Algorithms and Data Structures

Sheet 8

Valdrin Smakaj

Problem 8.1 *Sorting in Linear Time* (15 points)

(a) (3 points) Implement the algorithm for Counting Sort and then use it to sort the sequence $< 9, 1, 6, 7, 6, 2, 1 >$.

(b) (3 points) Implement the algorithm for Bucket Sort and then use it to sort the sequence $< 0.9, 0.1, 0.6, 0.7, 0.6, 0.3, 0.1 >$.

(c) (3 points) Given n integers in the range 0 to k, design and write down an algorithm (only pseudocode) with pre-processing time $\Theta(n+k)$ for building auxiliary storage, which counts in O(1) how many of the integers fall into the interval $[a, b]$.

(d) (4 points) Given a sequence of n English words of length k, implement an algorithm that sorts them alphabetically in $\Theta(n)$. Let k and n be flexible, i.e., automatically determined when reading the input sequence. You can consider k to behave like a constant in comparison with n. Example sequence of words to sort:
< *"word", "category", "cat", "new", "news", "world", "bear", "at", "work", "time"* >.

(e) (2 points) Given any input sequence of length n, determine the worst-case time complexity for Bucket Sort. Give an example of a worst-case scenario and the prove corresponding the complexity.

Answer to sub-question a) and b) has been solved and attached in the folder (Cpp file)

c) Answer in the following pseudocode ➔

```
$k = \max(A) + 1$
Declare $C$: Array with $k$ elements
for $i = 0$ to $k$ do
  $C[i] = 0$
end for
for $j = 1$ to $A.length$ do
  $C[A[j]] = C[A[j]] + 1$
end for
for $j = 1$ to $A.length$ do
  $C[i] = C[i] + C[i-1]$
end for
return $C[b] - C[a-1]$
```

d) Answer for this sub-question is attached in the folder (Cpp file)

e)Bucket Sort has a time complexity of $O(n^2)$ at the worst case, when all elements are placed in a single bucket. The overall performance would then be dominated by the algorithm used to sort each bucket, which is typically insertion sort. Now we need to ask ourselves, what is the worst-case time complexity for Insertion Sort. Of course, it occurs when elements are in descending order. Therefore, the time complexity for the Insertion Sort is $O(n^2)$. It proves that Bucket Sort has a time complexity of $O(n^2)$ at the worst case.

Example ➔ arr[] = {0.193, 0.134, 0.145, 0.174, 0.128}

$$\begin{aligned} T(n) &= O(n) + O(n) + O(n^2) \\ &= 2O(n) + O(n^2) \\ &= O(n^2) \end{aligned}$$

Problem 8.2 *Radix Sort* (7 points)

Consider Hollerith's original version of the Radix Sort, i.e., a Radix Sort that starts with the most significant bit and propagates step by step to the least significant bit (instead of vice versa).

(a) (4 points) Implement Hollerith's original version of the Radix Sort.

(b) (3 points) Determine and prove the asymptotic time complexity and the asymptotic storage space required for your implementation.

Answer to sub-question a) has been solved and attached in the folder (Cpp file)

b) Hollerith's version of Radix Sort uses the bucket sort subroutine and starts from the most significant bit. If the difference between the numbers is big, it might not be necessary to spread through the digits. The reason is that all numbers will fall into different buckets, and a bucket of size 1 is an already sorted array.
So the time complexity for the best case is $\Theta(n)$, same as bucket sort.
Let's consider the worst case, when all numbers in the array the same. In this case, all the numbers will fall into the same bucket every time bucket sort is called. So the time complexity for the worst case is $\Theta(dn)$, where $d = \log_b k$ and k is the maximum element in the array and b is the base.
In the average case, half of the buckets will have either 0 or 1, and the another half will have more than 1 element in them. Therefore, the total running time will be $\Theta(n + \frac{d}{2}n) \Rightarrow \Theta(\frac{d}{2}n) \Rightarrow \Theta(dn)$

The space complexity for the best case is $\Theta(n)$. For average cases and worst cases, n buckets are declared. For every recursive call of Bucket Sort, new buckets are created. Therefore, dn buckets are created. That means that the space compelxity is $\Theta(dn)$.