Q1.

a. We test through the algorithm by different philosophers, which in this case are different threads. For the equal comparison, we do not set the time for thinking and eating to a random number. We set the number to be 1 second for fair comparison. Moreover, we set the iteration for each philosopher to 3 for less waiting time. Note that the experiment is done on the COE machine with 24 cores. Here is the result.

# of philosophers(threads)	Running time(ms)
3	10002.49
5	9002.23
7	9002.77
9	9002.33
11	9002.20
13	9002.12
15	9002.13

We could find that except for 3 philosophers, all the other situations are similar. So the number of philosophers does not affect the solution's running time, of course nor the correctness either. The reason that the running time of 3 philosophers is the slowest is that only 1 philosopher could eat at the time. The other 2 have to be thinking, so it is nearly unparalleled.

If there are only 3 forks in the middle, it is the same as only 2 forks in the middle, which means that only 1 philosopher could eat at the time. So now we do not need the state array in the Dijkstra algorithm. Alternatively, we need a state variable to indicate whether the 3 forks are available at the time. If available, whoever gets into the test function will eat at the time. The difficulty of this situation is when we

put down the fork. Previously, the philosopher of putting down the forks would give access to his left and right philosopher so that they could have a chance to eat. But now, we will find an algorithm to determine who will get the forks. It could be Round Robin, FIFO, or Randomly(which has a risk of waiting infinitely).

b. If the forks are still placed between philosophers and 1 philosopher has higher priority, then the result will depend on the eating time of his adjacent philosophers, because although he has higher priority, he could not eat until his adjacent philosophers finish eating. So, higher priority will not guarantee earlier finish time under this scenario. However, if there are only 3 forks in the middle, higher priority will lead to earlier finish time because every time whether a philosopher could get the forks is determined by the priority algorithm.

c. The possibility is that the eating will be more concentrated, which means that the eating is not fully interspersed between philosophers. Some philosophers may not start to eat while others have finished eating, because the rule on acquiring the forks becomes more strict so that the thinking time for some specific philosophers becomes longer.

d. Edsger Wybe Dijkstra(11 May 1930 – 6 August 2002) was a Dutch computer scientist, programmer, software engineer, systems scientist, and science essayist. He received the 1972 Turing Award for fundamental contributions to developing programming languages, and was the Schlumberger Centennial Chair of Computer Sciences at The University of Texas at Austin from 1984 until 2000. The reason why the dining problem is so important is that it illustrates the challenges of avoiding deadlock between concurrent algorithm design. Dijkstra's Algorithm uses one mutex, one semaphore per philosopher and one state variable per philosopher. The function test() and its use in take_forks() and put_forks() make the Dijkstra solution deadlock-free so that it is important in concurrent algorithm design.

References

1. Dining_philosophers_problem: https://en.wikipedia.org/wiki/Dining_philosophers_problem
2. Dining Philosopher Problem Using Semaphores: https://www.geeksforgeeks.org/dining-philosopher-problem-using-semaphores/
3. Edsger W. Dijkstra: https://en.wikipedia.org/wiki/Edsger_W._Dijkstra