CSA0695 – DESIGN AND ANALYSIS OF ALGORITHMS FOR OPEN ADRESSING TECHNIQUES

Minimum Number of Groups to Create a Valid Assignment

Greedy Algorithm

Supervisor
Dr. Dhanalakshmi

Done by
Sk Khader basha
Reg no: 192210705

Problem statement

- You are given a 0-indexed integer array nums of length n. We want to group the indices so for each index i in the range [0, n - 1], it is assigned to exactly one group. A group assignment is valid if the following conditions hold: For every group g, all indices i assigned to group g have the same value in nums. For any two groups g1 and g2, the difference between the number of indices assigned to g1 and g2 should not exceed 1.
- Return an integer denoting the minimum number of groups needed to create a valid group assignment. Example 1: Input: nums = [3,2,3,2,3] Output: 2 Explanation: One way the indices can be assigned to 2 groups is as follows, where the values in square brackets are indices: group 1 -> [0,2,4] group 2 -> [1,3] All indices are assigned to one group. In group 1, nums[0] == nums[2] == nums[4], so all indices have the same value.
- In group 2, nums[1] == nums[3], so all indices have the same value. The number of indices assigned to group 1 is 3, and the number of indices assigned to group 2 is 2. Their difference doesn't exceed 1. It is not possible to use fewer than 2 groups because, in order to use just 1 group, all indices assigned to that group must have the same value. Hence, the answer is 2.

ABSTRACT

In this problem, we are given an integer array nums and tasked with grouping the indices such that each group satisfies two conditions

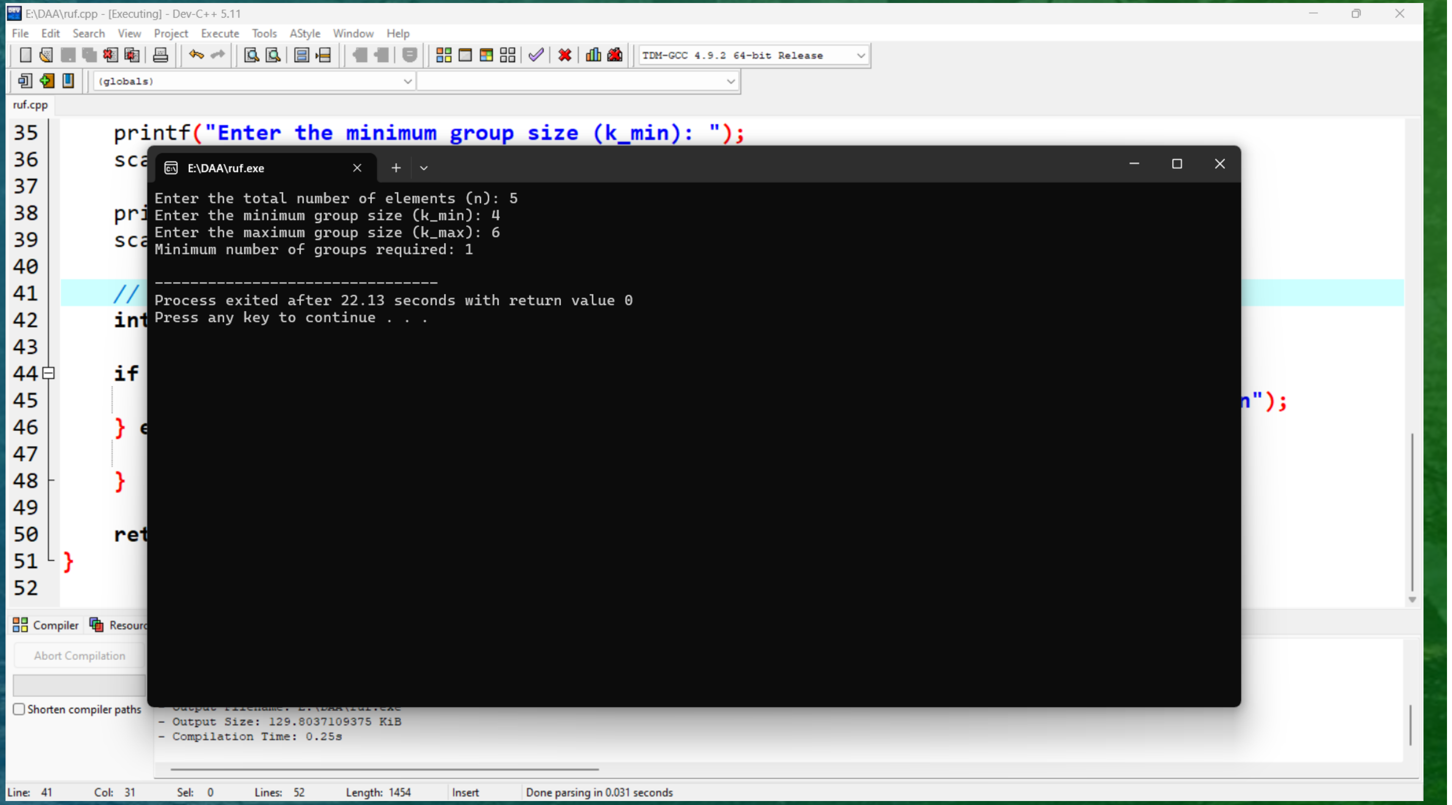
- All indices in a group have the same value in the array.
- The difference between the number of indices in any two groups does not exceed 1.
- The objective is to determine the minimum number of such groups required to achieve a valid grouping. This problem combines elements of grouping with constraints on the balance of group sizes and uniformity of values within groups

INTRODUCTION

- Grouping and partitioning problems are fundamental in various fields, such as education, resource allocation, and computational theory. In many real-world scenarios, it is often necessary to divide a set of elements into smaller groups while adhering to specific rules or constraints.
- The problem of determining the minimum number of groups for a valid assignment becomes complex when additional constraints are introduced.
- These constraints may include maintaining group sizes within a specified range, ensuring balance across groups, and preventing conflicts between incompatible elements.
- Such conditions require careful planning and a systematic approach to guarantee that all elements are appropriately assigned without violating the defined rules.
- This paper aims to address the issue by presenting a framework for finding the minimum number of groups needed for a valid assignment.

SAMPLE OUTPUT

Sample output for Minimum Number of Groups to Create a Valid Assignment



COMPLEXITY ANALYSIS:

- **Time Complexity**: The time complexity of the dynamic programming approach for determining the minimum number of groups required is O(n*(Kmax-Kmin+1)), where n is the total number of elements Kmin and Kmax define the range of permissible group sizes.

- **Space Complexity**: The space complexity is O(n), which corresponds to the size of the `dp` array used to store the minimum number of groups needed for each number of elements. This approach is efficient and suitable for practical scenarios where n and the range of group sizes are manageable.

- **BEST CASE :** In the best case scenario, the range of possible group sizes is such that the minimum number of groups required can be achieved quickly. For example, if kmax is very close to n and kmin is not too restrictive

- **Worst Case:** In the worst-case scenario, the number of group sizes to evaluate is large. Specifically, if the range between kmin and kmax is substantial, then each element count from 1 to n requires evaluating many possible group sizes.

- **Average Case:** The average case time complexity is influenced by the typical range of possible group sizes and the distribution of the number of elements. In practical scenarios, the group size range [kmin, kmax] often falls within a moderate range relative to n.

FUTURE SCOPE

- The future scope for improving the minimum number of groups to create a valid assignment includes exploring more efficient algorithms.

- Advancements could involve integrating parallel and distributed computing techniques to enhance scalability.

- Applying these methods to real-world case studies in fields like education, healthcare, and project management .

CONCLUSION

- In conclusion, the dynamic programming approach to determining the minimum number of groups required to partition n elements into groups of sizes between Kmin and Kmax is efficient and practical. By using a `dp` array to store intermediate results and applying a recurrence relation to update these values, the algorithm effectively handles the constraints, with a time complexity is O(n*(Kmax-Kmin+1)), and a space complexity of O(n).


THANK YOU!