Algorithms and Data Structures 1

Student names: Jacopo Ceccuti s215158
Hand-in: Superheros

1 Exercise

1.1

The array A contains the costs of all superheros. To find the cheapest team T of 7 superheroes we can start by taking the array and applying the merge-sort algorithm to sort the array in ascending order. Now we just have to return the value of the sum of the first 7 elements. The time complexity (worst case) of this algorithm in big "Oh" notation is $O(m \cdot log(m))$, where m is the length of the array A. This is given by:

$$T(m) = O(m \cdot log(m)) + O(7)$$

1.2

To find the cheapest team T of $\lfloor \sqrt{m} \rfloor$ superheroes we can start by taking the array and applying the merge-sort algorithm to sort the array in ascending order. Now we just have to return the value of the sum of the first $\lfloor \sqrt{m} \rfloor$ elements. The time complexity (worst case) of this algorithm in big "Oh" notation is $O(m \cdot log(m))$, where m is the length of the array A. This is given by:

$$T(m) = O(m \cdot log(m)) + O(\lfloor \sqrt{m} \rfloor)$$

1.3

To find the number of distinct costs of superheros we can start by taking the array A and applying the merge-sort algorithm to sort the array in ascending or descending order. Now we can use a two pointer method to eliminate the duplicates and count how many distinct costs are present in the array A. Initialize a variable "counter" to one if the array has length grater than zero. Set the unique pointer to the index one (considering a 0-indexed array type) and the current pointer at index zero. Now we will move the current pointer through every element of the array comparing it with the unique pointer at every step. We can have two cases:

- They are equal, therefore we need to update the current pointer and continue.

- They are different, we need to set the unique pointer to the value of the current pointer, add one to our counter, update the current pointer and continue.

At the end our counter will store the number of unique prices of superheros. The time complexity (worst case) of this algorithm in big "Oh" notation is $O(m \cdot log(m))$, where m is the length of the array A. This is given by:

$$T(m) = O(m \cdot log(m)) + O(m)$$