Problem 1

Iteration (j)	Selected Job	Start Time (s_j)	Finish Time (f_j)	t after iteration
1	4	0	1	1
2	2	1	3	3
3	1	3	6	6
4	3	6	10	10

Problem 2

(a) The release time for game G_i is influenced by the time it takes to develop all previous games, as the development is sequential. Let F_i represent the finish time for game G_i, which includes the time spent developing all previous games:

$$F_i = h_1 + h_2 + \cdots + h_i$$

Thus, the total release time for all games is:

$$F_1 + F_2 + \cdots + F_n = h_1 + (h_1 + h_2) + \cdots + (h_1 + h_2 + \cdots + h_n)$$

This can be rewritten as:

$$\text{Total Time} = h_1 \cdot n + h_2 \cdot (n-1) + \cdots + h_n \cdot 1$$

To compute the average release time, we divide the total release time by the number of games n:

$$\text{Average Release Time} = \frac{1}{n} \sum_{i=1}^{n} h_i \cdot (n - i + 1)$$

(b) To minimize the average release time, we should focus on the formula derived earlier:

$$\text{Total Time} = h_1 \cdot n + h_2 \cdot (n-1) + \cdots + h_n \cdot 1$$

It is evident that the game development times that appear earlier in the sequence contribute more heavily to the total time because they are multiplied by larger factors. Therefore, the optimal strategy is to sort the games in increasing order of their development times h_i. In other words, the company should prioritize developing the games with shorter development times first.

(c) With announcements at time t_i for each game, it is no longer possible to sort all the games at the start. The company must adopt a dynamic strategy, managing the development order as new games are announced.

The optimal strategy is as follows:

- When a new game is announced, if it has a shorter remaining development time than the remaining time of the current game being worked on, the company should pause the current game and start working on the newly announced, shorter game.

Problem 2 continued on next page...

- If the newly announced game has a longer development time than the remaining time of the current game, the company should finish the current game and then assess the next shortest game to develop, considering all announced games.

This dynamic reordering minimizes the average release time from the moment a game is announced to the time it is released. Similar techniques to insertion sort can be used to implement this dynamic strategy, which will result in $O(n^2)$ time complexity.

Problem 3

(a) We can use a greedy algorithm to minimize the number of stations. Here's the step-by-step process:

- Start from the leftmost treasure location that hasn't been covered.
- Place a station at a point 3 miles to the right of this treasure (this ensures the station covers the current treasure and possibly more to the right).
- Move to the next uncovered treasure and repeat the process until all treasures are covered.

Since each treasure location is considered only once, the algorithm runs in $O(n)$, where n is the number of treasures.

This greedy algorithm is optimal. To see why, notice that each station is placed because there was a treasure that was not previously covered. Specifically, each of these treasures needs one station, and no station can cover two such treasures simultaneously. This means that any placement covering all the treasures will require at least as many stations as the greedy algorithm uses. This is a structural property that must be satisfied, making it impossible to use fewer stations than the greedy algorithm. Therefore, the greedy approach minimizes the number of stations needed to cover all treasures.

(b) Similar to the previous case, the algorithm proceeds as follows:

- Start from the leftmost treasure location.
- Retain the rightmost station that covers this treasure and deactivate any stations in between.
- Mark all treasures covered by this station, then move to the next uncovered treasure and repeat the process.

Each treasure is checked once, and each station is considered only once. Therefore, the algorithm runs in $O(m + n)$, where m is the number of stations and n is the number of treasures.

The optimality of the algorithm follows from the same reasoning as in the previous case.

Problem 4

(a) A ship's completion time is

$$F_i = c_i + a_i$$

The time it takes for a ship to complete is simply the time from the start of its construction, to the end of its calibration. As such, the time it takes for the previous ships to be complete will not be factored in.

Therefore, the time it takes to assemble the whole fleet will be

$$\sum_{i=1}^{n} c_i + a_i$$

Problem 4 continued on next page...

However, the time it takes for a ship i to be deployed does depend on previous ship construction, since we are instead measuring the time since the beginning of assembling the fleet until the completion of a specific ship, rather than the total development time of the ships.

Therefore, the summation is different and will factor in previous development time, as

$$\sum_{j=1}^{i} c_j + a_j$$

(b) Since this problem focuses on the time it takes for a ship to be deployed, rather than just the time it takes for all ships to be complete, the time it takes for previous ships to complete is important, like in question 2.

The time it takes for a ship to be deployed is from the beginning of the fleet's assembly to the end of that ship's calibration, so it makes sense to finish the ships that take the least time first, so that they can be deployed as fast as possible.

See 2b for more information on the methodology behind sorting from shortest to longest completion time.

(c) With the new calibration technology, the calibration for all ships can be done simultaneously. This technology significantly reduces the total preparation time because it eliminates the need to wait for one ship's calibration to finish before starting the next.

To fully utilize this technology, the optimal strategy is to complete the hull construction of all ships sequentially, after which calibration will occur automatically. Since calibration is independent of hull construction once a ship is built, the focus shifts to minimizing the wait time from hull construction.

Notice that hull construction still depends on the sequence of ships being built. The best strategy to minimize the total time is to construct ships in order of increasing construction times, i.e., sorting ships by their construction time c_i.

Once ships are sorted by increasing hull construction time, the total time it takes to prepare ship i is given by:

$$F_i = a_i + \sum_{j=1}^{i} c_j$$

where c_j is the hull construction time for ship j, and a_i is the calibration time for ship i. The total average time (including both hull construction and calibration) is:

$$\frac{1}{n}\sum_{i=1}^{n} F_i = \frac{1}{n}\sum_{i=1}^{n}\left(a_i + \sum_{j=1}^{i} c_j\right) = \frac{1}{n}\sum_{i=1}^{n} a_i + \sum_{i=1}^{n} c_i \frac{n-i+1}{n}$$