CS301: Assignment #2

Muhammed Orhun Gale
morhun@sabanciuniv.edu

Sabanci University — April 8, 2022

1 Order Statistics

1.a

Since a comparison-based sorting algorithm will be employed and the lower bound for comparison-based sorting algorithms is $\Omega(n * logn)$, an asymptotically optimal comparison-based sorting algorithm can be used.

Merge-sort is a divide conquer algorithm which divides the input array exactly to half, recursively sorts divided pieces until hit the base-case and lastly merges all of the part via in $\Theta(n)$ time. Its recurrence relation is $T(n) = 2T(n/2) + n$. Merge-sort's worst case occurs when in the merge step all elements must be compared while its best-case occurs when the merge is performed on sorted arrays. By using Master Theorem, merge-sort's time complexity can be found as $\Theta(n * logn)$ which indicates that it is also $0(n * logn)$. Since the merge-sort divides array to half independent from the input its worst case matches with the average and best case asymptotically which is $0(n * logn)$.

After the array is sorted in ascending order, in order to find the k-th smallest element, returning the k-th element is enough. To do that, in worst case where it is not possible to reach the k-th element in $\Theta(1)$ time, maximum k iteration is required which would completed in $\Theta(k)$ time.

Therefore, an algorithm which applies comparison-based sorting and returning the k-th smallest element has time complexity $\Theta(nlogn) + \Theta(k)$

Therefore, time complexity = $\Theta(nlogn) + \Theta(k)$ and when the k is equal to n (which is the worst case) it will be $\Theta(nlogn) + \Theta(n) = \Theta(nlogn)$

1.b

In order to perform the RAND-SELECT in linear time, as it is shown in the class, first n elements into groups of 5. After that sort groups and get the medians and medians of medians until obtain a pivot. After that apply RAND-SELECT around the found pivot to get k-th smallest element. Then, continue to partitioning around k-th smallest element recursively. Lastly, sort the k smallest element via using merge-sort which is a comparison-based sorting algorithm with $\Theta(nlogn)$ time complexity.

Complexity of this algorithm is composed of dividing the array into groups of 5 which has $\Theta(n)$ time complexity, selecting recursively the medians which is T (n/5), partitioning around the medians which is $\Theta(n)$ and the RAND-SELECT which has at least $3\lfloor(n/10)\rfloor$ elements smaller than median and $\lfloor\lfloor(n/5)\rfloor/2\rfloor$ medians smaller than median of the medians which means $n- 3\lfloor n/10\rfloor \leq 3n/4$ at max. Then its recursion is to be $T(3n/4)$ and the whole RAND-SELECT can be defined as $T(n) = T(n/5) + T(3n/4) + \Theta(n)$

$T(n) = T(n/5) + T(3n/4) + (n)$ then by substitution method,

$T(n) \leq (1/5)c * n + (3/4) * c * n + \Theta(n)$

$= (19/20) * c * n + \Theta(n)$

$= c * n - ((1/20) * c * n - \Theta(n))$

$\leq c * n$

Then the complexity of RAND-SELECT is $\Theta(n)$ for an adequate c value which means and therefore, the total complexity is $\Theta(n)$ + $\Theta(klogk)$ for the worst case ($k = n$) it is $\Theta(nlogn)$.

I would use the second algorithm since the running time can be better than the first one for when the list is not sorted. Also, it is possible to give a better performance in best-case since the if k is smaller enough it is possible for the second algorithm to be linear.

2 Linear-time Sorting

2.a

Assuming that strings are consist of 256 ASCII characters (therefore only English alphabet is included) with any length, radix sort can be modified to sort the ASCII values of characters on the same digit of all input strings.

To do that, input strings with different lengths must be considered. In the integer case, 0's added to the left hand-side until all inputs has same length and this leads to an order that inputs which have more 0's on the left-hand side have higher priority (in ascending order). For string sorting, shorter inputs can have higher priority over only the inputs that comprise the shorter string as a substring. For example, string "abc" must have higher priority over "abcd" since the "abc" is a substring of "abcd". In this sense, to favor shorter strings, strings that are shorter than the longest string must be filled with ASCII 0 (NULL character) from the right-most character until it has same length with the longest string.

Then, as mentioned above, starting from the right-most character, ASCII values of characters on the same digit of all input strings can be sort via using a stable sorting algorithm which is counting sort in this case.

2.b

"MERTøø"	"MERTøø"	"DILARA"	"ERDEMø"	"SELINø"	"AYS Uøø"
"AYSUøø"	"AYSUøø"	"ERDEMø"	"DI LARA"	"MER Tøø"	"DILARA"
"SELINø"	"ERDEMø"	"SELINø"	"SELINø"	"DILARA"	"ERDEMø"
"ERDEMø"	"SELINø"	"MERTøø"	"MERTøø"	"ERDEMø"	"MER Tøø"
"DILARA"	"DILARA"	"AYSUøø"	"AYSUøø"	"AYS Uøø"	"SELINø"

2.c

Let number of input strings is N and maximum length of a string L.

As stable sorting algorithm that will be used in radix sort's subroutine counting sort is used whose run-time is $\Theta(N + K)$ where N is number of elements that will be sorted and K is the maximum number of possible types. Since the number of input strings is N and the maximum number of possible types is 256 since it is assumed that strings will consist of ASCII characters the complexity of the counting sort

will take $\Theta(N + 256)$.

Since each character is 8 bit, number of iterations will be $\frac{L}{8}$ at max. Therefore, run-time complexity of the radix sort will be $\Theta(\frac{L}{8}(N + 256))$. When $\frac{L}{8}$ and 256 are ignored, the complexity is $\Theta(L * N)$