# BLG 335E
# Homework 2 Report
# 150130032 – Baran Kaya

**a)** Worst case:

Cost of Partition: Θ(n)
Recurrence for Quicksort: T(n) = T(n-1) + T(0) + Θ(n) = T(n-1) + Θ(n)
Total : T(n) = Θ(n) T(n-1)
T(n) = Θ(n) + Θ(n-1) + Θ(n-2) + ... + Θ(1)
T(n) = Θ(n^2) -> Upperbound = O(n^2)

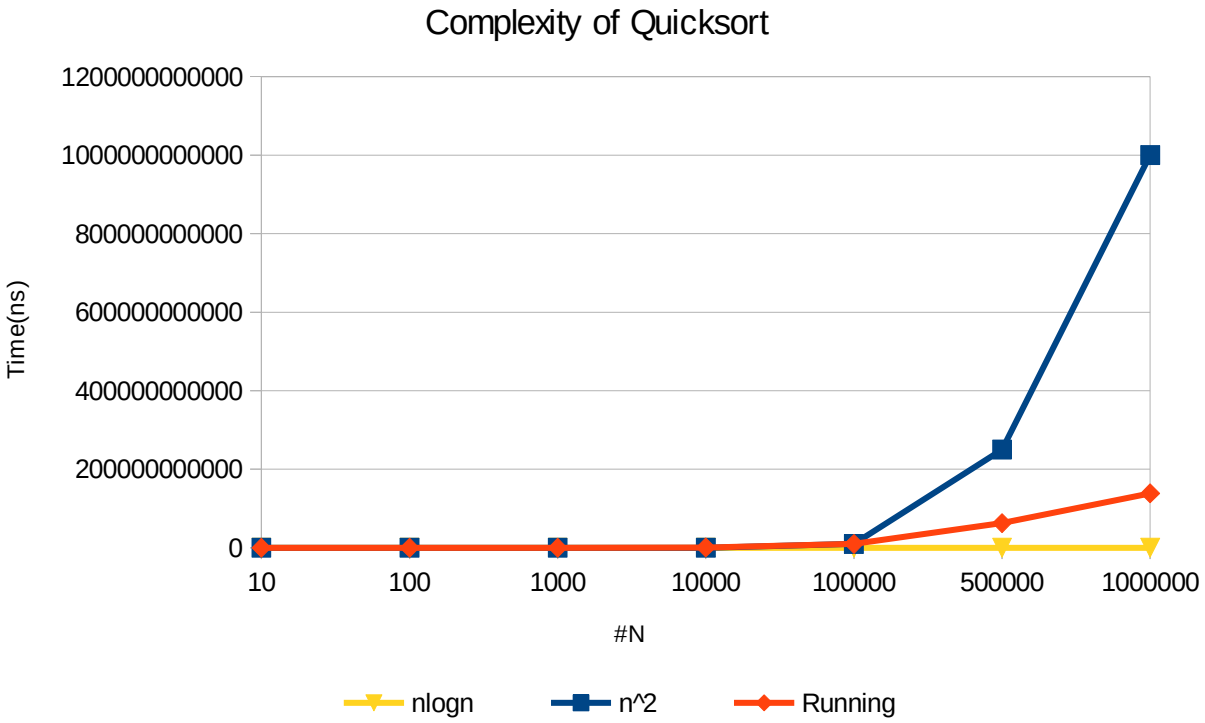
Best/Average case:

Recurrence for Quicksort: T(n) ≤ 2T(n/2) + Θ(n)
Solving Recurrence: T(n) = O(n log n) <- Upperbound

T(n) ≤ 2T(n/2) + Θ(n)
Master Theorem case 2:
a = 2, b = 2, f(n) = n
nn^logb(a) = n^log2 = n
f(n) = n -> f(n) = n^logb(a)
T(n) = Θ( (n^logb(a)) * logk+1(n) )
T(n) = Θ( (n^log2(2)) * logk+1(n) )
T(n) = Θ(n log n)
T(n) = O(n log n) -> Upperbound

**b)**

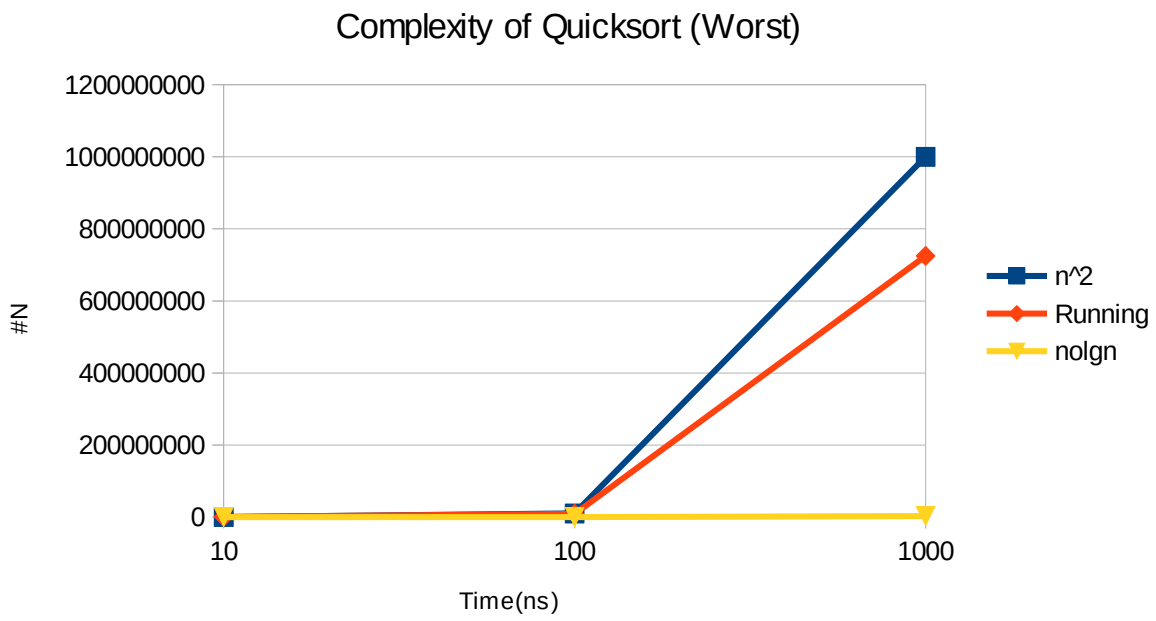
| ns | 10 | 100 | 1000 | 10000 | 100000 | 500000 | 1000000 |
|---|---|---|---|---|---|---|---|
| 1 | 156419 | 2214900 | 44304567 | 713745435 | 9998538864 | 62264656931 | 138872147521 |
| 2 | 152313 | 2218595 | 44264333 | 710636775 | 9847431373 | 62510915618 | 142297856617 |
| 3 | 156007 | 2207100 | 42574111 | 705008999 | 9830114510 | 62473075110 | 138143475837 |
| 4 | 153545 | 2261292 | 45814969 | 709628882 | 9762085677 | 62316047535 | 137827167605 |
| 5 | 153545 | 2240354 | 43598837 | 701725446 | 9879535310 | 62507054425 | 138534734310 |
| 6 | 169145 | 2333959 | 41681583 | 717297896 | 9865521706 | 65048414961 | 137646938389 |
| 7 | 149850 | 2228448 | 43102075 | 711981315 | 9831696346 | 62675197144 | 137891664504 |
| 8 | 153544 | 2211616 | 42704256 | 707775263 | 9853215977 | 62735865284 | 137756525641 |
| 9 | 150670 | 2331905 | 42366785 | 729653301 | 9894270245 | 63438940198 | 138326167513 |
| 10 | 159292 | 2240354 | 42391419 | 712414442 | 9881496903 | 62672317980 | 138569564689 |
| Average | 155433 ns | 2248852.3 ns | 43280293.5 ns | 711986775.4 ns | 9864390691.1 ns | 62864248518.6 ns | 138586624263 ns |
| Ave. Sec. | 0.000155433 s | 0.0022488523 s | 0.043280293 5 s | 0.7119867754 s | 9.8643906911 s | 62.8642485186 s | 138.586624263 s |




**c)** If pivot is the biggest or the smallest value in the array than this is the worst case for quick sort. For that purpose i choose the pivot as the one of the (population=1) residence so that it can be smaller than nearly the whole array.

| ns | 10 | 100 | 1000 | 10000 | 100000 | 500000 | 1000000 |
|---|---|---|---|---|---|---|---|
| 1 | 244686 | 8723709 | 686306951 | | | | |
| 2 | 243455 | 8300025 | 711402855 | | | | |
| 3 | 243454 | 8570574 | 730010067 | | | | |
| 4 | 242223 | 8272928 | 722209267 | | | | |
| 5 | 238117 | 9630606 | 749444530 | | | | |
| 6 | 238528 | 9675356 | 719475026 | | | | |
| 7 | 254949 | 9235660 | 706984961 | | | | |
| 8 | 244275 | 9419175 | 759552192 | | | | |
| 9 | 305447 | 9419175 | 735287235 | | | | |
| 10 | 250433 | 8214630 | 727295121 | | | | |
| Average | 250556.7 ns | 8946183.8 ns | 724796820.5 ns | | | | |
| Ave. Sec. | 0.0002505567 s | 0.0089461838 s | 0.07247968205 s | | | | |

After N=1000 i tried N=10000 but the program failed and i couldn't figure out why it failed. So i decided to make this graph with N=10, N=100 and N=1000




**d)** Quicksort is not stable also my quicksort in code is not stable neither. Because quicksort changes the elements position that its value equals to pivot ( if(A[i]<=pivot) ).

ex.
if (residenceVec[j].get_population() < pivot.get_population() || (residenceVec[j].get_population() == pivot.get_population() && residenceVec[j].get_geo_id() <= pivot.get_geo_id()))

-In this code if these two residences populations are the same it will change their position.
-If their populations and geo_ids are the same then it will swap them too.

1 - 5,85,0,male,64120,8600000US64120,
2 - 5,55,60,male,52127,8600000US52127 ,
3 - 5,34,40,female,52127,8600000US52127 ,

-When comparing 1 and 2, first it compares their population and if they are equal it compares their geo_id. After comparing it will swap 1 and 2's position.

-When comparing 2 and 3, first it compares their population and they are equal so it compares their geo_id. Geo_id's are equal too so it will swap 2 and 3's position.

-Therefore quicksort is not a stable algorithm because it swaps equal elements.

**Compiling on SSH**: g++ -std=c++11 ZipcodeQ.cpp -o z --> It need C++11 for 'stoi' function.

*150130032*
*Baran KAYA*