Yuhan Yang (001094267)

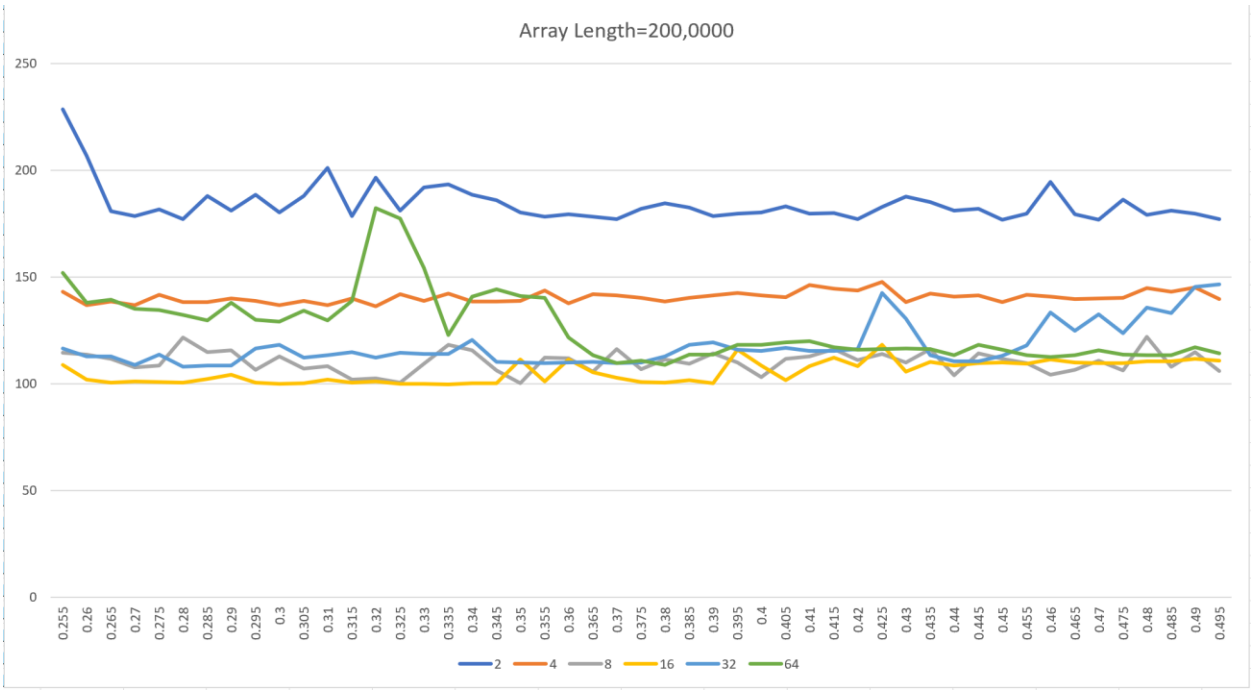
Program Structures & Algorithms

Fall 2021

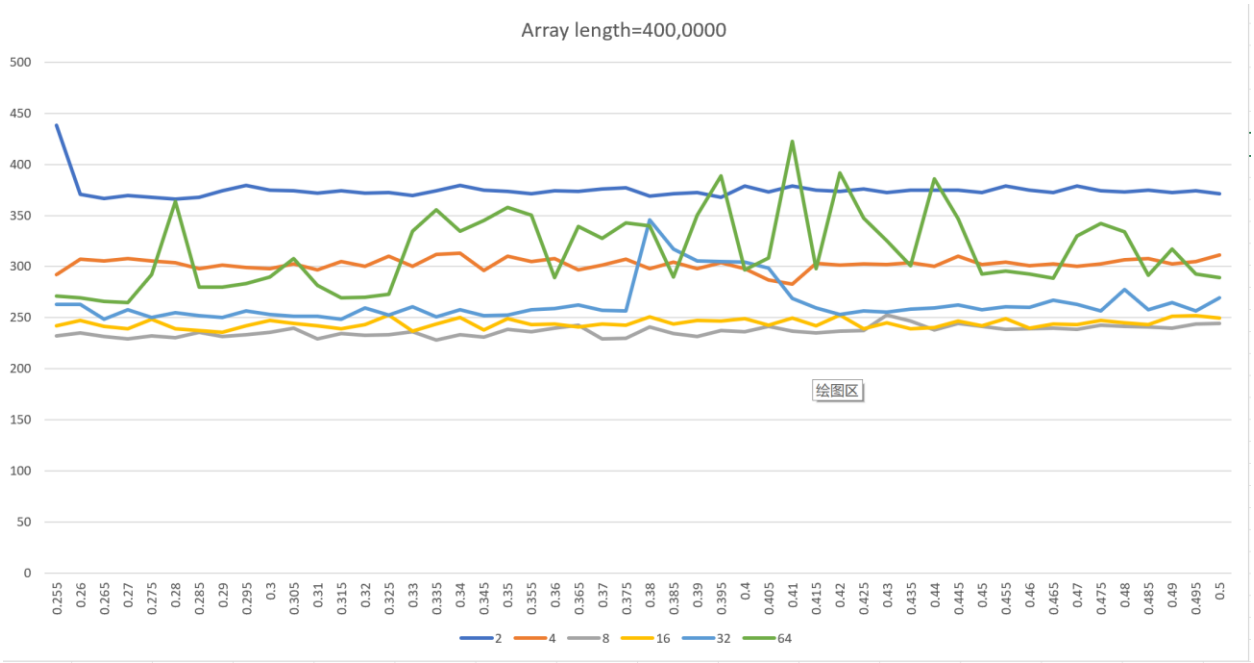
Assignment No. 5

- **Task (List down the tasks performed in the Assignment)**
  1. To experiment and come up with a good value for cut off.
  2. To decide on an ideal number (t) of separate threads (stick to powers of 2)
  3. To find an appropriate combination of these
- **Conclusion:**
  - The number of separate threads: 8
  - The good value of cut off: about 25% of the array length
- **Evidence to support the conclusion:**
  - **The number of separate threads**

    Since the performance won't change much when the number of separate threads become too big, I choose the thread number from 2 to 64.
    - **Graphical Representation**



When the array length is not very big, as we can see from the graph, the performance between thread number 8, 16 and 32 is quite ambiguous. However, this difference become much more obvious.



When the array length is 400,0000, we can see that the performance reach the best when the thread number is 8. If we add more thread into the program, the performance won't be better and even become worse.

Array length=800,0000

0.5	835.6	722.6	588.8	719.1	885.4	743.7
	810.272	**751.154**	**581.068**	**734.848**	**644.682**	**629.642**
	2	4	8	16	32	64

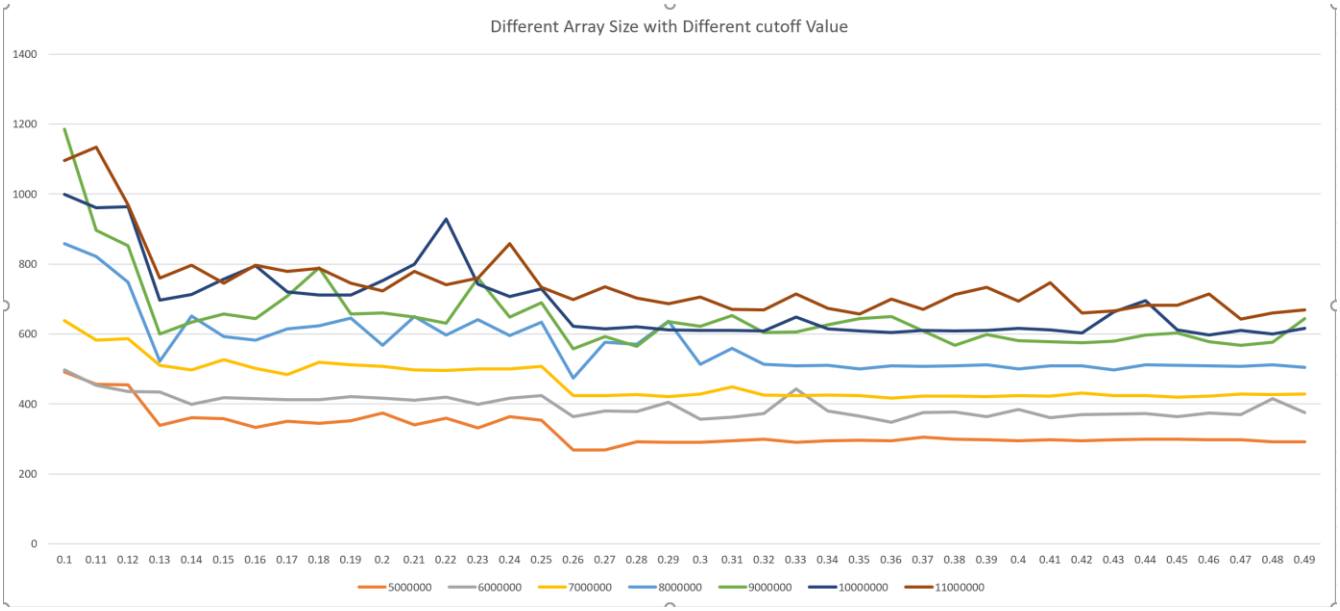
When the array length is 800,0000, it is obvious from the average time that the performance of thread number 8 is better than all the other situation.

From the evidence I showed above, I think the number of separate thread should be 8.

- **The good value of cut off**

From last section and the evidence provided above, we can draw a conclusion that the number of separate threads should be 8. So in the next step we will fix the thread number as 8 and find a good value of cut off by using the different array length.

- **Graphical Representation**



During the experiment, the length of array varies from 500,0000 to 1400,0000. Each time I add up 50,0000 to the array length. Since the performance are quite the same, I only choose some of the data to build the graph.

As we can see from the graph, the time for sorting will have the first big drop down when cutoff value reach about 12%-13% of the array length. Then when the

cutoff value reach about 25%-26% of the array length, there will be another slight drop down.

I think the reason is that when the cutoff value is 25%-26% of the array length, the last partition will be cutting the array from 4 pieces into 8 pieces, which make every thread occupied by one of the sorting processes. If we cutting into smaller pieces, there will be too much thread switching processes which may take times. But if we cutting into bigger pieces, there will be some free threads not in use.