University of M'Sila
Department of Computer Science
Instructor: Noureddine Amraoui

Level: Master 1 (IA and SIGL)
Course: Advanced Algorithms
Academic Year: 2023/2024

Final Exam (Solution with Scale)

(**Scale:** 20 Points, **Duration:** 1 Hour and 30 Minutes)

Single Choice Questions: (4 pts)

Check the ***only*** correct answer.

1. Binary Search is not considered as a pure Divide and Conquer method because:
☐ A. It has no worst-case scenario
☒ B. Only one sub-array is conquered after each division **(0.5)**
☐ C. It is not efficient for large input sizes
☐ D. All of the above

2. Which of the following is a FALSE statement about a spanning tree of a graph G?
☐ A. It is a subgraph of G
☐ B. It includes every vertex of G
☒ C. It can be cyclic **(0.5)**
☐ D. None of the above

3. What is the main drawback of an ideal hashing?
☐ A. Causes collision problem
☒ B. Requires too much space **(0.5)**
☐ C. Both A and B
☐ D. None of the above

4. Which of the following is an adaptive sorting algorithm?
☒ A. Bubble Sort **(0.5)**
☐ B. Selection Sort
☐ C. Merge Sort
☐ D. All of the above

5. Which problem from the following can be easily solved if the elements of a list are sorted?
☐ A. Searching for an element
☐ B. Identifying largest/smallest element
☐ C. Detecting duplicate values
☒ D. All of the above **(0.5)**

6. If we want to analyze the elements frequency of a list, which technique would make the problem easier?
☒ A. Sorting the list **(0.5)**
☐ B. Divide and Conquer
☐ C. Recursion
☐ D. None of the above

7. The time complexity of a sorting algorithms depends mostly on:
☐ A. Adaptability
☐ B. Number of swaps
☒ C. Number of comparisons **(0.5)**
☐ D. All of the above

8. Is it possible to make Selection Sort adaptive?
☐ A. Yes
☒ B. No **(0.5)**

Exercise 1: (6 pts)

1. For arr = [3, 3, 5, 6, 6, 7, 7, 8, 11, 12] and target = 21, algo it returns **False**: **(1.0)**

Index	0	1	2	3	4	5	6	7	8	9	
Elements	3	3	5	6	6	7	7	8	11	12	
Iteration #1	left									right	s = 15
Iteration #2		left								right	s = 15
Iteration #3			left							right	s = 17
Iteration #4				left						right	s = 18
Iteration #5					left					right	s = 18
Iteration #6						left				right	s = 19
Iteration #7							left			right	s = 19
Iteration #8								left		right	s = 20
Iteration #9									left	right	s = 23
End.									left = right		

For arr = [3, 3, 5, 6, 6, 7, 7, 8, 11, 12] and target = 13, algo returns **True**. **(1.0)**

Index	0	1	2	3	4	5	6	7	8	9	
Elements	3	3	5	6	6	7	7	8	11	12	
Iteration #1	left									right	s = 15
Iteration #2	left								right		s = 14
Iteration #3	left							right			s = 11
Iteration #4		left						right			s = 11
Iteration #5			left					right			s = 13
End.											

algo checks if there is any pair in the array that sums up to the given target. **(0.5)**

2. Total number of primitive instructions: $T(n) = \frac{13}{2}n + \frac{5}{2} = O(n)$ **(1.5)**

At each iteration, either left index is incremented or right index is decremented. So, the loop will be executed at most $n - 1$ times.

```python
def algo(arr, target):                    # T(n) = (13/2)n-5/2
    left, right = 0, len(arr) - 1         # 2
    while left < right:                   # (n-1)+1
        s = arr[left] + arr[right]        # 2(n-1)
        if s == target: return True       # 1(n-1)
        elif s < target: left += 1        # 3(n-1)/2 (on average)
        else: right -= 1                  # 2(n-1)/2 (on average)
    return False                          # 1
```

3. A naïve algorithm: $T(n) = O(n^2)$ **(1.5)**

```python
def algo2(arr, target):
    for i in range(len(arr)-1):
        for j in range(i+1, len(arr)):
            if arr[i]+arr[j] == target:
                return True
    return False
```

4. algo is more time efficient since $O(n) < O(n^2)$ **(0.5)**

Exercise 2: (5 pts)

1. Resultant HT: **(1.0)**

i	0	1	2	3	4	5	6	7	8	9	10	11	12
$HT(i)$	26	11			30							37	24

2. The hash table size is not sufficient **(1.0)**. In this case, there are two solutions:

- We increase the size of the hash table. In this case, all the elements must be inserted again. It is common to double the size of the hash table. **(0.5)**
- We use separate chaining. **(0.5)**

3. Resultant new HT: **(2.0)**

i	0	1	2	3	4	5	6	7	8	9	10	11	12
$HT(i)$	[26]	[1]	[2]	[3]	[4, 30]	[5]	[6]	[7]	[8]	[9]	[10]	[11, 24, 37]	[]

Exercise 3: (5 pts)

The problem is better known as Bin Packing Problem.

1. Strategy 1 is sequential; Strategy 2 is greedy. **(1.0 pt)**

2.

Following strategy 1, 5 boxes are required. **(0.75 pt)**

3 (3)	4 (4)	8 (2)		
1 (7)	2 (6)	5 (8)	6 (5)	7 (9)
Bin#1	**Bin2**	**Bin3**	**Bin4**	**Bin5**

Following the strategy 2, 4 boxes are required, only. **(0.75 pt)**

8 (2)	3 (3)	4 (4)	6 (5)
7 (9)	5 (8)	1 (7)	2 (6)
Bin1	**Bin2**	**Bin3**	**Bin4**

3. The second strategy gives an optimal solution as only 4 boxes are required. **(1.0 pt)**

4. In the second case, the strategy 2 gives a solution with 3 boxes required: **(0.75 pt)**

	5 (3)	
1 (5)	3 (3)	
2 (6)	4 (5)	6 (2)
Bin1	**Bin2**	**Bin3**

However, this is not an optimal solution since there is another one with 2 boxes required only: **(0.75 pt)**

5 (3)	6 (2)
3 (3)	4 (5)
2 (6)	1 (5)
Bin1	**Bin2**