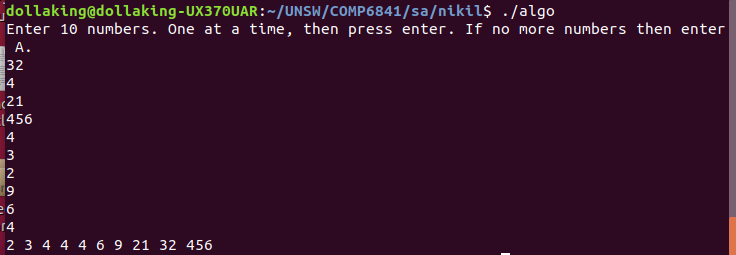
Sorting Algorithm Write Up

This binary contains a sorting algorithm and I need to find what sorting algorithm it is using reverse engineering.

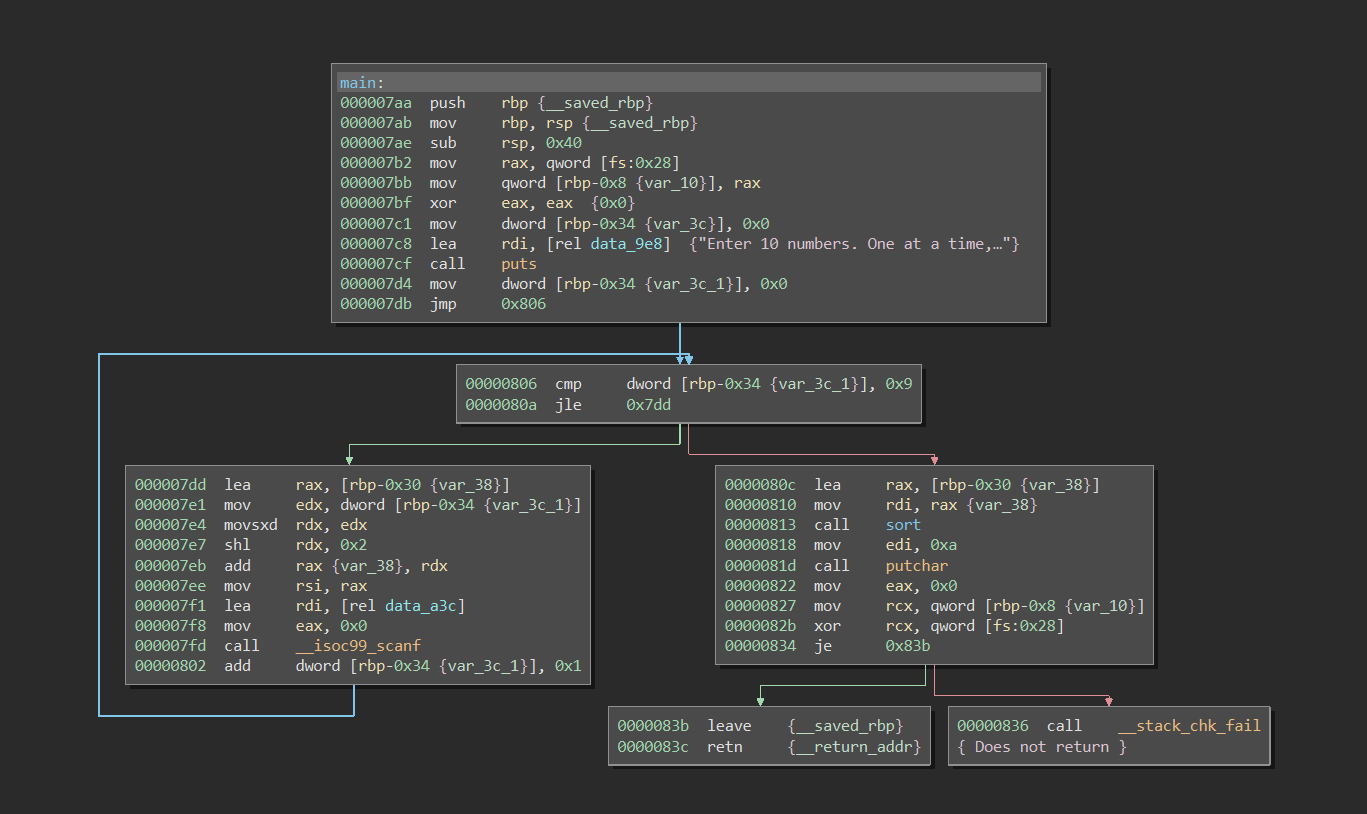
When I run the program this is what I see:



As we can see here the sorting algorithm only takes in 10 numbers and the numbers are taking one at a time, then after the 10 numbers are inputted the sorted numbers are printed out into the standard output.

So for this I used a tool called Binary Ninja which graphs out the jumps so it will be easier to look at.

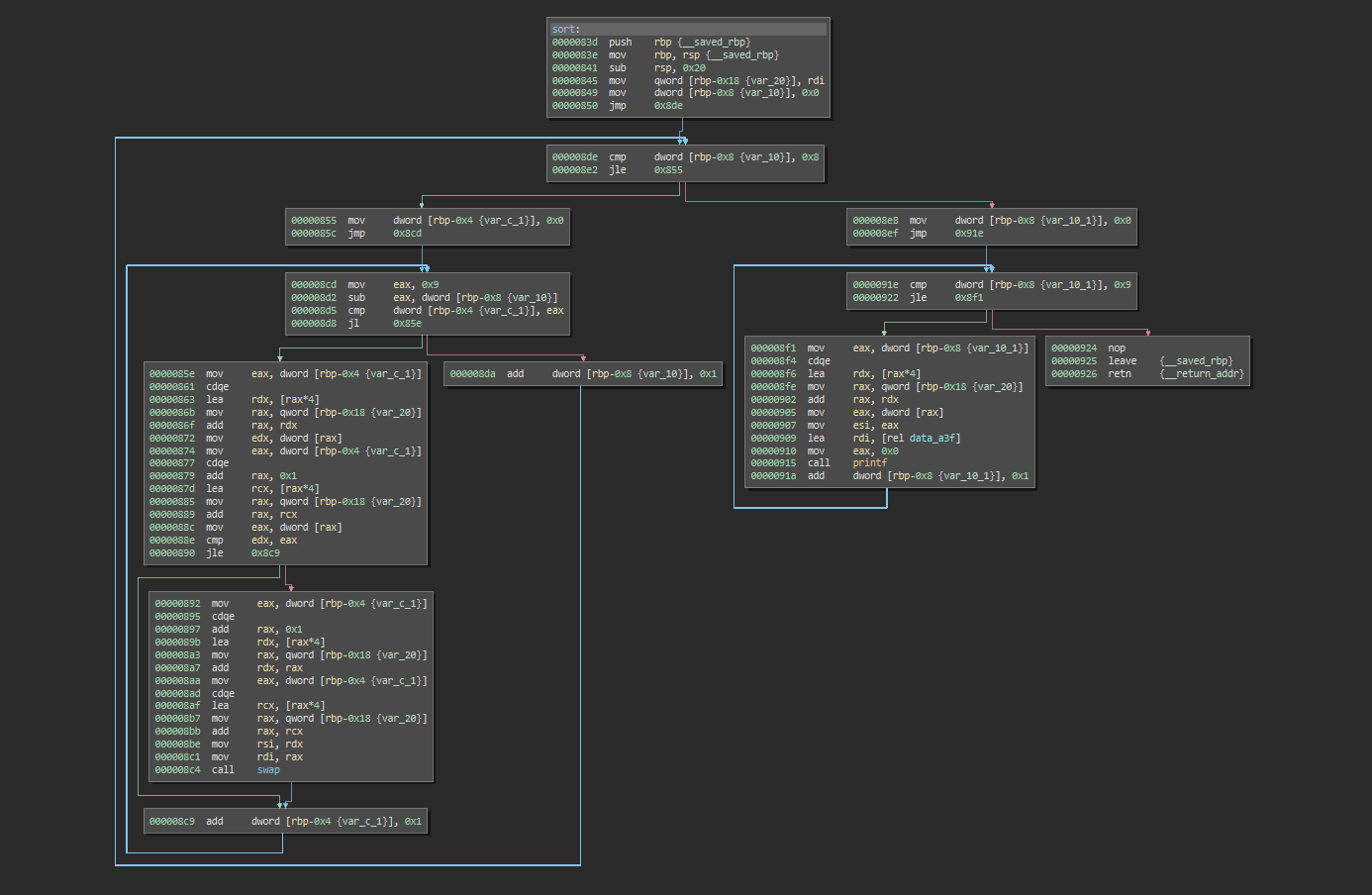
When I load up the binary on Binary Ninja this is what I see:



In the main function we see that “Enter 10 numbers. One at a time”} is being printed out at 0x7c8. Then it jumps to 0x806. In 0x7d4 we see that 0x0 which is 0 in decimal is moved into dword [rbp-0x34 {var\_3c\_1}]. In 0x806 it is comparing it to 0x9 which is 9. If it is less that 9 then it will jump to the address 0x7dd. In 0x802 we see that we are adding 1 to the pointer then it goes back to comparing whether it is less than 9. Here it is safe to say that this is a loop to get 10 numbers to sort.

When the number is greater than 9 then the jle command is ignored.

The next thing that caught my attention the call of the function “sort” at 0x813.



We see that an array of characters were passed into the $rdi register back in 0x810 before the sort function is being called. In the beginning of the sort function the $rdi register is being pushed into qword[rbp-0x18 {var\_20}]. We also see that dword [rbp-0x8 {var\_10}] is now 0 as seen in 0x849.

In 0x8de we see that dword [rbp-0x8 {var\_10}] that was assigned zero is compared and check if it is less than 8. I also noticed that at the end of their process after the jle jump they add 1 to dword [rbp-0x8 {var\_10}] which means that it is running a loop to check if the it is at the second last number of the array.

We also see that there is another loop starting from 0x8cd. This indicates that this is a nested loop thus the time complexity of this algorithm is O(n^2).

I notice that dword [rbp-0x4 {var\_c\_1}] is reset to 0 after it is done with its loop seen in 0x855.

In the section starting from 0x8cd:

$eax = 9

$eax = $eax - dword [rbp-0x8 {var\_10}]

$eax = $eax – 0

If (dword [rbp-0x4 {var\_c\_1}] == $eax) {

Jump to 0x85e

} else {

dword [rbp-0x8 {var\_10}]++

}

At this point I am very confident this is bubble sort because in bubble sort the end of the array will be sorted after every iteration of the loop. Since this is O(n^2) and a swap function is called at 0x8c4, this algorithm is bubble sort.