CodeExpert Beispielaufgabe

Maximum Subset

You are given two integers n and k and an array A of n distinct integers. You want to find the maximum number of elements of A that you can pick such that all of their values are at most k apart from each other (that is, the maximum value that you pick minus the minimum value that you pick must be at most k).

For example, for $n = 7$, $k = 4$, and $A = [5, 1, 12, 6, 9, 11, 8]$, the answer is 4, and it is obtained by picking the elements with value 5, 6, 9, and 8; these values are at most 4 apart.

You need to implement your solution as a method **maxSubset(n, A)** in the file Main.java. You get one point for each passing test set. To pass both test sets, your solution is expected to run in time $O(n \log n)$.

Hint: Sort the array. You are allowed to create new methods if that is helpful to you.

Attention: You are **NOT** allowed to use additional imports, other than the imports already included in the code template.

```
class Main {
  public static void main(String[] args) {
    // Uncomment the following two lines if you want to read from a file.
    In.open("public/example.in");
    Out.compareTo("public/example.out");

    int n = In.readInt();
    int k = In.readInt();
    int[] A = new int[n];
    for (int i = 0; i < n; i++) {
      A[i] = In.readInt();
    }
    Out.println(maxSubset(n, k, A));

    // Uncomment this line if you want to read from a file
    // In.close();
  }

  public static int maxSubset(int n, int k, int[] A) {
    mergeSort(A, 0, n-1);
    int mx = 0;
    int ind = 0;
    for(int i=0; i<n; i++){
      while(A[i] - A[ind] > k) ind++;
      mx = Math.max(mx, i-ind+1);
    }
    return mx;
  }
```

```
  public static void mergeSort(int[] A, int l, int r){
    if(l<r){
      int m = (l+r)/2;
      mergeSort(A, l, m);
      mergeSort(A, m+1, r);
      merge(A, l, m, r);
    }
  }

  public static void merge(int[] A, int l, int m, int r){
    int[] B = new int[r-l+1]; //helper array
    int i = l; //current position in left subarray
    int j = m+1; //current position in right subarray
    int k = 0; //current position in helper array

    while(i<=m && j<=r){
      if(A[i]<A[j]){
        B[k] = A[i]; i++;
      } else {
        B[k] = A[j]; j++;
      }
      k++;
    }
    //if one subarray has already been copied completely, copy rest
    if(i>m){
      for(int a = j; a<=r; a++) B[k+a-j] = A[a];
    } else {
      for(int a = i; a<=m; a++) B[k+a-i] = A[a];
    }
    //copy B into A
    for(int a=l; a<=r; a++) A[a] = B[a-l];
  }
}
```