1-) Asymptotic upper bound for quicksort

a-) worst case

For worst case to occur, input must be sorted or reverse sorted. The reason is one side of partition always has no elements and the other contains remaining elements.

$$T(n) = T(0) + T(n-1) + \Theta(n)$$
$$= \Theta(1) + T(n-1) + \Theta(n)$$
$$T(n) = T(n-1) + \Theta(n)$$

$n, n-1, n-2, \ldots, 2$ — arithmetic series

$\Theta(n^2)$

b-) best case

If we're lucky, partition splits the array exactly from middle.

$T(n) = 2T(n/2) + \Theta(n)$

From the master theorem we know that if $f(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(f(n)\log n)$

We obtain $\Omega(n \log n)$

2-) Write down the asymptotic upper bound for Quicksort with randomized pivot selection.

Method (Lecture Slides) - Prof. Hazim Kemal

$T(n)$ = the RV for the running time of randomized quicksort on an input of size n, assuming random numbers are independent.

For $k = 0, 1, \ldots, n-1$, define indicator RV

$$X_k = \begin{cases} 1, & \text{partition generates a } k:n-k-1 \text{ split} \\ 0, & \text{otherwise} \end{cases}$$

$E[X_k] = P(X_k = 1) = 2/n$ since all splits share the same probability.

$$T(n) = \begin{cases} T(0) + T(n-1) + \Theta(n), & \text{if } 0:n-1 \text{ split} \\ T(1) + T(n-2) + \Theta(n), & \text{if } 1:n-2 \text{ split} \\ \vdots & \vdots \\ T(n-1) + T(0) + \Theta(n), & \text{if } n-1:0 \text{ split} \end{cases}$$

$$\sum_{k=0}^{n-1} X_k (T(k) + T(n-k-1) + \Theta(n))$$

Take expectation

$$E[T(n)] = E\left[\sum_{k=0}^{n-1} X_k (T(k) + T(n-k-1) + \Theta(n))\right]$$
$$= \sum_{k=0}^{n-1} E[X_k (T(k) + T(n-k-1) + \Theta(n))]$$

Independence of X_k

$$= \sum_{k=0}^{n-1} E[X_k] \, E[T(k) + T(n-k-1) + \Theta(n)]$$

Linearity of expectation and $E[X_k] = 1/n$

Method 2: (I found more intuitive)

Let's assume no two elements are equal since it is the worst case and make our notation simpler. We will be writing the quantity we care (total number of comparisons) as a sum of simpler random variables.

Define

$$X_{ij} = \begin{cases} 1, & \text{if algo compares } i^{th} \text{ and } j^{th} \text{ element in the course of sorting} \\ 0, & \text{otherwise} \end{cases}$$

Let X denote the total number of comparison

$$X = \sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{ij} \quad \text{therefore} \Rightarrow E[X] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} E[X_{ij}]$$

Let's consider one of these X_{ij} for $i<j$. Denote i^{th} smallest element in the array by e_i and j^{th} smallest e_j, and imagine lining up elements in sorted order. If the pivot we choose is between e_i and e_j, then these two end up in different buckets and we will never compare them to each other. If the pivot is either e_i or e_j then we do compare them. If the pivot is less than e_i and greater than e_j, then both e_i and e_j end up in the same bucket and we have to pick another pivot. So, if we choose e_i or e_j

$$\frac{1}{n}\sum_{k=0}^{n-1} E[T(k)] + \frac{1}{n}\sum_{k=0}^{n-1} E[T(n-k-1)] + \frac{1}{n}\sum_{k=0}^{n-1} \Theta(n)$$

identical terms

$$\frac{2}{n}\sum_{k=1}^{n-1} E[T(k)] + \Theta(n)$$

then X_{ij} becomes 1, if we choose an element between e_i or e_j then X_{ij} becomes 0, otherwise we need to select another pivot. At each step, probability that $X_{ij}=1$ conditioned on the event that the game ends in that step is exactly $2/(j-i+1)$. Therefore, overall probability that $X_{ij}=1$ is $2/(j-i+1)$

$$E[T(n)] = \frac{2}{n}\sum_{k=2}^{n-1} E[T(k)] + \Theta(n)$$

$k=0,1$ terms can be absorbed in the $\Theta(n)$.

Prove $E[T(n)] \le an\lg n$ for constant $a>0$ → choose large so that $an\lg n$ dominates for $E[T(n)]$ for small inputs

Fact: $\sum_{k=2}^{n-1} k\lg k \le \frac{1}{2}n^2\lg n - \frac{1}{8}n^2$

$$E[T(n)] \le \frac{2}{n}\sum_{k=2}^{n-1} ak\lg k + \Theta(n)$$

Substitute inductive hypothesis

$$E[T(n)] \le \frac{2}{n}\sum_{k=2}^{n-1} ak\lg k + \Theta(n)$$
$$\le \frac{2a}{n}\left(\frac{1}{2}n^2\lg n - \frac{1}{8}n^2\right) + \Theta(n)$$
$$= an\lg n - \left(\frac{an}{4} - \Theta(n)\right)$$

desired, residual

$\le an\lg n$ if a is chosen large we can satisfy equation

So. $O(n\lg n)$

In other words, for a given element i, it is compared to $i+1$ with probability $\frac{2}{3}$, $i+2$ with $\frac{2}{4}$, $i+3$ with $\frac{2}{5}$ and so on.

So we have,

$$E[X] = \sum_{i=1}^{n} 2\left(\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-i+2}\right)$$

The quantity $1+\frac{1}{2}+\frac{1}{3}\cdots+\frac{1}{n}$ is denoted as H_n "n^{th} harmonic number" and it is in the range of $[\ln n,\ 1+\ln n]$. So we have

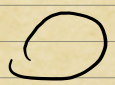
$$E[X] \le 2n(H_n - 1) \le 2n\log n \quad O(n\log n)$$

Question 3 and 4 are answered in the algo. xlsx file answers page and plots are displayed in the second sheet.

Q5) Dual Pivot Algorithm:

I consider deterministic dual pivot quicksort. The implementation divides the array into 3 subparts. For the best case, assume we partition the array into 3 arrays which have the same size. So recurrence relation would be

$$T(n) = 3T(n/3) + \Theta(n) \rightarrow \text{combine step.}$$



$n^{\log_3 3} = n$

From Master Theorem we know that if $f(n)$ and $n^{\log_b a}$ is equal, which in this case it is true we can write complexity as $\Omega(n \lg n)$

$\Omega(n \lg n)$

For the worst case let's assume we select minimum 2 elements & max 2 elements. Then recurrence relation would be

$T(n) = T(n-2) + \Theta(n)$
$T(n-2) = T(n-4) + \Theta(n)$
⋮
$O(n^2)$

goes like that

As a result of this outcome, we can say that the worst-case is $\frac{n}{2} \cdot n$, $O(n^2)$ for this method.

Dual pivot quicksort is faster than traditional quicksort, although in theory (classical analysis) it should be slower. CNIST Linux website). And it is used in Java. Vladimir Yaroslavskiy proved it in his paper.

dual pivot quicksort:
$2n \lg n$ → avg num of comparisons
$0.8 n \lg n$ → avg num of swaps

normal quicksort:
$2n \lg n$ → avg num of comparisons
$n \lg n$ → avg num of swaps