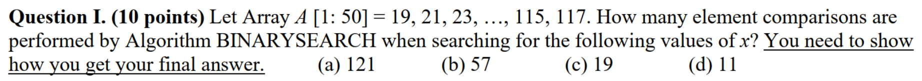
ICS 202 Assignment 1

Loay Shqair, 202365030

February 8, 2025

Question I



Question I. (10 points) Let Array *A* [1: 50] = 19, 21, 23, …, 115, 117. How many element comparisons are performed by Algorithm BINARYSEARCH when searching for the following values of *x*? You need to show how you get your final answer. (a) 121 (b) 57 (c) 19 (d) 11

(a)

First note that the element at index n is $f(n) = 17 + 2n$. Initialize $i = 1$ and $j = 50$, then $c = (1 + 50)/2 = 25$. $f(25) = 67 < 121$, so we keep searching. Now, $i = 25 + 1 = 26$, $j = 50$, and $c = (26 + 50)/2 = 38$. $f(38) = 93 < 121$, so keep searching. Now, $i = 38 + 1 = 39$, $j = 50$, and $c = (39 + 50)/2 = 44$. $f(44) = 105 < 121$, so keep searching. Now, $i = 44 + 1 = 45$, $j = 50$, $c = (45 + 50)/2 = 47$. $f(47) = 111 < 121$, so we keep searching. Now, $i = 47 + 1 = 48$, $j = 50$, and $c = (48 + 50)/2 = 49$. $f(49) = 115 < 121$, so we keep searching. Now, $i = 49 + 1 = 50$, $j = 50$, $c = (50 + 50)/2 = 50$. $f(50) = 119 < 121$. Now, $i = 50+1 = 51$ and $j = 50$. Since $i > j$, we terminate the algorithm. **121 was compared 6 times.** Note that $6 = \lceil \log_2 50 \rceil$.

(b)

Initialize $i = 1$ and $j = 50$, then $c = (1 + 50)/2 = 25$. $f(25) = 67 > 57$, so we keep searching. Now $i = 1$, $j = 25 - 1 = 24$, and $c = (1 + 24)/2 = 12$. $f(12) = 41 < 57$, so we keep searching. Now, $i = 12 + 1 = 13$, $j = 24$, and $c = (13 + 24)/2 = 18$. $f(18) = 53 < 57$, so we keep searching. Now, $i = 18 + 1 = 19$, $j = 24$, and $c = (19 + 24)/2 = 21$. $f(21) = 59 > 57$, so we keep searching. Now, $i = 19$, $j = 21 - 1 = 20$, and $c = (19 + 20)/2 = 19$. $f(19) = 55 < 57$, so we keep searching. Now, $i = 19 + 1 = 20$, $j = 20$, and $c = (20 + 20)/2 = 20$. $f(20) = 57 = 57$. We found the desired value. **57 was compared 6 times.**

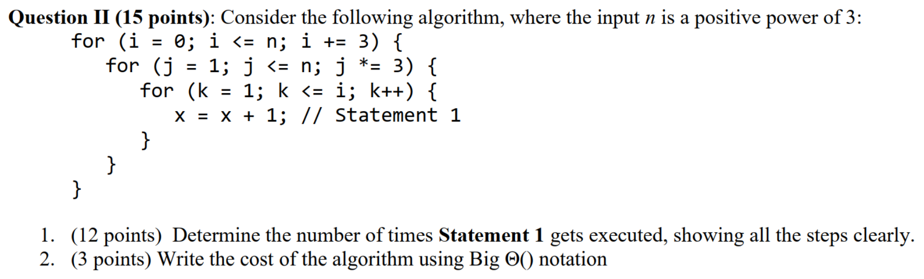
(c)

Initialize $i = 1$ and $j = 50$, then $c = (1 + 50)/2 = 25$. $f(25) = 67 > 19$, so we keep searching. Now $i = 1$, $j = 25 - 1 = 24$, and $c = (1 + 24)/2 = 12$. $f(12) = 41 > 19$, so we keep searching. Now, $i = 1$, $j = 12 - 1 = 11$, and $c = (1 + 11)/2 = 6$. $f(6) = 29 < 19$, so we keep searching. Now, $i = 1$, $j = 6 - 1 = 5$, and $c = (1 + 5)/2 = 3$. $f(3) = 23 > 19$, so we keep searching. Now, $i = 1$, $j = 3 - 1 = 2$, and $c = (1 + 2)/2 = 1$. $f(1) = 19 = 19$. We found the desired value. **19 was compared 5 times.**

(d)

Using our analysis from part (c). We know, after performing 5 comparisons, that we're going to end up with $i = 1$, $j = 2$, and $c = 1$. However the last comparison will not terminate the loop yet as $f(1) = 19 > 11$. We update the values and obtain $i = 1$, $j = 1 - 1 = 0$. Since $j < 1$, we terminate the loop. **11 will be compared to 5 values.**

Question II



Question II (15 points): Consider the following algorithm, where the input *n* is a positive power of 3:

```
for (i = 0; i <= n; i += 3) {
   for (j = 1; j <= n; j *= 3) {
      for (k = 1; k <= i; k++) {
         x = x + 1; // Statement 1
      }
   }
}
```

1. (12 points) Determine the number of times **Statement 1** gets executed, showing all the steps clearly.
2. (3 points) Write the cost of the algorithm using Big Θ() notation

1.

The innermost loop will be executed

$$\sum_{k=1}^{i} 1$$

If $n = 3^p$, then the middle loop will be executed $p+1$ times (for $j = 3^0, 3^1, \ldots, 3^p$). Our sum then becomes

$$\sum_{r=0}^{p} \sum_{k=1}^{i} 1$$

For the outermost loop, we assume $n = 3m$ (which is definitely the case if n is a power of 3 that is larger than 1). Then the outermost loop is executed $m + 1$

times for $i = 3 \cdot 0, 3 \cdot 1, \ldots, 3 \cdot m$. Our final sum becomes

$$\sum_{q=0}^{m} \sum_{r=0}^{p} \sum_{k=1}^{i} 1$$

where $i = 3q, p = \log_3 n$, and $m = n/3$. Substituting in all these values, we obtain

$$\sum_{q=0}^{\frac{n}{3}} \sum_{r=0}^{\log_3 n} \sum_{k=1}^{3q} 1 = \sum_{q=0}^{\frac{n}{3}} \sum_{r=0}^{\log_3 n} 3q = \sum_{q=0}^{\frac{n}{3}} (\log_3 n{+}1) 3q = 3(\log_3 n + 1) \frac{(\frac{n}{3})(\frac{n}{3}+1)}{2} = \frac{n(n+3)(\log_3 n + 1)}{6}$$

$$\boxed{\text{execution count} = \frac{n(n+3)(\log_3 n + 1)}{6}}$$

2.

$$\frac{n(n+3)(\log_3 n + 1)}{6} = \frac{1}{6}(n^2 \log_3 n + 3n \log_3 n + n^2 + 3n)$$

The dominant term in this expression is $\frac{1}{6} n^2 \log_3 n$. Dropping the constants, we find that our algorithm is

$$\boxed{\Theta(n^2 \lg n)}$$

Q III

Question III (15 points): Write the function $T(n) = n^{1/3} \log^2(n^2) + \sqrt{n \log n}$ in terms of Big $\Theta()$ notation. Prove your answer.

We evaluate the limit of the ratio of the two terms making up $T(n)$. If the limit approaches ∞, then the numerator term is dominant. Otherwise, we pick the denominator term as the dominant term.

$$\frac{n^{\frac{1}{3}} \log^2(n^2)}{\sqrt{n \log(n)}} = \frac{4n^{\frac{1}{3}} \log^2(n)}{\sqrt{n \log(n)}} = \frac{4n^{\frac{1}{3}} \log^2(n)}{n^{\frac{1}{2}} \log^{\frac{1}{2}}(n)} = \frac{4 \log^{\frac{3}{2}}(n)}{n^{\frac{1}{6}}}$$

By repetitive application of L'Hôpital's rule, we get

$$\lim_{n \to \infty} \frac{4 \log^{\frac{3}{2}}(n)}{n^{\frac{1}{6}}} = \lim_{n \to \infty} \frac{36 \log^{\frac{1}{2}}(n)}{n^{\frac{1}{6}}} = \lim_{n \to \infty} \frac{108}{n^{\frac{1}{6}} \log^{\frac{1}{2}}(n)} = 0$$

Thus, the denominator term $\sqrt{n \log(n)}$ is dominant and the function is

$$\boxed{\Theta\left(\sqrt{n \lg(n)}\right)}$$

Question IV

Question IV (20 points): Consider the following algorithm, where the input is two arrays of sizes n and m, respectively.

```
boolean arraysComparison(int[] arr1, int[] arr2){
    int n = arr1.length;
    int m = arr2.length;

    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++)
            if(arr1[i] > arr2[j] && arr1[i] - arr2[j] > 5)
                return true;
    }

    return false;
}
```

1. (6 points) What is the best case time complexity of this algorithm? Justify your answer, indicating which statement did you use to find the complexity.
2. (8 points) What is the worst case time complexity of this algorithm? Justify your answer, indicating which statement did you use to find the complexity.
3. (6 points) What is the space complexity of this algorithm? Justify your answer.

1.

The best case time complexity is $O(1)$ (constant time). This would be the case if the conditional statement inside the nested loop evaluates to true on the first try. For example, if arr1 $=$ [10, 0, 0] and arr2 $=$ [3, 0, 0].

2.

The worst case time complexity occurs when, for all n elements in arr1, we compare them with all m elements in arr2. Thus the worst case time complexity is $O(mn)$.

3.

At any time during the execution of the program, we're only storing the values of n, m, i, j. The amount of data we're storing does not depend on the lengths of the arrays, n and m. Thus the space complexity is $O(1)$.

Question V

Question V (10 points): Determine whether the following statements are true or false. If true, prove your answer. If false, provide a counter example.

1. If $f(n)$ is $\Theta(g(n))$, then $e^{f(n)}$is $\Theta(e^{g(n)})$, where e is a mathematical constant
2. $f(n)+g(n)$ is $\Omega(\max(f(n),g(n))$ for all functions *f(n)* and *g(n)* where they are both positive and non-decreasing.
3. b^{na} is $O(b^n)$, where a, b are constants.
4. $n^{\log n}$ is $O(2^n)$, where n is a positive integer, but only when n is sufficiently large.

1.

Let $f(n) = 2n$ and $g(n) = n$, then $f(n)$ is $\Theta(g(n))$. However, $e^{g(n)} = e^n$ and $e^{f(n)} = e^{2n} = (e^n)^2 = \left(e^{g(n)}\right)^2$. Clearly then, $e^{f(n)}$ is not $\Theta\left(e^{g(n)}\right)$. The statement is **False**.

2.

$f(n) + g(n)$ is $\Omega(\max(f(n), g(n))$ is equivalent to

$$c \cdot \max(f(n), g(n)) \leq f(n) + g(n), \forall n \geq N$$

if $\max(f(n), g(n)) = f(n)$, then

$$cf(n) \leq f(n) + g(n)$$

If we let $c = 1$, then it is true since $g(n)$ is positive. On the other hand, if $\max(f(n), g(n)) = g(n)$, then

$$cg(n) \leq f(n) + g(n)$$

if we let $c = 1$, then it also holds true since $f(n)$ is positive. Thus, the statement is **True**.

3.

b^{na} is $O(b^n)$ is equivalent to

$$b^{na} \leq cb^n, \forall n \geq N$$

Divide by b^n to get

$$b^{n(a-1)} \leq c, \forall n \geq N$$
$$(b^{a-1})^n \leq c, \forall n \geq N$$

For $a > 1$, no matter what value of c we choose, it will have to be greater than $(b^{a-1})^n$ for all $n > N$ which is impossible. The statement is **False**.

4

$n^{\log n}$ being $O(2^n)$ is equivalent to

$$n^{\log n} \leq c2^n, \forall n \geq N$$

Taking the log of both sides

$$\log(n^{\log n}) \leq \log(c2^n), \forall n \geq N$$
$$\log^2(n) \leq \log(c) + n\log(2), \forall n \geq N$$

which we know is true as $\log^p(n)$ is $O(n)$. Since our steps are reversible, the statement is **True**.

Question VI

Definition: $f(n)$ is said to be in small $o(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.
The small $o()$ notation is usually used to determine whether two functions belong to two different complexity classes or not.

Question VI (15 points): Fill in the following blanks with either T for true or F for false:

$f(n)$	$g(n)$	$f = O(g)$	$f = \Omega(g)$	$f = \Theta(g)$	$f = o(g)$
$1000n^{3/2}$	$10 + n^2 + n$				
$\sqrt{n}\log^2 n$	$n \log n$				
2^n	n^n				
n^n	3^{3n}				

$f(n)$	$g(n)$	$f = O(g)$	$f = \Omega(g)$	$f = \Theta(g)$	$f = o(g)$
$1000n^{3/2}$	$10 + n^2 + n$	T	F	F	T
$\sqrt{n}\log^2 n$	$n \log n$	T	F	F	T
2^n	n^n	T	F	F	T
n^n	3^{3n}	F	T	F	F

Question VII

Question VII (15 points): Given the code segment below and that n and x are the input, where n is a power of 3, answer the following questions:

```
// . . .
int total = 0;

for (int i = 0; i < n * n; i++)
    total++;          // statement1

if (x < 3) {
    for (int i = 1; i <= n; i++)
        for (int j = 0; j < n - i; j++)
            total++;     // statement2
}
else {
    for (int i = 1; i <= n * n; i *= 5)
        total++;         // statement3
}

for (int i = 1; i <= n * n; i *= 4)
    for (int j = 1; j <= i / 2; j++)
        total++;         // statement4
```

1) (13 points) Find the number of times `statement1`, `statement2`, `statement3` and `statement4` get executed, showing all the details of your solution, if
   a) `x = 2`
   b) `x = 5`

2) (1 points) Determine the Big-Θ() complexity of this program fragment in the best case.
3) (1 points) Determine the Big- Θ() complexity of this program fragment in the worst case.

1)

a)

statement 1:

$$\sum_{i=0}^{n^2-1} 1 = n^2, O(n^2)$$

statement 2:

$$\sum_{i=1}^{n}\sum_{j=0}^{n-i-1} 1 = \sum_{i=1}^{n}(n-i) = \sum_{i=1}^{n} n - \sum_{i=1}^{n} i = n^2 - \frac{n(n+1)}{2} = \frac{n(n-1)}{2}, O(n^2)$$

statement 3

since the if-statement of statement 2 evaluates to true, the else block will not execute.

$$0$$

statement 4

$$\sum_{r=0}^{\lfloor \log_4 n^2 \rfloor}\sum_{j=1}^{4^r/2} 1 = \sum_{r=0}^{\lfloor \log_4 n^2 \rfloor} 4^r/2 = \frac{4^{\lfloor \log_4 n^2 \rfloor + 1} - 1}{6}, O(n^2)$$

b)

statement 1:

$$\sum_{i=0}^{n^2-1} 1 = n^2, O(n^2)$$

statement 2:

The conditional evaluates to false

$$0$$

statement 3

$$\sum_{r=0}^{\lfloor \log_5 n^2 \rfloor} 1 = \lfloor \log_5 n^2 \rfloor + 1, O(\lg n)$$

statement 4

$$\sum_{r=0}^{\lfloor \log_4 n^2 \rfloor}\sum_{j=1}^{4^r/2} 1 = \sum_{r=0}^{\lfloor \log_4 n^2 \rfloor} 4^r/2 = \frac{4^{\lfloor \log_4 n^2 \rfloor + 1} - 1}{6}, O(n^2)$$

2)

In the best case scenario, the dominant terms are $\Theta(n^2)$.

3

In the worst case scenario, the dominant terms are also $\Theta(n^2)$.