

**TASK 4**

**Ahmed Adel Hassan 1900311**

**Hassan Ahmed Fathy El bahnasy 1901371**

Contents

[I. Introduction 2](#_Toc132834070)

[II. Assumptions 2](#_Toc132834071)

[III. Problem Description 2](#_Toc132834072)

[IV. (a) Can this be done before a feast scheduled in 5 weeks? 2](#_Toc132834073)

[V. (b) Can the king achieve his goal with just eight slaves? 2](#_Toc132834074)

[VI. Pseudo-code 4](#_Toc132834075)

[VII. The description of Pseudo-code 4](#_Toc132834076)

[VIII. the implementation of the Divide and Conquer algorithm in Java 5](#_Toc132834077)

[IX. The description of the java code 7](#_Toc132834078)

[X. Complexity Analysis 7](#_Toc132834079)

[XI. A comparison between your algorithm and at least one other technique that can be used to solve the problem (brute force) 7](#_Toc132834080)

[I. Pseudo-code of brute force 8](#_Toc132834081)

[II. Description of brute force 8](#_Toc132834082)

[III. Complexity Analysis of brute force 8](#_Toc132834083)

[IV. Comparison between brute force and divide and conquer 8](#_Toc132834084)

Table of Figures:  
[Figure 1:Screenshot of Output 1 6](#_Toc132903293)

[Figure 2:Screenshot of Output 2 6](#_Toc132903294)

[Figure 3:Screenshot of Output 3 6](#_Toc132903295)

# Introduction

In this problem, we are given 1000 wine barrels and one of them is poisoned. The poison is potent enough to kill a person within 30 days. The king is willing to sacrifice 10 slaves to determine the poisoned barrel before a feast scheduled in 5 weeks. We need to design a divide and conquer algorithm to solve this problem and analyze its complexity. We will also compare our algorithm with another technique that can be used to solve this problem.

# Assumptions

We assume that:

* The poison is evenly distributed in the poisoned barrel.
* The potency of the poison remains constant over time.
* We can sacrifice slaves to test the wine barrels.

# Problem Description

We are given 1000 wine barrels, one of which is poisoned with a potent poison that can kill a person in exactly 30 days. The poison is so potent that even a minuscule amount of it, no matter how diluted, will cause death within the given time frame. We need to determine the poisoned barrel using a divide and conquer algorithm, and we are allowed to sacrifice 10 slaves for this purpose. Additionally, we need to determine if we can solve this problem before a feast scheduled in 5 weeks, and if it is possible to do so using only 8 slaves.

# (a) Can this be done before a feast scheduled in 5 weeks?

Yes, the poisoned barrel can be determined before the feast scheduled in 5 weeks.

The king has 1000 barrels and is ready to sacrifice 10 slaves to determine the poisoned barrel. If the king gives each slave one wine barrel to taste, then the worst-case scenario is that the poisoned barrel is the last one, and all 10 slaves will die in 30 days. The king can then determine the poisoned barrel on the 30th day, well before the feast scheduled in 5 weeks.

However, if we use the divide-and-conquer approach, we can minimize the number of slaves required and determine the poisoned barrel even earlier.

# (b) Can the king achieve his goal with just eight slaves?

Yes, the king can determine the poisoned barrel with just eight slaves.

The divide-and-conquer approach can be used to determine the poisoned barrel with a smaller number of slaves. We can divide the 1000 barrels into two groups of 500 each and give one group to four slaves and the other group to another four slaves. Each slave tastes 125 wine barrels (half of 250). If none of them die after 30 days, then the poisoned barrel is in the remaining 500 barrels.

Next, we can divide the remaining 500 barrels into two groups of 250 each and give one group to two slaves and the other group to another two slaves. Each slave tastes 125 wine barrels (half of 250). If none of them die after 30 days, then the poisoned barrel is in the remaining 250 barrels.

Next, we can divide the remaining 250 barrels into two groups of 125 each and give one group to one slave and the other group to another slave. Each slave tastes 62 wine barrels (half of 125). If none of them die after 30 days, then the poisoned barrel is in the remaining 125 barrels.

Finally, we can give one barrel to each of the last two slaves. One barrel contains the poisoned wine, so one of the slaves will die after 30 days, and the poisoned barrel can be determined.

# Pseudo-code

DivideAndConquer(n, s, e):

if s > e

return -1

mid = (s + e) // 2

poison\_day = TestWineBarrels(s, e, mid)

if poison\_day == -1

return -1

elif s == e

return s

else

left\_result = DivideAndConquer(n, s, mid)

right\_result = DivideAndConquer(n, mid + 1, e)

if left\_result == -1:

return right\_result

else:

return left\_result

# The description of Pseudo-code

We start by dividing the 1000 wine barrels into two groups of 500 each. We use one slave to test each group of 500 wine barrels. If a slave dies, we know that the poisoned barrel is in that group, and we can continue dividing that group further. We can repeat this process until we narrow down the search space to one barrel.

At each step of the recursion, we test the middle barrel of the group. If the slave dies before 30 days, we know that the poisoned barrel is in the first half of the group. Otherwise, we know that the poisoned barrel is in the second half of the group. We can then recursively apply the same approach to the appropriate half of the group until we find the poisoned barrel.

We can terminate the recursion when we reach a group of one barrel, and we can return the index of the poisoned barrel.

# the implementation of the Divide and Conquer algorithm in Java

import java.util.Random;

import java.util.random.RandomGenerator;  
  
public class WineBarrel {

// Method to test wine barrels for poison

public static int testWineBarrels(int[] barrels, int start, int end, int day) {

int count = 0;

int poisonedIndex = -1;

for (int i = start; i <= end; i++) {

if (barrels[i] != i+1) {

count++;

poisonedIndex = i;

}

}

if (count == 0) {

return 0;

}

if (count > 10) {

return -1;

}

return poisonedIndex;

}

// Divide and Conquer algorithm to find poisoned barrel

public static int divideAndConquer(int[] barrels, int start, int end) {

if (start > end) {

return -1;

}

int mid = (start + end) / 2;

int poisonIndex = testWineBarrels(barrels, start, end, mid);

if (poisonIndex == -1) {

return -1;

} else if (start == end) {

return start;

} else {

int leftResult = divideAndConquer(barrels, start, mid);

int rightResult = divideAndConquer(barrels, mid+1, end);

if (leftResult == -1 && rightResult == -1) {

return -1;

} else if (leftResult == -1) {

return rightResult;

} else if (rightResult == -1) {

return leftResult;

} else {

return poisonIndex;

}

}

}

public static void main(String[] args) {

int[] barrels = new int[1000];

for (int i = 0; i < 1000; i++) {

barrels[i] = i+1;

}

// Replace one of the barrels with a poisoned barrel

Random rn = new Random();

int x = rn.nextInt(1000) + 1;

barrels[x] = 30;

int poisonedBarrel = divideAndConquer(barrels, 0, 999);

System.out.println("The poisoned barrel is: " + poisonedBarrel);

}

}

# The Screenshots of The Output

Graphical user interface, text

Description automatically generated with medium confidence

Figure 1:Screenshot of Output 1

Graphical user interface, text, application

Description automatically generated

Figure 2:Screenshot of Output 2

Graphical user interface, text

Description automatically generated

Figure 3:Screenshot of Output 3

# The description of the java code

The testWineBarrels method tests a range of wine barrels for poison given a certain day. It returns the poison day if the number of poisoned barrels is less than or equal to 10, 0 if there are no poisoned barrels in the range, and -1 if there are more than 10 poisoned barrels in the range.

The divideAndConquer method is the main algorithm that recursively divides the wine barrels into halves and tests them for poison using the testWineBarrels method. If the poison day is found, it returns the index of the poisoned barrel. If the poison day is not found, it returns -1.

In the main method, we create an array of 1000 wine barrels and randomly select one to poison. We then call the divideAndConquer method to find the poisoned barrel, and print the result

# Complexity Analysis

The time complexity of the given code is O(n\*log n), where n is the number of wine barrels.

The function “divideAndConquer” uses a binary search algorithm to find the poisoned wine barrel. At each step, it divides the search range into two halves and recursively searches one of the halves based on the result of testing the midpoint. This process continues until either the poisoned wine barrel is found or the search range is reduced to a single wine barrel. The time complexity of the binary search algorithm is O(log n).

The “testWineBarrels” function takes constant time to test each wine barrel on a given day, and it is called at most log(n) times during the binary search algorithm. Therefore, the total time complexity of the algorithm is O(n\*log n).

The time complexity of the “divideAndConquer” method can be expressed by the recurrence relation T(n) = 2T(n/2) + O(n), where n is the number of wine barrels. This is because the method recursively divides the array in half, and then performs a linear-time operation (calling” testWineBarrels”) on each half. By applying the Master theorem, we can see that the time complexity of this method is O(n log n).

**Therefore, the overall time complexity of the code is O(n log n).**

# A comparison between your algorithm and at least one other technique that can be used to solve the problem (brute force)

One technique that can be used to solve this problem is the brute force approach. In this approach, we would test each of the 1000 wine barrels with a slave until we find the poisoned barrel. This would require 1000 tests and the sacrifice of 10 slaves.

## Pseudo-code of brute force

BruteForce(n):

for i = 0 to n - 1

poisoned = TestWineBarrels(i, i, 30)

if poisoned != 0

return i

return -1

## Description of brute force

We start by testing the first wine barrel with a slave. If the slave dies, we know that the poisoned barrel is the first barrel. Otherwise, we move on to test the second barrel, and so on until we find the poisoned barrel.

At each step, we test one barrel with a slave for 30 days. If the slave dies during this period, we know that the barrel is poisoned. If the slave survives, we move on to test the next barrel.

We terminate the loop when we find the poisoned barrel and return its index. If we have tested all 1000 barrels and none of them are poisoned, we return -1 to indicate that there is no poisoned barrel.

## Complexity Analysis of brute force

The brute force algorithm has a time complexity of O(n), where n is the number of wine barrels. We need to test each barrel one by one until we find the poisoned one, so the worst-case time complexity is O(n).

The algorithm has a space complexity of O(1) because we do not use any additional space to solve the problem.

## Comparison between brute force and divide and conquer

The divide and conquer algorithm has a better time complexity of O(log n) compared to the brute force algorithm's O(n). This is because the divide and conquer algorithm can narrow down the search space quickly by dividing the barrels into smaller groups, while the brute force algorithm has to test each barrel individually.

However, the brute force algorithm requires only 10 slaves, while the divide and conquer algorithm requires more slaves because we need to test each group of barrels with a slave. In fact, the divide and conquer algorithm requires at least log2(1000) = 10 slaves to solve the problem, which is the same number of slaves required by the brute force algorithm.

Therefore, the divide and conquer algorithm is faster but requires more slaves, while the brute force algorithm is slower but requires fewer slaves. The choice between the two approaches depends on the trade-off between speed and the number of slaves that can be sacrificed.