Qiyao Zhou

Z5379852

Question 2

2.1:

1.Due to the fact that $a \leqslant b \leqslant c \leqslant d$, it is easy to get $b - c \leqslant 0$ and $c - d \leqslant 0$.

2. $a \leqslant d$, $b - c \leq 0 \Leftrightarrow a(b-c) \geqslant d(b-c) \Leftrightarrow ab - ac \geqslant bd - cd \Leftrightarrow ab + cd \geqslant ac + bd$

$a \leqslant b$, $c - d \leq 0 \Leftrightarrow a(c-d) \geqslant b(c-d) \Leftrightarrow ac - ad \geqslant bc - bd \Leftrightarrow ac + bd \geqslant ad + bc$

3. $ab + cd \geqslant ac + bd$, $ac + bd \geqslant ad + bc \Leftrightarrow ab + cd \geqslant ac + bd \geqslant ad + bc$

2.2:

Combining the question with the analysis in 2.1, to get the maximum score we need to form pairs of high and high scores and pairs of low and low scores, so we only need to sort the 2k scores once and then traverse the sequence once in order, taking two scores at a time to form a pair, so that the resulting pairs of scores can achieve the maximum. The following is the detailed algorithm.

The 2k scores are sorted using a reductive sorting algorithm, which splits the original sequence into two halves, then recursively sub sorts the subsequences until there is only one number in the subsequence, then recursively merges the subsequences, comparing the heads of the two subsequences, and the smaller one enters the merged sequence, resulting in a non-decreasing sequence of scores.

Iterate through the sorted sequence, taking two scores at a time as a pair.

The result of the pairing can obtain the maximum score.

In terms of time complexity analysis, the time complexity of the subsumption sort $T(2k) = 2T(k) + k$, recursively this equation yields $T(2k) \in O(k \log k)$, while the time complexity of the traversal is $O(k)$, so the total time complexity is $O(k \log k)$.

2.3:

Compared with 2.2, the change here is only from finding the largest score pair to finding the smallest score pair, combining 2.1 and 2.3, the algorithm is still to first sort the score sequence by merging, then create a pointer at each end of the sorted sequence and traverse to the middle, each time the two pointers point to the score to form a team, so that the score pairs obtained when the two pointers are adjacent to each other can meet the requirements of the question. Such an algorithm has the same time complexity as 2.2, for $O(k \log k)$.