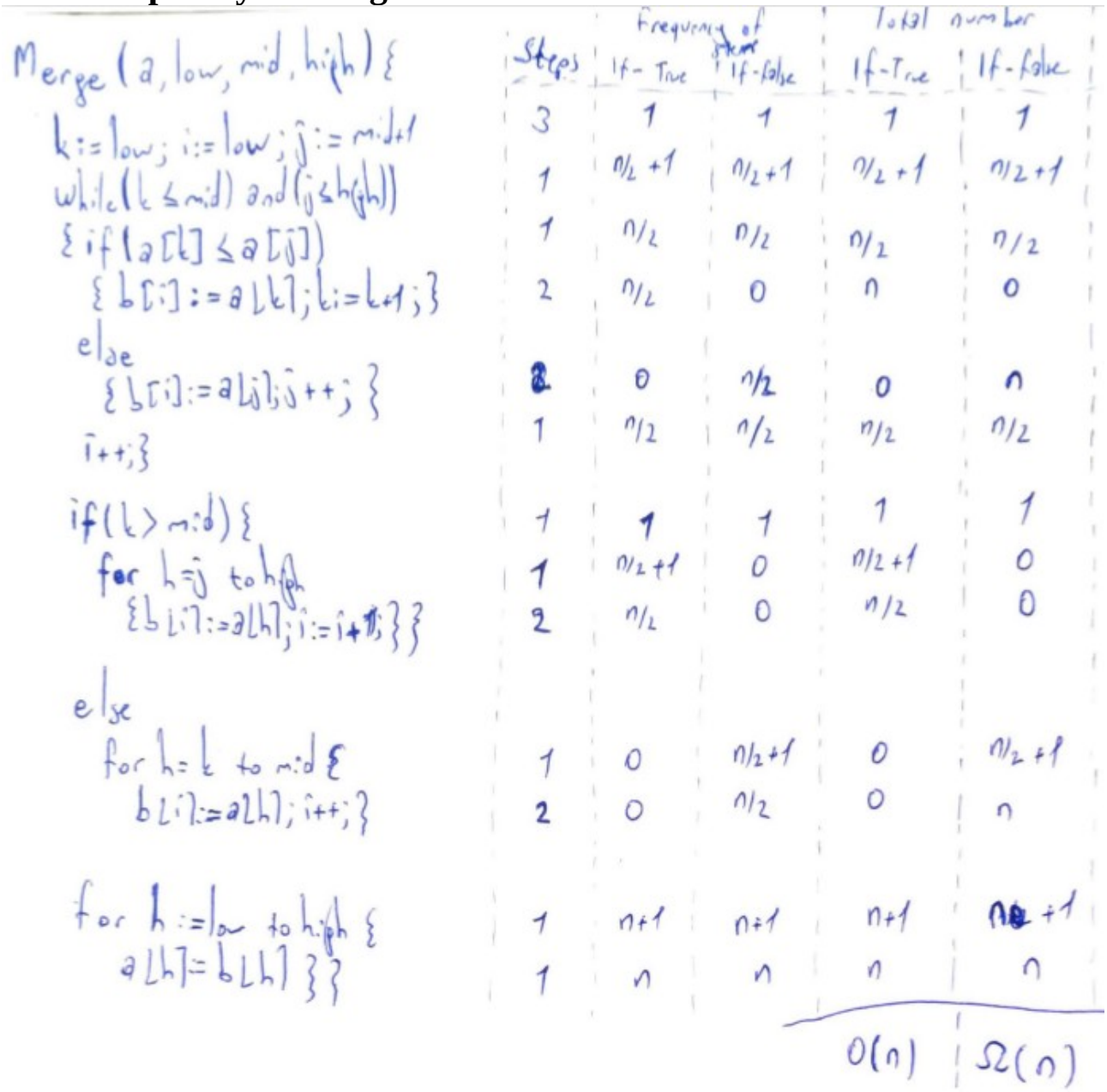
a)

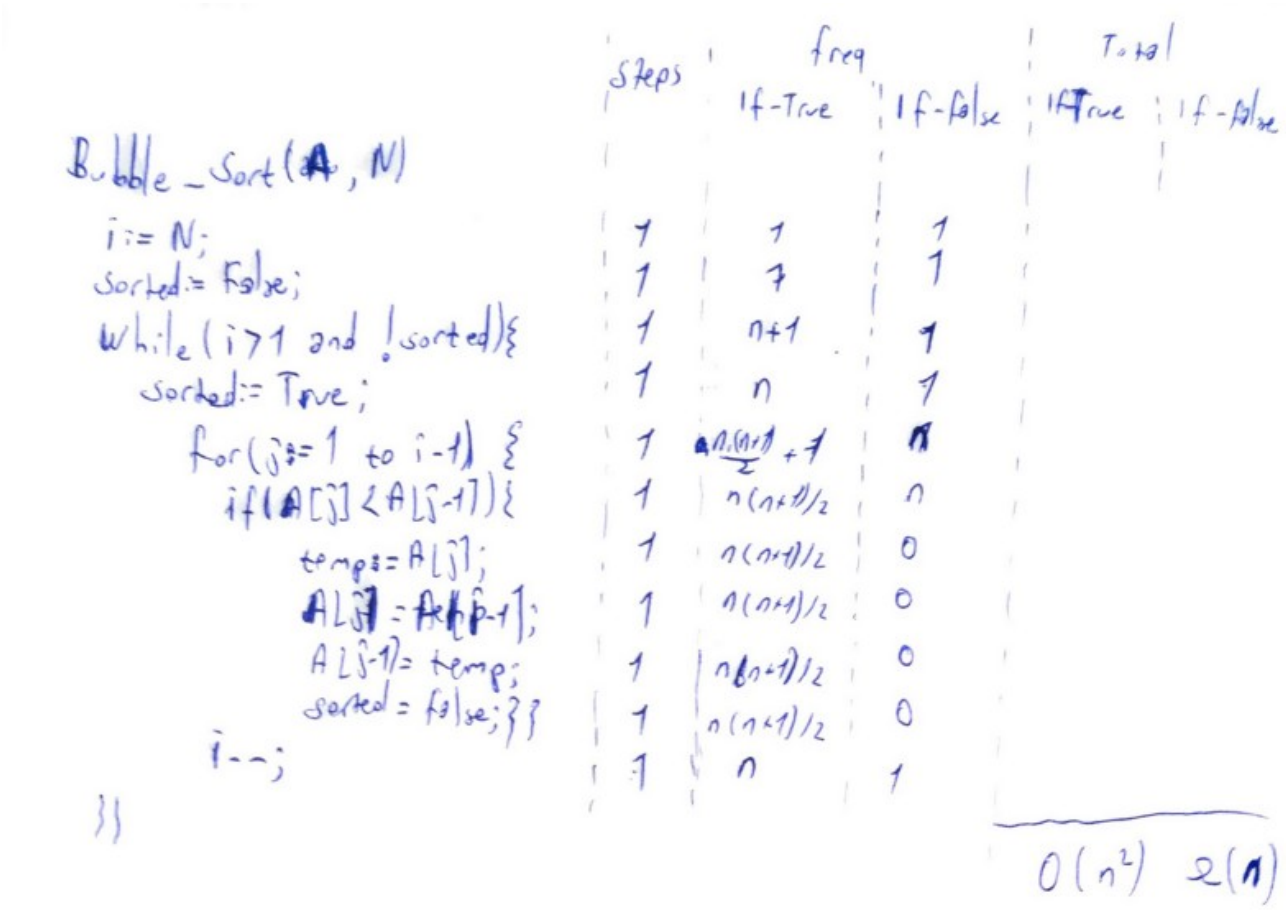
complexity of Merge



Merge (a, low, mid, high) {	Steps	Frequency of If-True	Frequency of If-false	Total number If-True	Total number If-false
k := low; i := low; j := mid+1	3	1	1	1	1
while (k ≤ mid) and (j ≤ high)	1	n/2 +1	n/2+1	n/2 +1	n/2+1
{ if (a[k] ≤ a[j])	1	n/2	n/2	n/2	n/2
{ b[i] := a[k]; k := k+1; }	2	n/2	0	n	0
else					
{ b[i] := a[j]; j++; }	1	0	n/2	0	n
i++; }	1	n/2	n/2	n/2	n/2
if (k > mid) {	1	1	1	1	1
for h = j to high	1	n/2 +1	0	n/2 +1	0
{ b[i] := a[h]; i := i+1; } }	2	n/2	0	n/2	0
else					
for h = k to mid {	1	0	n/2+1	0	n/2 +1
b[i] := a[h]; i++; }	2	0	n/2	0	n
for h := low to high {	1	n+1	n+1	n+1	n +1
a[h] := b[h] } }	1	n	n	n	n
				O(n)	Ω(n)

Worst and best case complexity: *O(nlogn)*

complexity of Bubble sort algorithm



Worst case complexity: O(n^2)
Best case complexity: O(n) (I used O because I don't know how to write theta)

B)

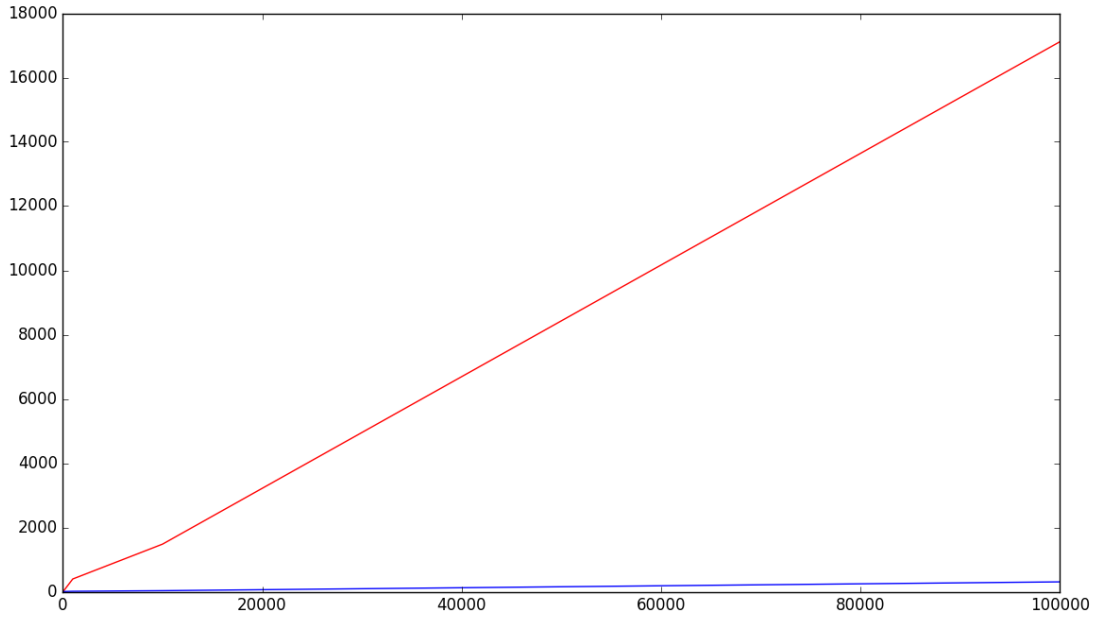
unsorted.txt

```
bzkrt@computer:~/analisysOfAlgorithms$ g++ hw1.cpp -o sort
bzkrt@computer:~/analisysOfAlgorithms$ ./sort m 100 unsorted.txt
sorting N=100 array with m take 51 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort m 1000 unsorted.txt
sorting N=1000 array with m take 633 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort m 10000 unsorted.txt
sorting N=10000 array with m take 4397 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort m 100000 unsorted.txt
sorting N=100000 array with m take 28044 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort m 1000000 unsorted.txt
sorting N=1000000 array with m take 253833 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort b 100 unsorted.txt
sorting N=100 array with b take 128 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort b 1000 unsorted.txt
sorting N=1000 array with b take 3863 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort b 10000 unsorted.txt
sorting N=10000 array with b take 311927 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort b 100000 unsorted.txt
sorting N=100000 array with b take 36074690 clock
```

sorted.txt

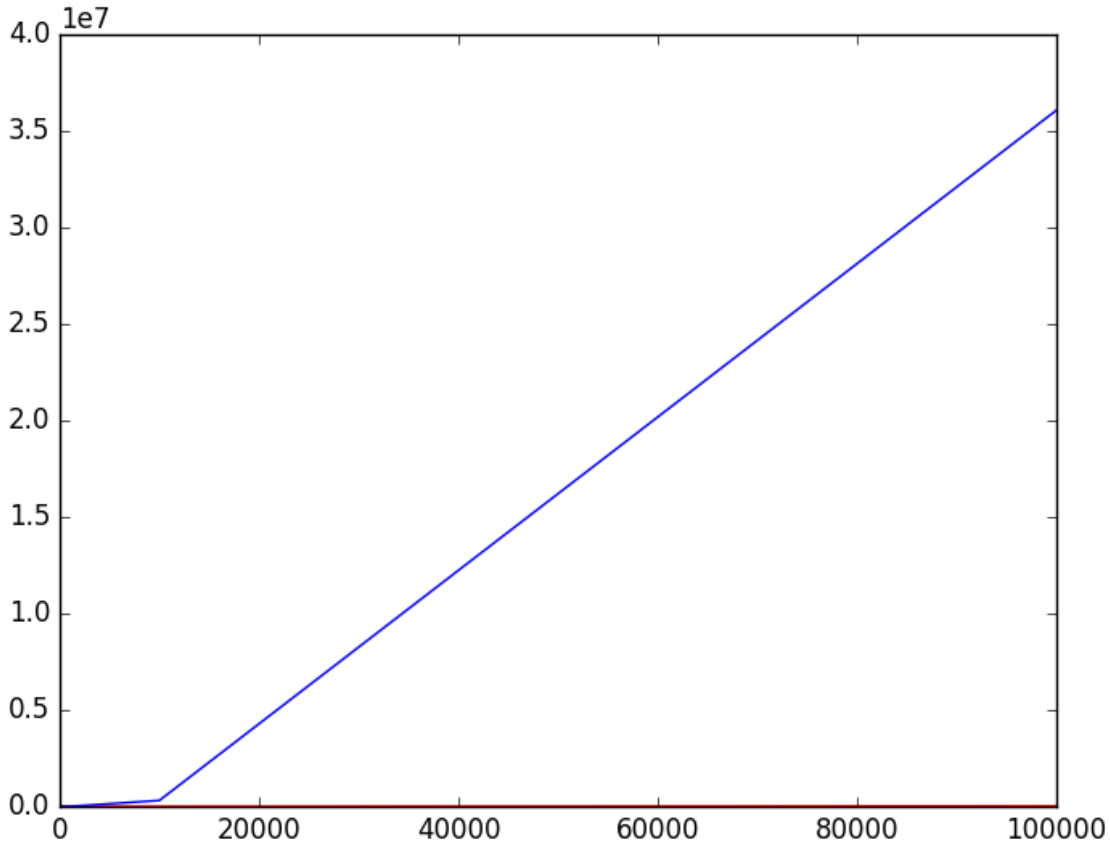
```
bzkrt@computer:~/analisysOfAlgorithms$ ./sort m 100 sorted.txt
sorting N=100 array with m take 32 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort m 1000 sorted.txt
sorting N=1000 array with m take 396 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort m 10000 sorted.txt
sorting N=10000 array with m take 1480 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort m 100000 sorted.txt
sorting N=100000 array with m take 17114 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort b 100 sorted.txt
sorting N=100 array with b take 6 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort b 1000 sorted.txt
sorting N=1000 array with b take 13 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort b 10000 sorted.txt
sorting N=10000 array with b take 32 clock

bzkrt@computer:~/analisysOfAlgorithms$ ./sort b 100000 sorted.txt
sorting N=100000 array with b take 306 clock
```

C) sorted figure:



red represents merge sort
in best scenario, bubble sort clearly much faster than merge sort as we can deduce from best case complexity.

Unsorted figure :



Red line represents merge sort
in this case merge sort clearly faster than bubble sort as worst case claims.

D)