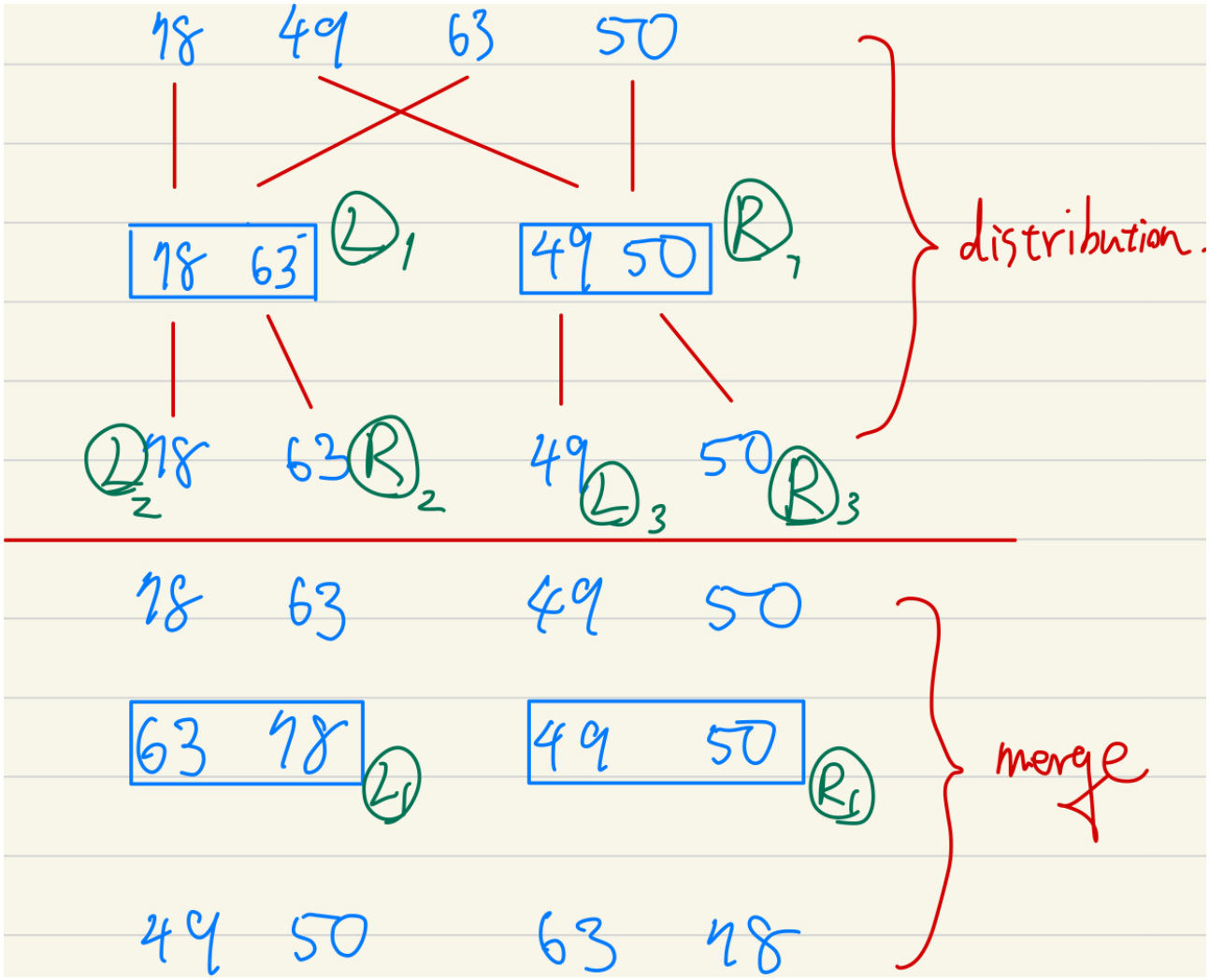
Program Assignment 3

StudentID: 0816067 Name: 李昀澤

1. How to sort with mergesort?

It's a typical example of **divide-and-conquer**. Given an array, we distribute the array to two subarray. Through the method of dividing an array given by the **program assignment 3**, the main idea of mergesort is showed below:



For array = {78, 49, 63, 50}, I first make the array into two subarray L and R. The method of construction of L and

R is to put the odd number element into L and even number element into R. Doing this until there is only one element in all subarray, and the distribution part is over. Now is to merge them up. For every L and R, compare the first element and pop the minor to the previous array. Now look at the graph above, L1 = {78, 63}, and it is distributed to L2 = {78} and R2 = {63}. Then, I compare the first element of L2 and R2, put the minor element into L1 = {63, 78}. Therefore, the L1 becomes {63, 78}. Through these steps, we can distribute R1 into L3 and R3, and merge then back to R1 = {49, 50}. Finally, merge L1 = {63, 78}, R1 = {49, 50}, since the minor of the first elements 63 and 49 is 49, so we pop it into array, next is min(63, 50), so pop 50 into array. After L and R are empty, the sorted array is done.

2. What's the complexity of mergesort?

As mergesort always divides the array into two halves and takes linear time to merge them up, the time complexity is **nlog(n)**.

3. The result of sorting 50 random numbers. (Value < 1000)

```
Before sorted:
671 880 893 916 708 68 331 362 118 369 120 854 96 569 977 703 482
272 624 541 602 947 644 60 696 741 521 70 956 429 712 542 998 549
119 436 891 836 399 37 463 860 247 862 291 346 736 767 692 789
In detail:
After sorted:
37 60 68 70 96 118 119 120 247 272 291 331 346 362 369 399 429
436 463 482 521 541 542 549 569 602 624 644 671 692 696 703 708
712 736 741 767 789 836 854 860 862 880 891 893 916 947 956 977
998
[Finished in 0.6s]
```

4. How to mergesort using c++?

The concept is just like what I mention in section 1. Through the concept, I first came up with the method of recursion. Every time I call the function **yabai_mergesort** in my source code, what I should do is first check whether the input array has only one element, if yes, then **return**. After the first checking, it's time to divide the input array into two halves. The dividing method is mentioned in section 1 so what I do is to check the index is odd or even. Then, two subarray is here. The only thing I need to do is do mergesort on these two subarray, so I call **yabai_mergesort** and the input is L and R. In this form, it will run recursively and I need to merge the sorted array L and R to the previous input array. The method of merge is also mentioned in section 1. The picture right is the detail of sorting 8 elements in an array:

```
Before sorted:
792 712 461 308 709 510 366 193
In detail:
L: 792 461 709 366
R: 712 308 510 193
L: 792 709
R: 461 366
L: 792
R: 709
Merged: 709 792
L: 461
R: 366
Merged: 366 461
Merged: 366 461 709 792
L: 712 510
R: 308 193
L: 712
R: 510
Merged: 510 712
L: 308
R: 193
Merged: 193 308
Merged: 193 308 510 712
Merged: 193 308 366 461 510 709 712 792
After sorted:
193 308 366 461 510 709 712 792
[Finished in 1.0s]
```