Group member:

Yihe Guan **V00817207**

Rui Ma **V00800795**

Nannan Zhang **V00809956**

CSC 106 Assignment 2: Sorting through the bubbles, and in through the Or Gate

TASK 1: Sorting

How does the sorting algorithm Bubble Sort work?

According to [1], Bubble Sort can be explained listed below:

i. Compare the first pair of adjacent elements. If the value of the first element is bigger than the value of the second element, then change the position of two elements; and if the value of the first element is smaller than the value of the second element, then do not change.

ii. Use step 1 to check the values of the rest of adjacent elements in the first run.

iii. Repeat steps above for reducing elements, until there is no element need to be compared, by doing so; the value of the last element will be the biggest one of the whole.

Reference

[1] Sorting, *Data Structure*, [online] 2013, http://student.zjzk.cn/course_ware/ data_ structure/web/paixu/paixu8.3.1.1.htm (Accessed: 20 March 2014)

2. Give pseudo code for Bubble Sort that matches your description in (1).

```
set n = the amount of elements needs to be sorted

repeat

        for i = 0; i <= n; i++

                for j = i + 1; j++

                        if i > j then

                                swap i to j

                                else do not swap

                        end if

                end for

        end for

until n = 0

end
```

3. Consider this list: 5, 4, 3, 2, 1

How many comparisons in total would it take to sort this list into ascending order using the Bubble Sort you wrote pseudo code for? How many swaps would it take?

20 comparisons

4 swaps

What if the list was 1, 2, 3, 4, 5

How many comparisons in total would it take to sort this list using Bubble Sort? How many swaps would it take?

4 comparisons

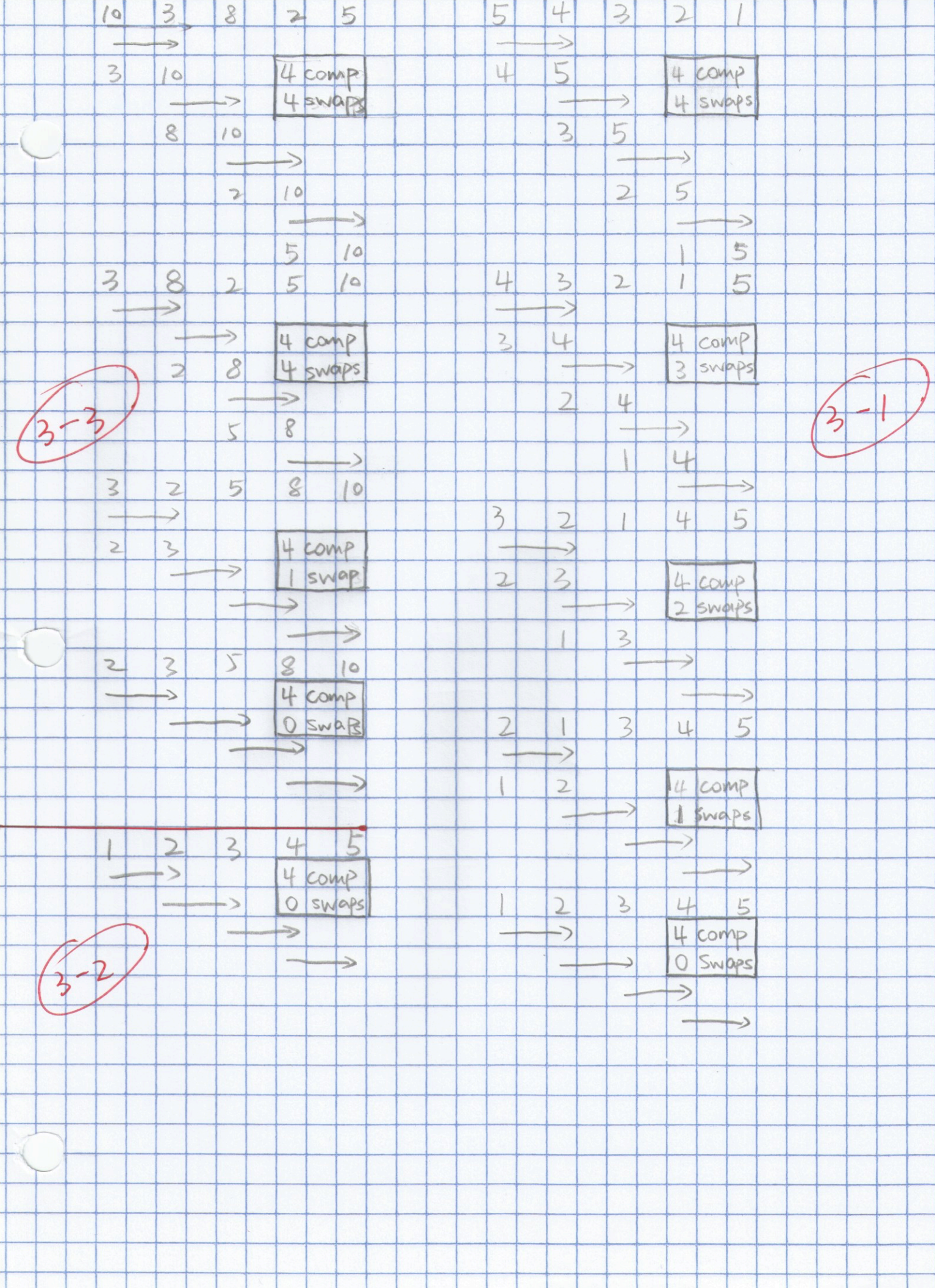
0 swaps

What about 10, 3, 8, 2, 5

How many comparisons in total, how many swaps?

16 comparisons

7 swaps


10 3 8 2 5
4 comp
4 swaps
3 10
8 10
2 10
5 10
3 8 2 5 10
4 comp
4 swaps
2 8
5 8
3-3
3 2 5 8 10
4 comp
1 swap
2 3
2 3 5 8 10
4 comp
0 swaps
1 2 3 4 5
4 comp
0 swaps
3-2
5 4 3 2 1
4 comp
4 swaps
4 5
3 5
2 5
1 5
4 3 2 1 5
4 comp
3 swaps
3 4
2 4
1 4
3-1
3 2 1 4 5
4 comp
2 swaps
2 3
1 3
2 1 3 4 5
4 comp
1 swaps
1 2
1 2 3 4 5
4 comp
0 Swaps

TASK 2: Picking the right sort of sort.

	Insertion sort	Quick sort
Space requirement	Small	Large
Worst case of number of comparisons	Assume n>n-1>n-2>…>1 n, n-1, n-2 ……., 1 n-1, n, n-2,……., 1 n-2, n-1, n,……., 1 1, ……., n-2, n-1, n By insertion sort algorithm, comparison times are (1+2+3+…..+n-1) times. The formula is n(n-1)/2.	Assume n>n-1>n-2>…..>1 n,n-1,n-2,……, 1 {n-1, n-2, …. 1} , n {n-2, ……..1}, n-1, n {1}, ……, n-1, n By quick sort algorithm, the times of comparison are (1+2+3+….. +n-1) times. So the formula is: n(n-1)/2.
Worst case number of swaps	Assume n>n-1>n-2>…>1 n, n-1, n-2 ……..,1 n-1, n, n-2,……..,1 n-2, n-1, n,……..,1 1, ……., n-2, n-1, n By insertion sort algorithm, first, compare the number of position 1 and the number of position 2. Assume the number of position 1 is greater than the number of position 2, and then swap them. Next, compare the number of position 3 to the front one. Then, arrange numbers from small to large. Based on this method, all numbers can be compared. So swap times can be counted as (1+2+3+…..+n-1) times and the formula is: n(n-1)/2.	Assume n>n-1>n-2>…..>1 n,n-1,n-2,……, 1 {n-1, n-2, …. 1} , n {n-2, ……..1}, n-1, n {1}, ……, n-1, n By quick sort algorithm, first, the whole numbers will revolved around the first one. All the numbers which less than the first one will be put before it. And then, the rest of numbers can be arranged based on this method. So, swap times can be counted as (1+2+3+….+n-1) times and the formula is: n(n-1)/2.
Ease of programming	Worst-case running time: $O(n^2)$ Best-case running time: O(n) Stable sort	Worst-case running time: $O(n^2)$ Best-case running time: O(n*logn) Unstable sort

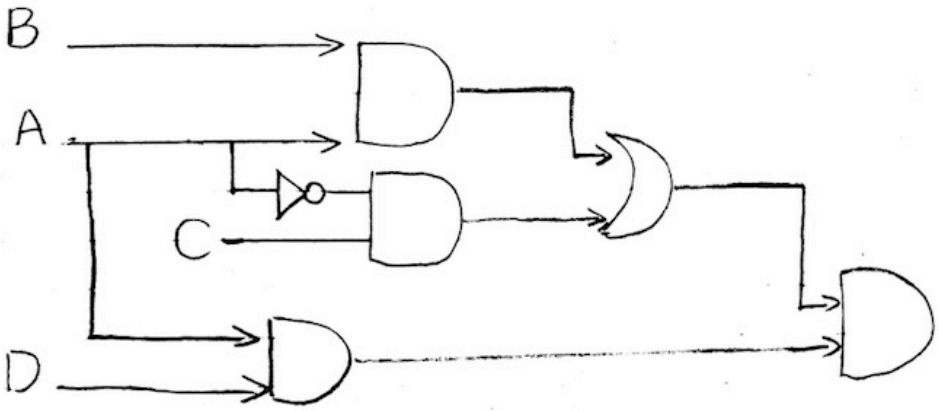
As a role-playing game, it should run fluently and accurately because there are many data contained in the game. With the development of playing, more and more date will be added in the game. As to the space requirement, quick sort can offer game a larger space for role-playing game data. Owing to the fact, quick sort is more appropriate than insertion sort for the game, because it can sort numbers more efficient.

TASK 3: Logic Gate Design

The XOR gate can be implemented using AND and OR gates according to the following formula:

((A AND B) OR (C AND NOT(A))) AND (A AND D)

1. Draw a diagram of this circuit using the logic gate symbols shown in the lab.



2). Fill in the truth table for this circuit

A	B	C	D	A∧B	C∧¬A	(A∧B)∨(C∧¬A)	A∧D	((A∧B)∨(C∧¬A))∧(A∧D)
1	1	1	1	1	0	1	1	1
1	1	1	0	1	0	1	0	0
1	1	0	1	1	0	1	1	1
1	1	0	0	1	0	1	0	0
1	0	1	1	0	0	0	1	0
1	0	1	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0
0	1	1	1	0	1	1	0	0
0	1	1	0	0	1	1	0	0
0	1	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	1	1	0	1	1	0	0
0	0	1	0	0	1	1	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0

TASK 4: Scratch – basic animation

For details, see the appendix of “Task4”.

TASK 5: Improving the Bubbler

For details, see the appendix of “myBubbler”.