One-Day Assignment 6 – algorithmic solution

Planks

One Day Assignment 6 – Algorithm

- When a new plank is added, add it to your data structure – (1)
- When you have a **c X** query:
  - Find plank **A** from data structure – (2)
  - Find plank **B** from data structure – (2)
  - Remove plank **A** and plank **B**
  - Calculate effort **E** from weight and length of **A** and **B**
    - store in **long** variable to prevent overflow
  - Print value of **E**
    - printing directly without storing/casting to **long** may result in overflow

One Day Assignment 6 – Representation

- The issue of TreeSet not supporting duplicate elements is the main concern for this problem
- The following slide documents a possible way to handle duplicates

One Day Assignment 6 – Representation

- Use TreeSet<IntegerTriple>
- IntegerTriple contains 3 values:
  - First value: weight of the plank
  - Second value: length of the plank
  - Third value: special unique id (to ensure we can support duplicates)
- (1) – Add new IntegerTriple (*weight*, *length*, id++)
- (2) – floor() of IntegerTriple (-INF, *targetLength*, -INF), where INF is a large integer
- (3) – ceiling() of IntegerTriple (INF, *targetLength*, INF), where INF is a large integer

One-Day Assignment 6 – Comparison

- You need to implement your own `compareTo method (or Comparator class)` for your TreeSet / TreeMap
- You need to determine how to compare a pair of Planks
  - What value do I compare first?
  - How do determine which Plank is smaller / greater?