Tue 02/04/16

Merge sort continued

```
So now T(n) = 4(2T(n/8) + n + 1) + 8n + 3
 => 8T(n/8) + 4n + 4 + 8n + 3
 => 8T(n/8) + 12n + 7
 => 2^i T(n/2^i) + 4in + 2^i - 1

Solve for i such that n/2^i = 1

n = 2^i

log(2)(n) = i

So
T(n) = 2^(log(2)(n)) T(n/2^(log(2)(n))) + 4nlog(2)(n) +
2^(log(2)(n)) - 1
 => n.T(n/n) + 4nlog(2)(n) + n - 1
 => n + 4n(log(2)(n)) + n - 1
 => 4nlog(2)(n) + 2n - 1
```

Time taken :

|n|Selection Sort (n^2) | MergeSort ($4n\log_2 n$) | |-|-| | 16 | 256 | 256 | | 1024 | ~million | 40000 | | 1m | 1trillion | 80 million |

Quick Sort

```
partition(A, x)
 i=0, j=A.length-1
 while (i<j):
```


© Copyright Thursday, Feb 4, 2016, 3:46 PM by COMPANYNAME

```
   if(A[i] <= x):
    i++
   else if(A[j] >= x):
    j--
   else:
    SWAP(A[i], A[j])
  return (A[i] > x) ? i : i+1

 QuickSort(A)
  if A.length <=1 then return
  t = partition(A[0 : A.length-2], A[A.length-1])
  QuickSort(A[0 ... t-1])
  QuickSort(A[t+1 .. A.length-1])
```

Running time

T(1) = 1 T(n) = 3(n+1) + T(t) + T(n-t-1)

For sorted array : 3n + T(n-1) => n^2

For unsorted array [1,3,7,8,5]: 2T(n/2) + 3n => $3n\log_3 n$

Cool fact : If quicksort picks the pivot for A, then quicksort runs in ~ nlogn time with high probability

© Copyright Thursday, Feb 4, 2016, 3:46 PM by COMPANYNAME