CS221 Final Project: Hybrid Packing Algorithm: Optimizing Space Utilization and Suitcase Efficiency

Gonzales, China Mae C., Javier, April Kate M., Labay, Caryll L.

BatStateU - Alangilan

May 19, 2023

Introduction

The packing algorithm aims to achieve efficient and space-optimal packing by combining the Best Fit and Next Fit strategies. Best Fit minimizes wasted space by selecting the most suitable suitcase, while Next Fit creates new suitcases when items cannot fit. This hybrid approach optimizes space utilization and minimizes the total number of suitcases, resulting in an efficient packing solution.

Problem Statement

The problem is to determine the minimum number of suitcases required to pack a given set of items, considering the size of each item and the capacity of the suitcases.

Algorithm Design

1. Initialize an empty list of suitcases.
2. Iterate through each item in the given set of items.
3. For each item, perform the following steps:

    a. Check if the item can fit into any existing suitcases using the Best Fit strategy.

    b. If a suitable suitcase is found, place the item in that suitcase.

    c. If the item cannot fit into any existing suitcases, create a new suitcase and place the item in it using the Next Fit strategy.
4. Repeat steps 2 and 3 until all items have been processed.
5. Return the list of suitcases with the packed items.

Implementation

```
FUNCTION hybridSuitcasePacking(size, n, c)
suitcaseCapacity = [c]
FOR i = 0 TO n-1
 placed = False
itemSize = size[i]
// First, try to fit the item in an existing suitcase using Best Fit strategy
FOR j = 0 TO length(suitcaseCapacity) - 1
            IF itemSize ¡= suitcaseCapacity[j] THEN
             suitcaseCapacity[j] -= itemSize
             placed = True
             BREAK
        END IF
  END FOR
  // If the item couldn't be placed in an existing suitcase, create a new suitcase (Next Fit strategy)
IF NOT placed THEN suitcaseCapacity.append(c - itemSize)
        END IF
  END FOR
  RETURN length(suitcaseCapacity)
END FUNCTION
size = [0.5, 0.7, 0.5, 0.2, 0.4, 0.2, 0.5, 0.5, 0.6]
suitcaseCapacity = 3
numItems = length(size)

minSuitcase = hybridSuitcasePacking(size, numItems, suitcaseCapacity)

PRINT "Number of suitcase required:", minSuitcase
```

Analysis

Time Complexity:
The time complexity of $O(n^2)$.

Space Complexity:
The space complexity of $O(n)$.

Conclusion

The customized packing algorithm has advantages in certain scenarios. It is simple, focused on item size, and easy to implement. By disregarding item weights, it reduces computational overhead and offers flexibility when weight differences are insignificant. While it may not always yield the optimal number of suitcases, it improves packing efficiency by distributing items evenly and optimizing space utilization. This is especially beneficial for space-conscious tasks like shipping or inventory organization. However, its effectiveness depends on specific requirements, and modifications may be necessary for weight disparities or additional factors like fragility or stacking limitations. Enhancements could include optimization techniques, adaptive resizing, and benchmarking. Overall, the customized packing algorithm offers simplicity, efficiency, and potential for further optimization and adaptability.