a6, a7, a9, a2)
C. (a8, a1, a3, a9, a2)
D. (a4, a3, a5)
E. The exercise can not be solved using a greedy algorithm. As seen in class, a dynamic programming approach is needed.

(c) (3 points) We use a tree representation to model disjoint sets, and implement unions as *union-by-height* with path compression. We have as input the two following trees T_1 and T_2 representing two distinct disjoint sets.



(A) T_1 (B) T_2

We perform the union of the set containing 6 with the set containing 1 (i.e. `union(6,1)`). Which of the options below is/are possible output(s)?



A. T_A B. T_B C. T_C D. T_D

(d) (3 points) I run three different algorithms at home. I am reporting the size of the input and the time taken by each algorithm in the tables shown below (one table per each algorithm). Please select the answer that form a correct tight conjecture about the asymptotic running time of my algorithms.

Alg1

n	time (in secs)
2^{15}	22.31
2^{16}	89.89
2^{17}	369.96
2^{18}	1722.01

Alg2

n	time (in secs)
2^{15}	10.21
2^{16}	20.31
2^{17}	42.01
2^{18}	83.27

Alg3

n	time (in secs)
2^{15}	2.01
2^{16}	5.06
2^{17}	12.99
2^{18}	30.01

A. Alg1 = O(nlog(n)), Alg2 = O(n), Alg3 = O(2.5 * n)

B. Alg1 = $O(n^2)$, Alg2 = O(n), Alg3 = O(nlog(n))

C. We need information about the computer where the experiments run in order to answer this question

D. Alg1 = O(4 * n), Alg2 = O(2 * n), Alg3 = O(0.25 * n)

Recall the master theorem.

Theorem 1 (Master theorem) *Let $a \geq 1$ and $b \geq 1$ be two constants, and $f(n)$ a function. $\forall n \in \mathbb{N}^+$ we define $T(n)$ as:*

$$T(n) = aT\left(\frac{n}{b}\right) + f(n), \text{where } \frac{n}{b} \text{ is interpreted as } \lfloor \frac{n}{b} \rfloor \text{ or } \lceil \frac{n}{b} \rceil.$$

Then, we can find a[illegible] r $T(n)$ such that:

1. *If $f(n) = O(n^{[illegible]})$ [illegible] $T(n) = \Theta(n^{\log_b a})$.*
2. *If $f(n) = \Theta(n^{[illegible]})$ [illegible] $n) = \Theta(n^{\log_b a} \log^{p+1} n)$.*
3. *If $f(n) = \Omega(n^{[illegible]})$ [illegible] $d\ a \cdot f\left(\frac{n}{b}\right) \leq cf(n), \forall n > n_0$ with $c < 1$ and $n_0 > 0$. Then $T(n) = \epsilon$*

(e) (3 points) Which o[illegible]nts is true when solving the recurrence $T(n) = 7T(\frac{n}{3}) + n^2$ using the Master Theorem.

A. It can be solved using case 1 of the Master Theorem.
B. It can be solved using case 2 of the Master Theorem.
C. It can be solved using case 3 of the Master Theorem.
D. It can not be solved using the Master Theorem because it does not hold the regularity condition of case 3.
E. It can not be solved using the Master Theorem because any of the initial conditions of the three cases apply to it.

(f) (3 points) We populated the hash table shown below by using open addressing with the following quadratic probing function $h(k, i) = (k + i^2) mod\ m$, where k is the key, i is the probe and m represent the number of slots. Please notice that the character '-' represents an empty slot. How many slots are examined if a search procedure to find the key 42 is performed (count also the position where the key 42 is finally found)?

Index	0	1	2	3	4	5	6	7	8	9	10	11
Value	-	-	50	42	-	29	41	31	19	-	22	-

A. 1 B. 2 C. 3 D. 4 E. 5

2. The array `A={27,14,18,9,7,33}` represents a complete binary tree (as seen in class).

(a) (3 points) Draw the Binary Tree represented by A. You are only required to show the final tree, although if you draw intermediate trees, please circle your final result for ANY credit.

(b) (4 points) Draw the Binary Search Tree obtained by inserting the elements of `A` one by one in order of the initial list. You are only required to show the final tree, although if you draw intermediate trees, please circle your final result for ANY credit.

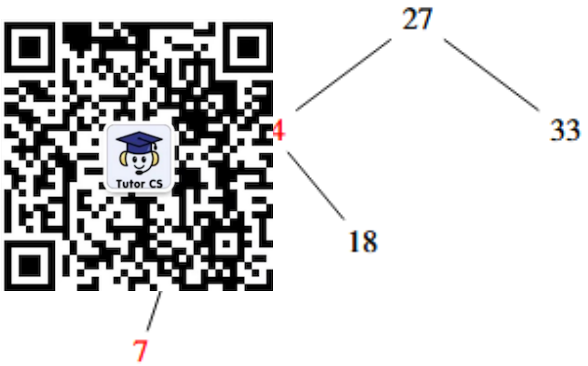
2. The array `A={27,14,18,9,7,33}` represents a complete binary tree (as seen in class).

(c) (6 points) Draw the Min-Heap Tree obtained by running an adaptation of the procedure `BuildMaxHeap()` seen in class. The adaptation consist in changing the procedure `MaxHeapify()` by `MinHeapify()` You are only required to show the final tree, although if you draw intermediate trees, please circle yo credit.

2. The array `A={27,14,18,9,7,33}` represents a complete binary tree (as seen in class).

(d) (9 points) An AVL is a self-balancing BST with the following balance condition: the difference between heights of left and right subtrees is not more than one for all nodes. After an insertion, some re-balancing operations are performed to re-establish (if needed) the balance condition. For the following exercise, we will relax the balance condition, and we will re-establish it as follows: the difference between heights of left and right subtrees is not more than **two** for all nodes. Draw the AVL Tree obtained by inserting the elements of `A` one by one in the order of the initial list considering the new balance condition. In other words, perform the standard AVL insertion algorithm; however, only restore the AVL properties when the relaxed balance condition (difference between heights of left and right subtrees is not more than **two** for all nodes) is violated. You are only required to show the final tree, although if you draw intermediate trees, you will be graded on your final result for ANY credit.

(e) (9 points) We have run a secret algorithm to generate a red-black tree composed by the elements of A (please see the tree below). Your task for this question is to insert the following 6 elements {28,3,11,35,10,17} in order as they appear in the list. You are only required to show the final red-black tree (please show the colors of the nodes), although if you draw intermediate trees, please circle your final result for ANY credit.



Algorithm Paradigms

During my years as instructor, I have been able to identify four main strategies that student use to prepare exams. Particularly, I have found relevant the following categories: *i*) The brute-force, *ii*) The divide and conquer, *iii*) ming and *iv*) The greedy strategy. For this exercise, we will identify what t usually use. In order to do that, we will consider this midterm as our test cas

This midterm is compos tures:

- A list of topics (i. $T_2, T_3 \ldots T_{14}$ that will be evaluated during the exam. In our case, remen ing the topics (lectures) from T_0 to T_{14}.
- A list of integral points (i.e., marks) $P = P_0, P_1, P_2, P_3 \ldots P_{14}$ that will be assigned (by the instructor) to each topic covered in the exam. In this case, the topic T_i will be assigned a total of P_i points for all $i \in \{0 \ldots 14\}$.
- A list of integral minutes $M = M_0, M_1, M_2, M_3 \ldots M_{14}$ that you will have to spend preparing (studying) each topic covered in the exam. In this case, you will need M_i minutes to correctly prepare the topic T_i for all $i \in \{0 \ldots 14\}$.

3. During the reading week, four Comp251 students (each student following a different strategy) made the decision to prepare the exam by studying together through a zoom meeting. The following (real) stories happened during their study sessions.

Story 1:

The four students made the decision to study the topics T (T as defined above) in increasingly order (i.e., from T_0 to T_{14}); however, the students noticed that the points array P (P as defined above) is not necessarily sorted in increasing order. For example, the students came up with the following possible values for the array $P = \{2, 1, 3, 2, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3\}$. This array implies that the topic T_0 will have a weight of 2 marks in the midterm, topic T_1 will have a weight of 1 mark in the midterm, topic T_2 will have a weight of 3 marks in the midterm and so on. The greedy-based student proposed the following challenge to the group. **Can you compute the number of topic pairs for which $P_i > P_j$ and $i < j$.** In the given P example, there are five (5) of those pairs. Particularly, for the following five pairs $\{P_0, P_1\}$, $\{P_0, P_4\}$, $\{P_2, P_3\}$, $\{P_2, P_4\}$ and $\{P_3, P_4\}$, $P_i > P_j$ and $i < j$.

(a) (10 points) The brute-force based student was in for the challenge. Please write down the (most efficient) complete-search program (using java code) that the student used to solve the challenge. Please complete the information in the function `challenge1_BF`

```
public int challenge1_BF(int[] P) {

}
```

}

(b) (10 points) At this point (and after seeing the proposed brute-force solution), the divide and conquer based student claimed that he could perform the same task in a faster fashion. Particularly, the student claimed that the task can be performed in `O(n log(n))` if a divide-and-conquer strategy is used. The student thinks that the algorithm `merge sort` can be modified to accomplish this task. We are providing to you, the modified version of the function `Merge(A,p,q,r)` (as seen in class). Please notice that the only change was the inclusion of a counter called `important_pair` and the change from a `void` procedure to a function returning and `int`. Your job for this exerci e modified version of the `MergeSort(A,p,r)` function (also as seen in lenge (and to accommodate the changes performed in `Merge(A,p,q,r)` eSort `(A,p,r)` must return an `int` representing the number of topic pai nd $i < j$. Please use pseudo-code notation in your answer (i.e., the same n ort `(A,p,r)` function seen in class).

Algorithm 1 Merge(A,p,

$n_1 \leftarrow q - p + 1$
$n_2 \leftarrow r - q$
for $i \leftarrow 1 to n_1$ **do**
  $L[i] \leftarrow A[p + i - 1]$
end for
for $j \leftarrow 1 to n_2$ **do**
  $R[j] \leftarrow A[q + j]$
end for
$L[n_1 + 1] \leftarrow \infty$
$R[n_2 + 1] \leftarrow \infty$
$i \leftarrow 1$
$j \leftarrow 1$
$important\_pair \leftarrow 0$
for $k \leftarrow ptor$ **do**
  if $L[i] \leq R[j]$ **then**
    $A[k] \leftarrow L[i]$
    $i \leftarrow i + 1$
  else
    $A[k] \leftarrow R[j]$
    $j \leftarrow j + 1$
    $important\_pair \leftarrow important\_pair + 1$
  end if
end for
return *important_pair*

(c) (5 points) In class, we studied the loop invariant property for the `Merge(A,p,q,r)` algorithm used by `MergeSort(A,p,r)`. Your job for this question is to establish a new loop invariant for the (provided) modified version of `Merge(A,p,q,r)`. Please, define your invariance in terms of the variable *important_pair*. Please define your invariant making sure that it must hold true during the initialization, maintenance and termination phases (no need to report the proofs). Also remember that the invariant should tell us something useful to understand the algorithm.

Story 2:

The four students start to discuss strategies about how to minimize the study-time and maximize the score gotten in the exam.

(d) (4 points) The greedy-based student told the others that he had other exams to prepare and that he will only have M_G minutes to study for the Comp251 midterm. Then, the student will try to maximize the number of topics that will be able to prepare given the time constraint. Please write down the greedy strategy that the student needs to implement in order to accomplish the task. In other words, given an array M representing a list of integral minutes $M = M_0, M_1, M_2, M_3 \ldots M_{14}$ that you will have to spend preparing (studying) each topic covered in the exam, and a time constraint M_G (i.e., maximum number of minutes that you can spend preparing the Comp251 topics), report (in words, pseudo code or mathematically) the greedy strategy that guarantee that the number of studied topics is maximized.

(e) (6 points) Given that the dynamic programming based student does not have a time constraint to prepare the exam, the student will try to come up with an algorithm to compute the minimum number of minutes required to earn at least p points on the exam. The greedy-based student proposed to use a greedy (what a surprise !!!) strategy to solve this problem. The exact idea was to sort (in decreasing order) the array P (a list of integral points (i.e., marks) $P = P_0, P_1, P_2, P_3 \ldots P_{14}$ that will be assigned to each topic covered in the exam). They will then iterate over the sorted array, choosing the topics until the value p is reached. The student claimed that by doing so, sooner the desired mark (i.e., the target score) having as consequence the minimum number of minutes.

Given the arrays P and p, please give a proof (if the claim is true) or provide a counter-example (if

(f) (11 points) The dynamic-programming student keeps with the idea that the best way to come up with an algorithm to compute the minimum number of minutes required to earn at least p points on the exam is by using dynamic programming (what a surprise !!!). In order to do that, the student defined the function $Minutes(i, p)$ to denote the minimum number of minutes needed to earn p points when you are restricted to selecting from topics 0 through i. Please give a recurrence expression for $Minutes(i, p)$. Please do not forget to include the base-cases.

(g) (2 points) What is the complexity of the DP algorithm (recursion) proposed in the previous step. Please let n be the total number of lectures.

A. Constant and it is bounded by O(p)
B. Linear and it is bounded by O(n)
C. Polynomial (quadratic) and it is bounded by O(n * p)
D. Pseudo-Polynomial and it is bounded by O(n * p)
E. Polynomial (cubic) and it is bounded by O(n * p * i)