SAVEETHA SCHOOL OF ENGINEERING

SIMATS, CHENNAI - 602105

CSA0695-DESIGN ANALYSIS AND ALGORITHMS FOR OPEN ADDRESSING TECHNIQUES

Minimum Number of Groups to Create a Valid Assignment

Guided BY,
Dr. R. Dhanalakshmi

Presented by:-
B.Manoj Kumar (192211695)

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.

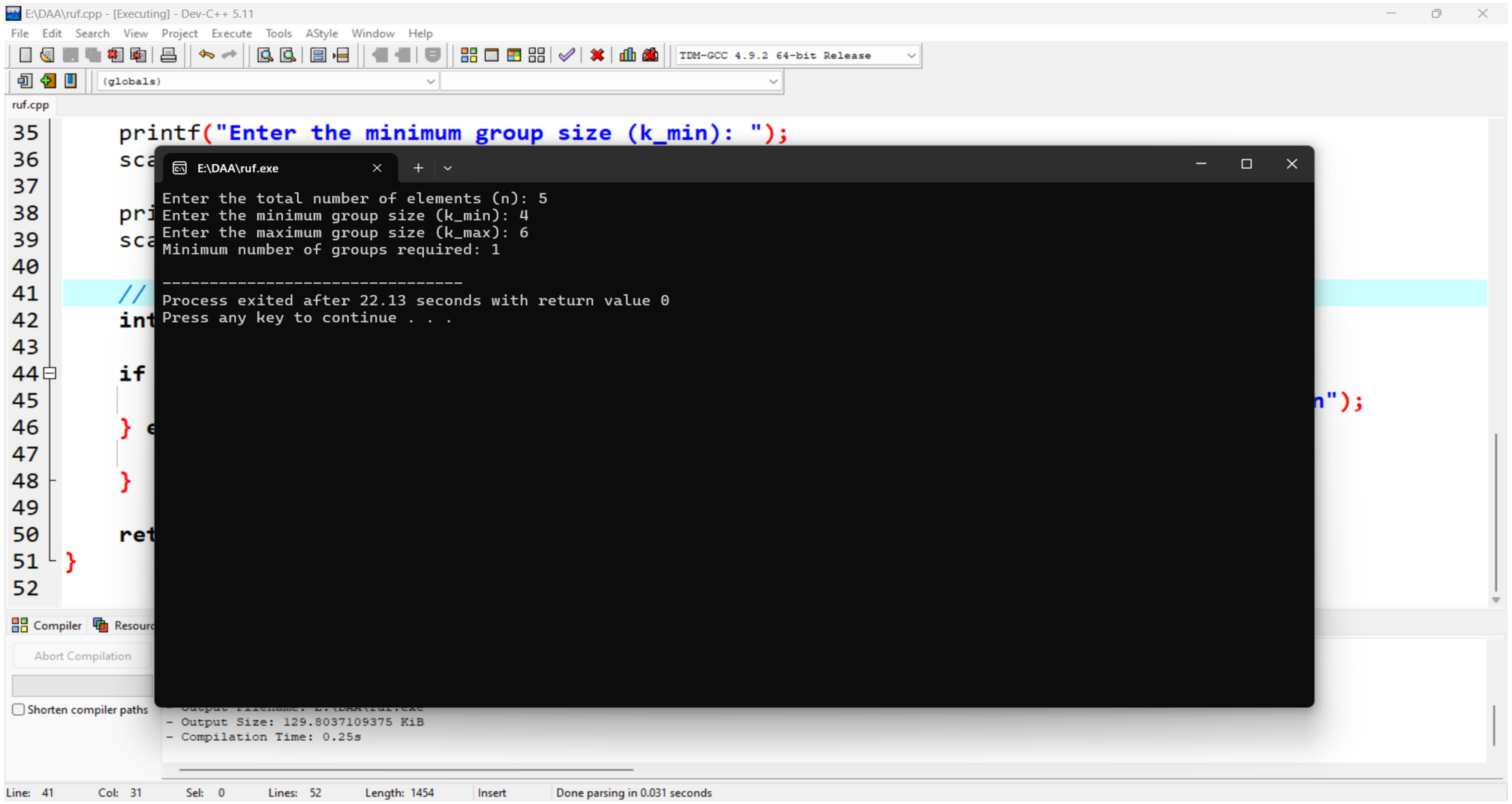
ABSTRACT:

This paper explores the problem of finding the minimum number of groups required to create a valid assignment of n elements into groups. Each group must adhere to predefined constraints, such as a minimum and maximum group size. Additionally, optional constraints like element compatibility, homogeneity, and balance across groups may apply. The objective is to develop a strategy to efficiently determine the minimum number of groups needed while satisfying all relevant conditions.

INTRODUCTION:

This paper aims to address the issue by presenting a framework for finding the minimum number of groups needed for a valid assignment. The study explores common constraints, such as minimum and maximum group size, homogeneity within groups, and conflict avoidance between incompatible elements. Through this exploration, we aim to develop efficient methods that can be applied to a wide range of grouping problems, offering practical solutions for educators, managers, and data scientists alike.

SAMPLE OUTPUT:


E:\DAA\ruf.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
ruf.cpp
E:\DAA\ruf.exe
Compiler
Abort Compilation
Shorten compiler paths
Line: 41 Col: 31 Sel: 0 Lines: 52 Length: 1454 Insert Done parsing in 0.031 seconds

```
    printf("Enter the minimum group size (k_min): ");
```

```
Enter the total number of elements (n): 5
Enter the minimum group size (k_min): 4
Enter the maximum group size (k_max): 6
Minimum number of groups required: 1

--------------------------------
Process exited after 22.13 seconds with return value 0
Press any key to continue . . .
```

```
- Output Size: 129.8037109375 KiB
- Compilation Time: 0.25s
```

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.

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.

Future Scope:

The future scope for improving the minimum number of groups to create a valid assignment includes exploring more efficient algorithms to handle larger datasets and complex constraints, such as dynamic or multi-dimensional constraints. Advancements could involve integrating parallel and distributed computing techniques to enhance scalability, as well as incorporating machine learning to predict optimal groupings based on historical data. Additionally, applying these methods to real-world case studies in fields like education, healthcare, and project management could refine the approach and demonstrate its practical applicability in various industries.

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). This method balances computational efficiency with accuracy, making it well-suited for real-world applications involving resource allocation, scheduling, and organizational tasks.

Thank you