



CSA0695 Design and Analysis of Algorithms for Open Addressing Techniques

capstone project

Splitting a String Into Descending Consecutive Values

Presented by: E.Leeladhar sai(192211479)

guided by: Dr. R. Dhanalakshmi

Splitting a String Into Descending Consecutive Values

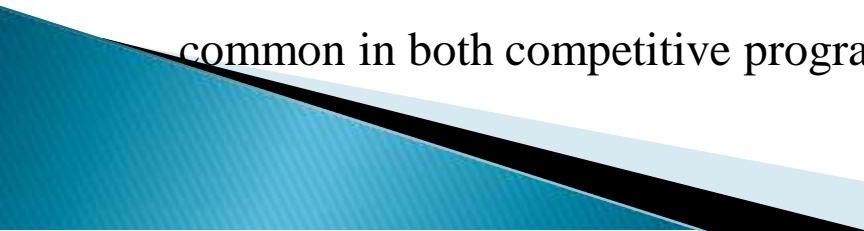
PROBLEM STATEMENT:

- You are given a string s that consists of only digits.
- Check if we can split s into two or more non-empty substrings such that the numerical values of the substrings are in descending order and the difference between numerical values of every two adjacent substrings is equal to 1.
- For example, the string $s = "0090089"$ can be split into $["0090", "089"]$ with numerical values $[90, 89]$.
- . A substring is a contiguous sequence of characters in a string.

ABSTRACT:

- ▶ This project explores the problem of splitting a string of digits into two or more non-empty substrings such that the numerical values of the substrings are in descending order, with the difference between adjacent values being exactly one.
- ▶ The objective is to develop an efficient algorithm capable of checking if such a split is possible.
- ▶ Edge cases, such as strings with leading zeros or strings that cannot form a valid sequence, are also considered in the implementation.

INTRODUCTION:

- ▶ String manipulation problems, especially those involving numerical substrings, are common in both competitive programming and real-world applications.
- 

- .The ability to split a string into valid numerical sequences with specific properties requires both a strong understanding of string manipulation and recursive problem-solving techniques.

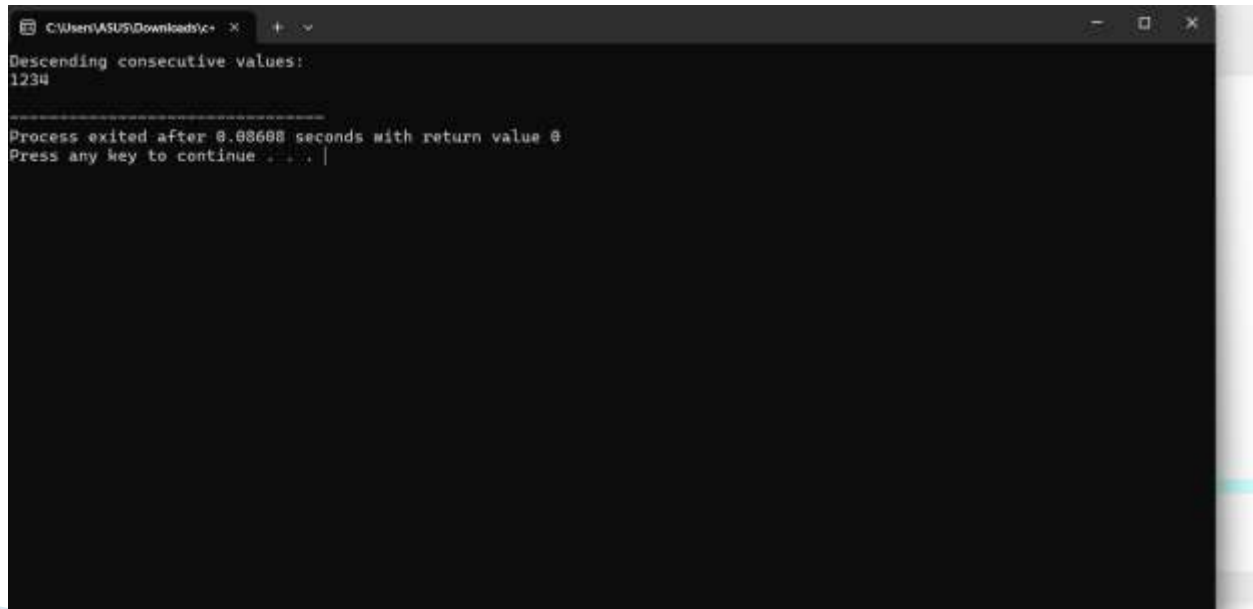
APPROACH:

Iterate through the String: Start at the beginning of the string and check each possible substring.

Validate Consecutiveness: For each substring, check if it forms a number that is part of a consecutive descending sequence.

Extract Values: Once a valid descending sequence is identified, record these values and continue with the remaining part of the string.

- ▶ **Repeat:** Continue this process until the entire string has been processed or no more valid descending sequences can be found.



```
C:\Users\ASUS\Downloads>c++ .\1234
Descending consecutive values:
1234
=====
Process exited after 0.00608 seconds with return value 0
Press any key to continue . . . |
```

COMPLEXITY ANALYSIS:

- ▶ **Time Complexity:** The time complexity of the program is determined by the number of ways the string can be split into substrings and the length of the string

BEST CASE:

- ▶ The best case occurs when the string can be split early into valid substrings. For example, a string like "4321" can be quickly identified as unsplittable since no valid splits are possible.

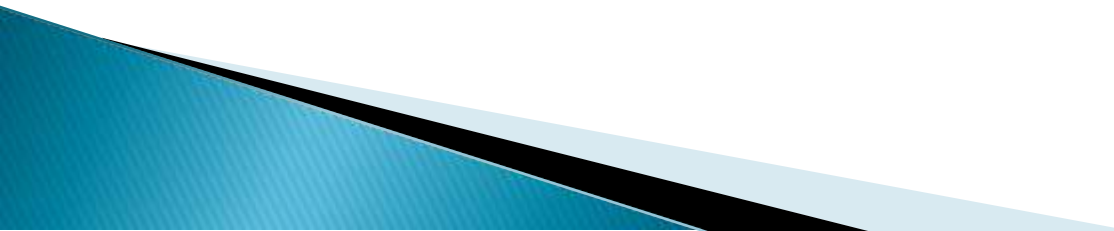
WORST CASE:

- ▶ The worst case occurs when the string requires exhaustive exploration of all possible split
- 

AVERAGE CASE:

- ▶ Depending on the distribution of digits, the recursive exploration may not need to explore all possible

FUTURE SCOPE:

- ▶ The future scope for splitting a string into descending consecutive values spans multiple areas, including optimization algorithms for performance improvements, parallel processing, and applications in natural language processing (NLP) for text segmentation and data extraction
- 

CONCLUSION:

- ▶ The project successfully addresses the problem of determining whether a string can be split into descending consecutive values. By employing a recursive backtracking approach, the algorithm explores various possible ways to partition the string, ensuring that the resulting numerical values meet the conditions of descending order and differ by exactly one.
- 