DTS203TC Coursework General Feedback

Some General Issues

1. Function name, parameters and return values should be consistent with coursework requirements. Some submissions use a different name, additional parameters or print results instead of returning them. For example:
   - find_fake_coin(coins)
   - findFakeCoin(coins, start)
2. For each task, code quality is 1 mark which include readability, code formatting and comments. Some submissions contain a lot of redundant code, or defined a number of variables (e.g., named a, b, c, d, etc.), or do not contain any comments (or just one line of comment), which makes me hard to follow your implementation.
3. There might be some bugs cause your code won't run. I've tried to adjusted the code so you don't lose all the marks, but some points are deducted for the implementation, function definition or code quality part.

Task 1

Some common issues for this task are:

1. The fake coin could be heavier or lighter than other coins.
2. You should use divide and conquer to solve this problem. Every step, the coins are divided into 3 groups, and you could choose 1 of them.

Task 2

Some common issues for task 2 are:

1. To find the last node with child, you should divide by d instead of 2.
2. To sort the array, you should first build max heap. Build max heap is more efficient than simply inserting all numbers into the heap one by one.

Task 3

Some common issues for task 3 are:

1. You can't simply sort the array by wait time or charge speed and then select the stations. Greedy algorithms are faster, but cannot guarantee optimal solution here.
2. You can use dynamic programming and create a table with $n * c$ size to track the time for each station given different distances.
3. If you use an approach slower than $O(n * c * c)$, where c is 400 in our task, you lose the marks for the time complexity part.

Task 4

Some common issues for task 4 are:

1. You can use Kruskal's algorithm to find minimum spanning tree. If two endpoints in one set (form a cycle), then the edge can be used as a backup edge.
2. You should deal with cases where there are no backup edges.
3. You implementation should be efficient. No need to sort the edges multiple times when finding backup edges.

Task 5

Some common issues for task 5 are:

1. T5-1: As mentioned in lecture sessions and LM Q&A forum, the weight can be check in O(1) time for a balance. So the recurrence is $T(n) = T(n/3) + O(1)$. When solving recurrence, you should show the steps of your methods, instead of jumping directly to the results.
2. T5-2: The heapify operation also includes comparisons with d children, so the time complexity is $O(dlog_d n)$.
3. T5-3: You need to explain clearly why your algorithm ensures the correctness. Greedy algorithms may not achieve the optimal solution.
4. T5-4: It's a graph with distinct weights, so we only have one minimum spanning tree. You need to PROVE it to get full marks.
5. Some reports are not well structured and contain many typos and grammar errors, which make them hard to follow. (Report quality is 4 marks)