ISTANBUL TECHNICAL UNIVERSITY
COMPUTER ENGINEERING DEPARTMENT

BLG 335E

ANALYSIS OF ALGORITHMS I

PROJECT I REPORT

DATE: 08.12.2020

STUDENT NAME: ABDULKADİR PAZAR
STUDENT NO: 150180028

AUTUMN 2020

Project I Report

a)

Asymptotic upper bounds of Quicksort:

Best Case: O(n*logn)

Proof: $T(N) = 2*T(N/2) + cN$

Since $a = b^d$, 1^{st} part of the Master theorem applies

$T(N) = n^1 * \log n = n * \log n$

Worst Case: $O(n^2)$

Proof:

$$
\begin{aligned}
T(N) &= T(N-1) + cN \quad N > 1 \\
T(N-1) &= T(N-2) + c(N-1) \\
T(N-2) &= T(N-3) + c(N-2) \\
&\vdots \\
T(2) &= T(1) + c(2)
\end{aligned}
$$

$T(N) = T(1) + c(2+3+4\ldots.+(N-1) + N)$

$T(N) = O(n^2)$ worst-case bound for quicksort.

Average Case: O(n*logn)

Proof: $T(N) = T(i) + T(N - i - 1) + cN$ where $i = |S_1|$

For average case, each size of S_1 is equally likely ($P = 1/N$)

Avg. value of $T(i)$ and $T(N - i - 1) = \frac{2}{N} \sum_{k=0}^{N-1} T(k)$

$$T(N) = \frac{2}{N} \sum_{k=0}^{N-1} T(k) + cN$$

$$N*T(N) = 2 \sum_{k=0}^{N-1} T(k) + cN^2 \qquad \textbf{(1)}$$

$$(N-1)*T(N-1) = 2 \sum_{k=0}^{N-2} T(k) + c(N-1)^2 \qquad \textbf{(2)}$$

Subtract (2) from (1)

$$N*T(N) - (N-1)*T(N-1) = 2T(N-1) + 2cN - c$$

Drop insignificant constant and rearrange terms:

$$N*T(N) = (N + 1)*T(N - 1) + 2cN$$

Divide by N*(N + 1) and telescope:

$$T(N)/(N + 1)= T(N - 1)/N+ 2c/(N + 1)$$
$$T(N - 1)/N= T(N)/(N - 1)+ 2c/N$$
$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$
$$T(2)/3 = T(1)/2 + 2c/3$$

Add all equations:

$$T(N)/(N + 1) = T(1)/2 + 2c\sum_{k=3}^{N+1} 1/k$$

The sum evaluates to $\ln(N + 1) + \gamma - 3/2$ where $\gamma = 0.577$

$$T(N)/(N + 1) = O(\log N)$$

$$T(N) = O(N * \log N)$$

b)

1. It will **not** give the desired output in all cases.

Example:

Sorting by profit is already done:

Country	Profit
Zambia	1000000
United Kingdom	750000
New Zealand	500000
France	250000
Albania	125000
Albania	62500
Zambia	31250

Last element is the pivot.

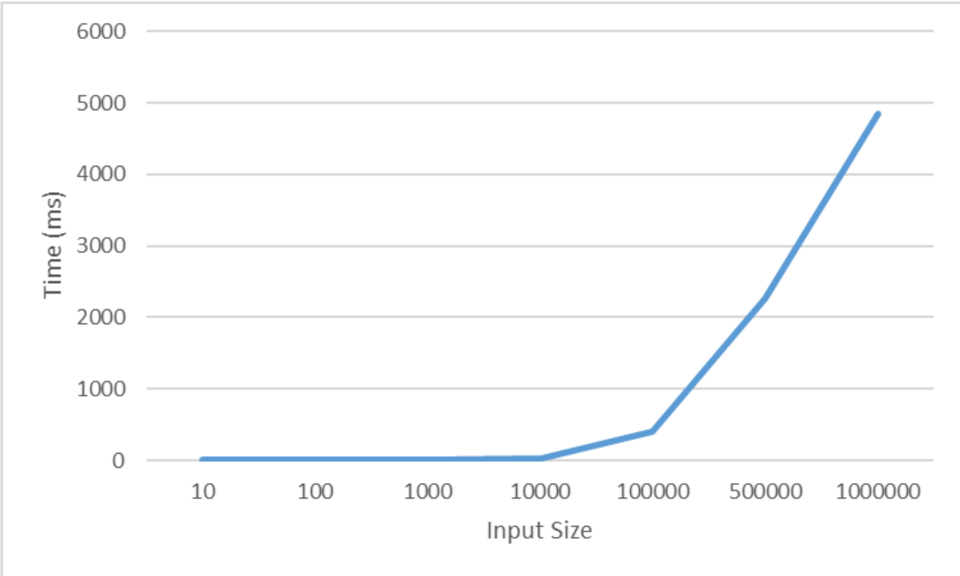
Result of sorting by country name:

Country	Profit
Albania	62500
Albania	125000
France	250000
New Zealand	500000
United Kingdom	750000
Zambia	31250
Zambia	1000000

Descending order of profits for Zambia is broken.

2. Insertion Sort, Bubble Sort, Merge Sort (Stable Sorts)

c)



Input size – time (ms) chart

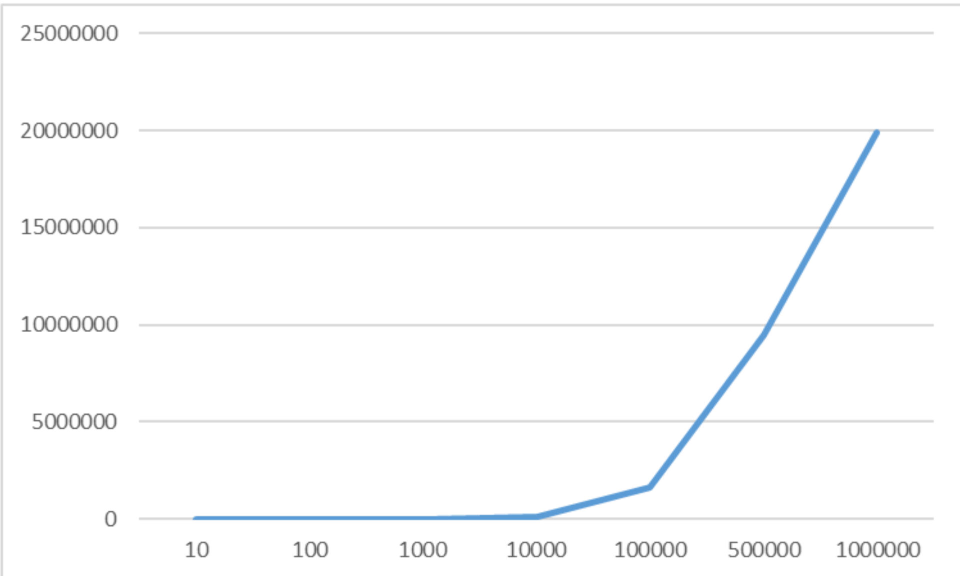
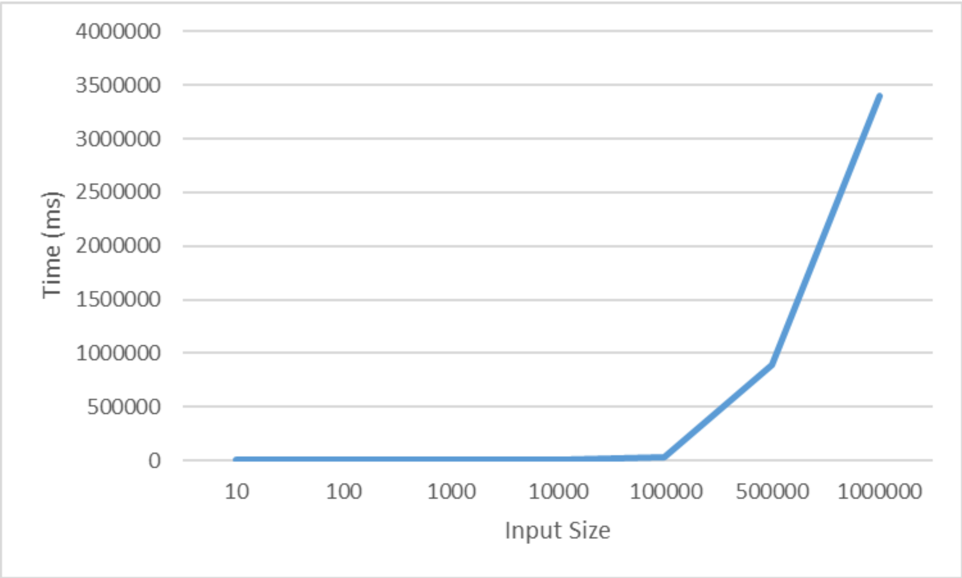


Chart of n * $\log_2 n$ for corresponding values

Quicksort's average time complexity is O(n*logn). This is the best time complexity a comparison-based sorting algorithm can have. This relationship can also be observed by comparing two charts. And as we can see from the chart, it sorts large arrays within reasonable time (~5s for 1 million sized array).

d)



Input size – time (ms) chart

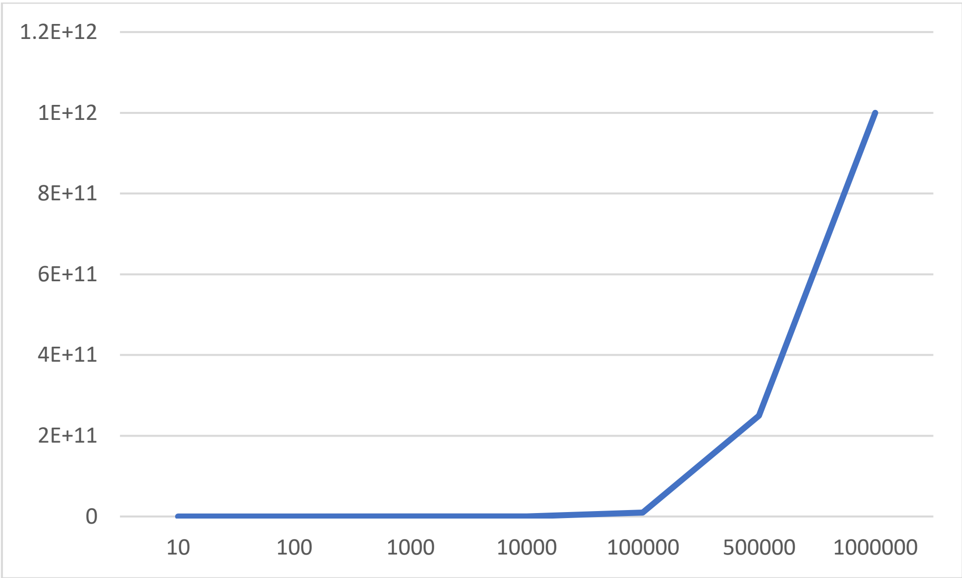


Chart of n^2 for corresponding values

1. Computation times are significantly worse than (c) since Quicksort with pivot as rightmost element performs in $O(n^2)$. This is significantly worse than O(n*logn) and the computing times clearly reflect that fact. I had to increase the stack size for input size larger than 10000 as program would experience stack overflow because it made too many recursive function calls.
2. An almost sorted array, an array sorted in reverse, an array filled with same value elements perform in worst case when pivot is the rightmost element.
3. Choosing the pivot element with a random number generator will drastically reduce the probability of worst-case scenario occurring.