```
/* low  --> Starting index,  high  --> Ending index */
quickSort(arr[], low, high)
{
    if (low < high)
    {
        /* pi is partitioning index, arr[pi] is now
           at right place */
        pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);  // Before pi
        quickSort(arr, pi + 1, high); // After pi
    }
}
```

n-1

a

```
partition (arr[], low, high)
{
    // pivot (Element to be placed at right position)
    pivot = arr[high];

    i = (low - 1)  // Index of smaller element and indicates the
                   // right position of pivot found so far

    for (j = low; j <= high- 1; j++)
    {
        // If current element is smaller than the pivot
        if (arr[j] < pivot)
        {
            i++;    // increment index of smaller element
            swap arr[i] and arr[j]
        }
    }
    swap arr[i + 1] and arr[high])
    return (i + 1)
}
```

El algoritmo lo analizare tomando en cuenta solo el peor de los casos.

Para el peor de los casos tendriamos un conjunto de datos como sigue:

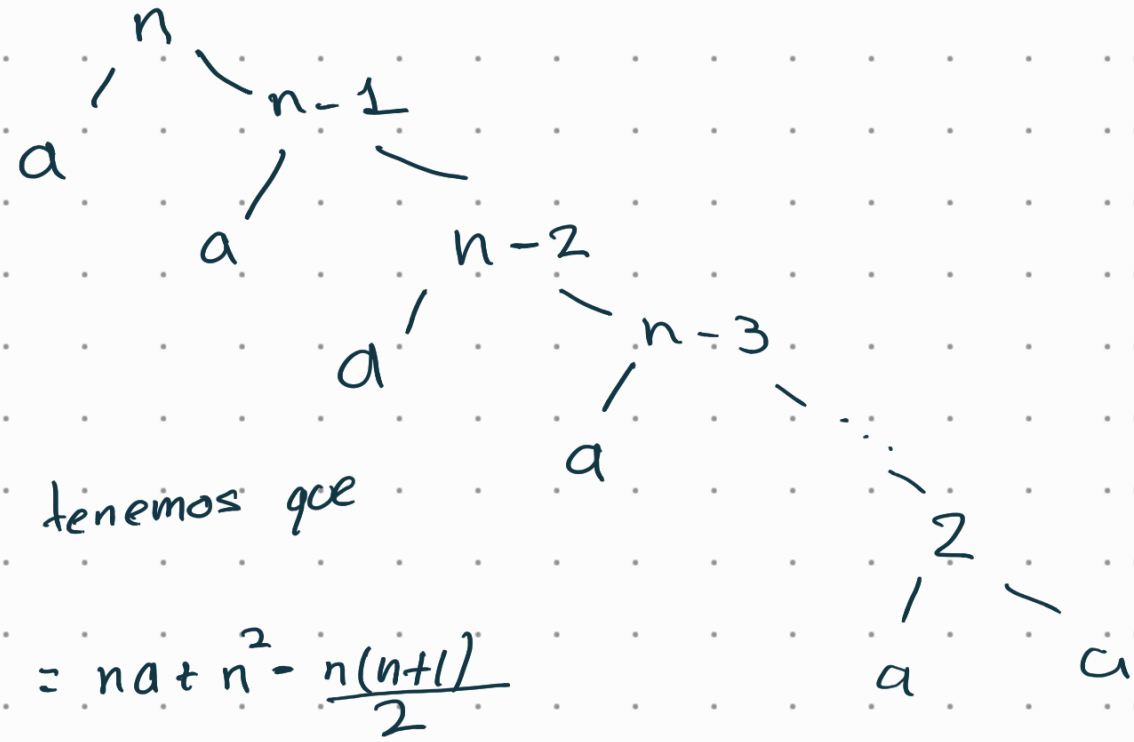
$[1, 2, 3, 4, \ldots, n-1, n-2]$

donde $n < n-1 < n-2 < \ldots < 3 < 2 < 1$

Cuando el algoritmo entra por primera vez a partition el valor final de i es igual al tamaño del arreglo, por lo que cuando entra al segcno quick sort() inmediatamente termina la función en un tiempo a.

Par el primer quicksort tendremos (n-1) iteraciones en el nuevo partition este de nuevo retornara un valor $i >$ tamaño del arreglo, y asi sucesivamente

Por lo que podemos visualizar que



Sumando tenemos que

$$na + \sum_{i=0}^{n}(n-i) = na + n^2 - \frac{n(n+1)}{2}$$

$$\frac{2na + 2n^2 - n^2 - n}{2} = \frac{n^2 + an}{2} \Rightarrow \boxed{O(n^2)}$$