Solution for Task #1

The main issue is to find out the pairs of ratings with the lowest difference. And at the same time we need to satisfy the core requirement of the task – all items should have a pair item. In this task we need to consider 2 cases:

1. <u>The number of teams is even.</u>
   Statement - the best pairs can be obtained just by passing through the sorted array. Let's consider the abstract sequence:

   $k_{i-1} < k_i < k_{i+1}$

   We need to prove that the proper choice is to throw k_{i-1} and k_i, even if the difference of k_i and k_{i+1} is lower:

   $diff_{i,i+1} < diff_{i-1,i}$

   If we generate a pair with k_i and k_{i+1} elements difference between k_{i-1} and k_{i+2} will be computed by sum of difference of all thrown elements:

   $diff_{i-1,i+2} = diff_{i-1,i} + diff_{i,i+1} + diff_{i+1,i+2}$

   In the best case, when $diff_{i-1,i+2} < diff_{i+2,i+3}$, we will generate k_{i-1} and k_{i+2}, and the final sum of the generated pairs will be:

   $diff_{i,i+1} + diff_{i-1,i+2} = diff_{i,i+1} + diff_{i-1,i} + diff_{i,i+1} + diff_{i+1,i+2}$

   And this sum will be definitely bigger than if we choose k_{i-1} and k_i to be generated:

   $diff_{i-1,i} + diff_{i+1,i+2} < diff_{i,i+1} + diff_{i-1,i} + diff_{i,i+1} + diff_{i+1,i+2}$

2. The number of items is odd.

The main issue here is to find out the odd item in the sorted array with the highest difference and thrown it out. And after that apply the even approach.

The Big O complexity of the algorithm:

n*log(n) + n = n*log(n), for even number of items,

n*log(n) + 2*((n-2)/2) + n = n*log(n), for odd number

As we can see, the complexity depends on sorting(in our case TimSort), and equals n*log(n).