CS301 Assignment 2

Ahmet Bilal Yildiz

9 November 2022

1 Question 1

PART A:

To find the k smallest number in sorted order, I choose merge sort algorithm because it performs the best which is $\Theta(n \log n)$ among the comparison-based sorting algorithms. Then just choosing the k elements which are already sorted which is $\Theta(k)$.

The algorithm for this solution is:
step1: start
step2: declare array and position variables which are left, right and mid.
step3: execute the merge function

if $left > right$:
  return
mid = $(left + rigt)/2$
$mergesort(array, left, mid)$
$mergesort(array, mid + 1, right)$
$merge(array, left, mid, right)$

step4: stop
step5: Get the first k element in the sorted list ($\Theta(k)$). As shown in the steps upper merge sort always splits data to 2 parts, so that merge sort is in the form:

$T(n) = 2T(n/2) + \Theta(n)$ (merge operation takes $\Theta(n)$ time)

By the masters theorem:

since $\Theta(n) = \Theta(n^{\log_2 2})$, running time of the merge sort part is $\Theta(n \log n)$.

And the running time of the algorithm is: $\Theta(k + n \log n)$. Since $k \leq n$, $\Theta(k + n \log n) \leq \Theta(n + n \log n)$ and so the algorithm runs in $\Theta(n \log n)$ time.

PART B:

To find the k smallest number in sorted order, I choose Blum,Floyd,Pratt,Rivest, and Tarjan(1973) algorithm as order-statistics algorithm whose time complexity is $\Theta(n)$. Then partition around the kth smallest number. Then merge sort the numbers which are smaller than the kth smallest number which is also pivot and that takes $\Theta(k \log k)$ time.
The algorithm for this solution is:

Select(i, n):

step1: Divide the n elements into groups of 5, find the median of each 3-element group.
step2: Recursively Select the median X of the $[n/5]$group medians to be the pivot.
step3: Partition around the pivot X, let $k = rank(X)$.
step4: If $i = X$then return X

Else if $i < k$:
Then recursively Select the ith smallest element in the lower part.
Else:
Recursively Select the $(i - k)th$ smallest element in the upper part.

step5: sorting k elements which are smaller than the pivot (kth smallest element) by using merge-sort.

As shown in the upper steps, Blum,Floyd,Pratt,Rivest, and Tarjan(1973) algorithm is in the form of:

$T(n) = T(n/5) + T(3n/4) + \Theta(n)$

By substitution method:

Assume $T(n) \leq cn$ Then,
$T(n) \leq \frac{1}{5}cn + \frac{3}{4}cn + \Theta(n)$

$= \frac{19}{20}cn + \Theta(n)$

$= cn - (\frac{1}{20}cn - \Theta(n))$

$\leq cn$, if c is chosen large enough to handle both the $\Theta(n)$ and initial conditions.

Therefore, this algorithm works in $\Theta(n)$ time.
Since the running time of the merge sort for k elements is $\Theta(k \log k)$,
The time-complexity of this solution is: $\Theta(n + k \log k)$.

PREFERENCE: I prefer the second algorithm because this it is more efficient than the first one. Time complexity of this algorithm is depending on k and n values and when k is near to the n, running time of this algorithm is similar to the second algorithm. However, this algorithm performs better than the first algorithm when k is small and far from n. Therefore, the second algorithm is more efficient.

2 Question 2

PART A:

I can modify the radix sort algorithm by selecting the longest string as base, and converting all other strings suitable to sort in this base by adding $\star$ characters on the right-most-ends of the

strings. While the comparison operation is running $\star$ character has priority to any other characters to ensure that the letters are the same the smallest word has priority in lexicographical order.

Steps of the algorithms are:
step1: assign the base-length = length of the longest string
step2: put $\star$ character to the right-most-end of the strings whose lengths are smaller than the base until their length will be equal to base-length.
step3: implement radix sort to the strings just same as the radix sort for integers
From right-most character to the left-most character implement counting sort for the letters. Note that the ($\star$) character has priority to any other alphabetical character while doing the comparison.

PART B:
Passes are shown below according to the character numbers. The sorting order is from up to down in each step.

7th character:	6th character:	5th character:
$BATURAY$	$GORKEM\star$	$GIRAY\star\star$
$GORKEM\star^1$	$GIRAY\star^1\star$	$TAHIR^1\star\star$
$GIRAY\star\star^2$	$TAHIR\star^2\star$	$BARIS\star\star$
$TAHIR\star\star^3$	$BARIS\star^3\star$	$BATUR^2AY$
$BARIS\star\star^4$	$BATURAY$	$GORKEM\star$

4th character:	3th character:	2th character:
$GORKEM\star$	$GIR^1AY\star\star$	$TA^1HIR\star\star$
$TAHI^1R\star\star$	$TAHIR\star\star$	$GIRAY\star\star$
$BATURAY$	$BAR^2IS\star\star$	$BA^2RIS\star\star$
$BARI^2S\star\star$	$GOR^3KEM\star$	$GORKEM\star$
$GIRAY\star\star$	$BATURAY$	BA^3TURAY

1th character:	sorted list:
$TAHIR\star\star$	$BARIS\star\star$
$B^1ARIS\star\star$	$BATURAY$
$B^2ATURAY$	$GIRAY\star\star$
$G^1IRAY\star\star$	$GORKEM\star$
$G^2ORKEM\star$	$TAHIR\star\star$

PART C:

Running time of this algorithm is basically, base-length $*$ running-time of the counting sort because we implement counting sort to the strings from right to left letter by letter. And the running time of the counting sort is $\Theta(n+k)$ where n represents number of strings and k represents the range which is from a-z and also we have $\star$ character, so the k = 27 in this algorithm. Therefore, the running time of the algorithm is:

$\Theta(L*(N+27))$

where:
L: base-length (length of the strings in the comparison)
N: number of strings that will be sorted
27: number of the range that a character can be in which is a-z and also $\star$.