Analysis of Algorithms

BLG 335E

Project 2 Report

Beyza Türk
turkbe22@itu.edu.tr

Faculty of Computer and Informatics Engineering
Department of Computer Engineering
Date of submission: 19.11.2024

1. Implementation

1.1. Sort the Collection by Age Using Counting Sort

Technical Details:
I have implemented a counting sort algorithm to sort artifacts by their age. The first getMax() function provides the largest age value, which will be useful for the index vector size. The age of each artifact is used as an index to increment the corresponding value in the index vector. After calculating the cumulative frequency with the addition method, I have determined the final position of each age in the sorted vector. The initialization of the index vector takes O(max age) complexity. The calculation of the frequency takes the complexity O (n), where n is the size of the items, and each artifact causes one increment operation. Each artifact is processed only once to find the correct position in the sorted vector; therefore, the time complexity of the sorting part is O(n). Overall, complexity of Counting Sort is O(n).

1.2. Calculate Rarity Scores Using Age Windows (with a Probability Twist)

To calculate rarity scores, after sorting artifacts by age, each artifacts uniqueness is evaluated within the given ageWindow.Each item is compared with the other items in given ageWindow.
Probability of finding similar artifacts in ageWindow is calculated by the given formula.
CountSimilar:number of items with same type and origin
CountTotal:number of items in agewindow

probability=countSimilar/countTotal

Rarity Score is calculated with using probability.

RarityScore=(1-Probability)(1+(age/ageMax))

Overall sorting by age with counting sort takes O(n), and evaluating the rarity score for given ageWindow is O(cn).

1.3. Sort by Rarity Using Heap Sort

I implemented HeapSort to sort artifacts according to their rarity scores.Heapify function is used to maintain the heap property (A[parent(i) >=A[i]]) which has the time complexity O(logn). First, like build max heap , array is tarsformed to heap, then recursively root of the heap and last unsorted element swapped. After each swap, the heapify function is called to maintain heap property. It is an in-place sorting algorithm, so the space

complexity is O(1). Instead of given rarity score calculation, I have implemented a new algorithm to calculate the rarity score only with respect to origin of artifact on main file.

1.4. Analysis

Counting Sort performs better than Heap Sort due to its linear O(n+k) complexity. Heap Sort's O(nlogn) complexity is less efficient for this case. However sorting by rarity scores requires comparisons.

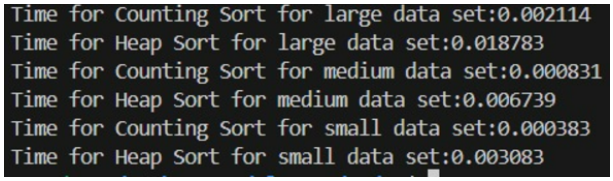

Figure 1.1: Benchmarking of sorting algorithms with different size of datasets