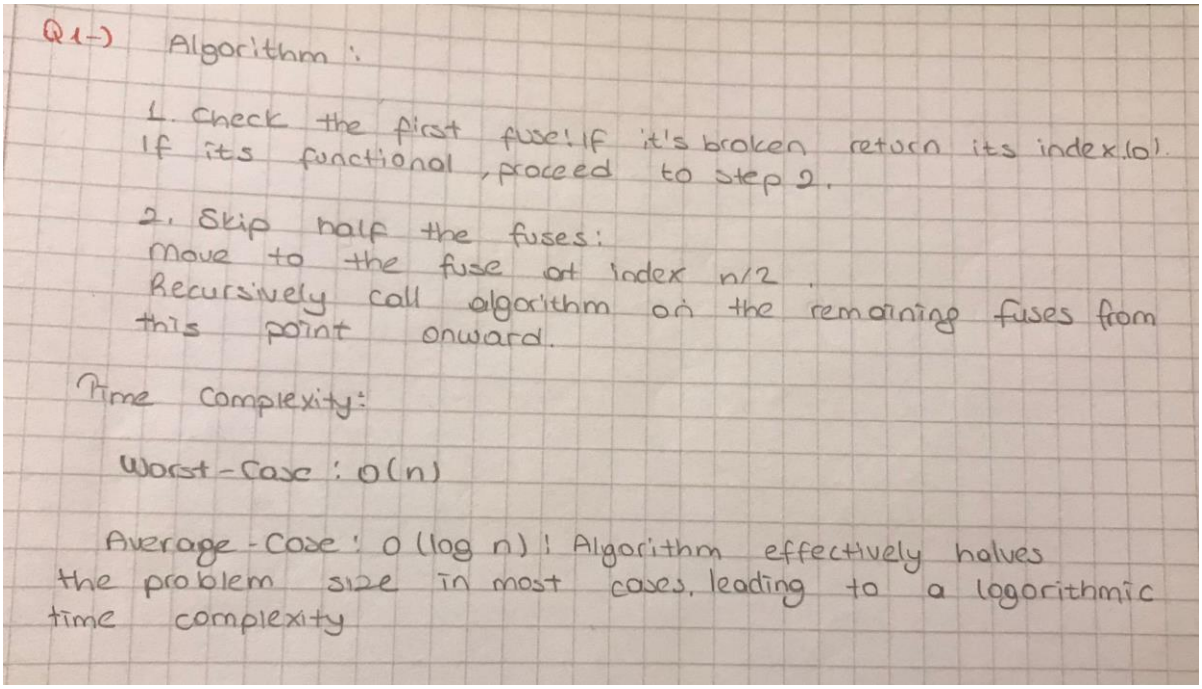
CSE 321 Introduction to Algorithm Design

Tuba TOPRAK- 161044116

Hw2

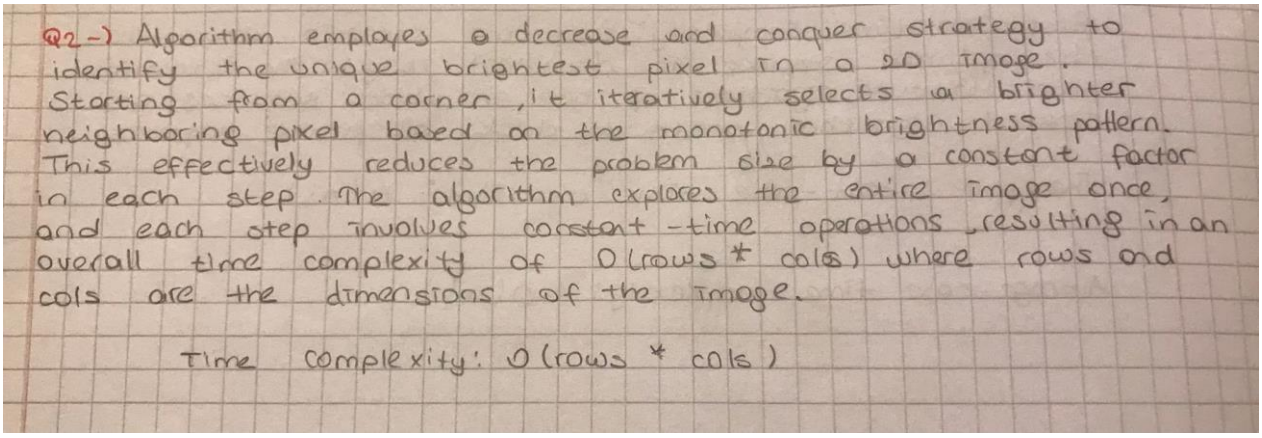
Please note that for Questions 1, 2, and 3, you are expected to provide solutions using the decrease and conquer approach, not the divide and conquer method. In the decrease and conquer approach, the problem sizes are reduced by a constant, not by dividing a factor.

1. In a complex electronic circuit, n fuses are sequentially connected to transfer electricity to various circuit components. Unfortunately, one fuse is malfunctioning and disabling the circuit's operation. The healthy fuses function normally. Develop an algorithm to identify the flawed fuse while minimizing the number of fuses inspected. Please note that electricity can only be transferred until the broken fuse and cannot reach the remaining fuses. Provide a decrease and conquer algorithm to find the flawed fuse. Include the pseudo-code and analyze the time complexity (Big-O notation) of your proposed algorithm.



2. You are required to analyze an image represented as a 2D grid of pixels, where each pixel holds a numerical value representing its brightness level. The image possesses a unique property where the brightness of pixels follows a strictly monotonic pattern between neighboring pixels: it monotonically increases or decreases between adjacent pixels. Only one pixel in the image breaks this monotonicity and is the brightest among its neighbors. Design an algorithm

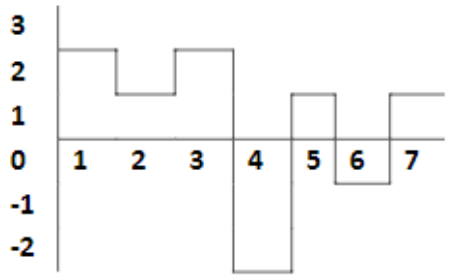
using a decrease and conquer strategy to efficiently identify this unique brightest pixel that is brighter than all of its four immediate neighbors (top, bottom, left, and right), considering the monotonicity in brightness levels between pixels. Provide the pseudo code and analyze the time complexity (Big-Oh notation) of your proposed algorithm.


Q2-) Algorithm employes a decrease and conquer strategy to identify the unique brightest pixel in a 2D image. Starting from a corner, it iteratively selects a brighter neighboring pixel based on the monotonic brightness pattern. This effectively reduces the problem size by a constant factor in each step. The algorithm explores the entire image once, and each step involves constant-time operations, resulting in an overall time complexity of O(rows * cols) where rows and cols are the dimensions of the image.

Time complexity: O(rows * cols)

3. Imagine a discrete function $f(x)$ defined over the interval $[0, n]$, where n denotes a positive integer. This function $f(x)$ spans the range of positive and negative values within this interval. Your objective is to devise a decrease and conquer algorithm to pinpoint the subsequent interval that results in the largest total area under the function $f(x)$. Notably, this total area encompasses both positive and negative values. Negative values within the function will diminish the total area, impacting the overall calculation of the maximal area under the curve. Develop an algorithm utilizing the decrease and conquer strategy to efficiently identify the interval within $[0, n]$ that produces the maximal total area under the discrete function $f(x)$, considering the effects of both positive and negative values on the overall area calculation. Provide the pseudo code and analyze the time complexity (Big-Oh notation) of your proposed algorithm.

Example:



The area between $[3, 5]$ is $= 2 + (-2) + 1 = 1$.
The maximum total area is $[0, 3] = 2 + 1 + 2 = 5$.

Q3-) Algorithm is a divide and conquer algorithm that finds the maximum total area of a function over the interval [0,n].

Base case: IF n<=1, then the interval [0,n] has only one element, so that element has the maximum area.

Inductive step: If n>1 then algorithm does the following:

- Calculates the maximum total area of the function over the interval [0,n-1]
- " " " [0,n-2]
- " " " [n-1,n]
- Chooses the largest of these three areas.

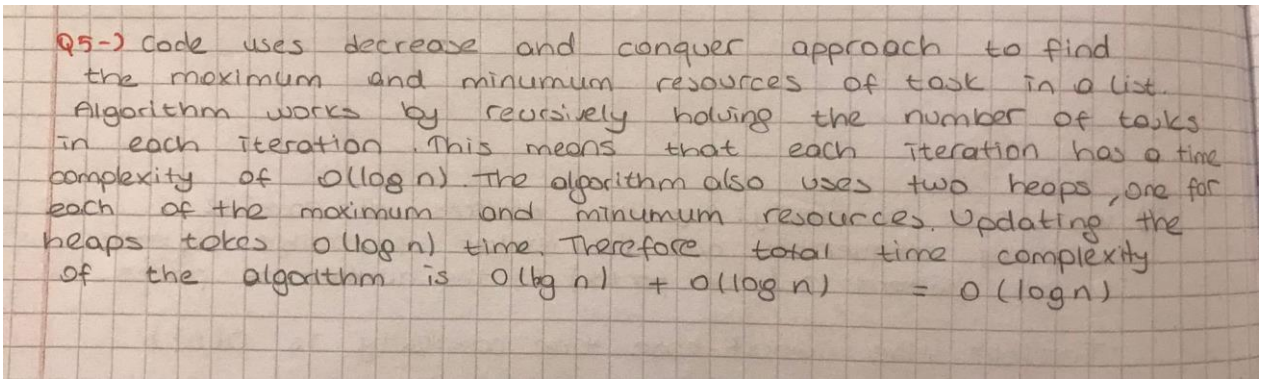
Time complexity is O(n * log n) because it reduces the problem by a factor of 2 in each iteration. This allows the algorithm to efficiently explore all possible solutions and find the optimal solution.

4. The network topology is represented as a graph, where each node represents a location, and edges between nodes signify possible connections or links. Each edge is associated with a certain latency. Develop an exhaustive search algorithm that exhaustively explores all possible routing paths between the source and destination nodes, evaluating the total latency incurred along each path. The algorithm must systematically examine all potential paths and determine the one with the minimum latency. Provide the pseudo code and analyze the time complexity (Big-Oh notation) of your proposed algorithm.

Q4-) Solution involves a decrease and concuquer algorithm that finds the lowest-latency path from a source node to a destination node, based on a graph representing a network topology.
The code includes a graph class responsible for mointing graph data and adding edges.
- The exhaustive_search function implements a decrease and concuquer algorithm. It systematically explores all possible paths by visiting one node at a time.
- the time complexity of the algorithm can be expressed as O(b^d), where b is the average number of neighbors and d is the depth of the recursive tree. This arises from the need to explore all possible path.

Time Complexity : $O(b^d)$

5. As a project manager overseeing multiple endeavors, you encounter a critical challenge in efficient resource allocation across diverse project tasks. Each task demands varying resources and contributes uniquely to project progress. Design a divide and conquer algorithm that efficiently evaluates resource distribution across project tasks, simultaneously identifying the tasks demanding the maximum and minimum resources within a single execution. Provide the pseudo code and analyze the time complexity (Big-Oh notation) of your proposed algorithm.


Q5-) Code uses decrease and conquer approach to find the maximum and minumum resources of task in a list. Algorithm works by recursively holving the number of tasks in each iteration. This means that each iteration has a time complexity of O(log n). The algorithm also uses two heaps, one for each of the maximum and minumum resources. Updating the heaps takes O(log n) time. Therefore total time complexity of the algorithm is O(log n) + O(log n) = O(log n)