Q1 Special Sorting

10 Points

Consider the problem of sorting an array $A[1, ..., n]$ of integers. We presented an O(n log n)-time algorithm in class and, also, proved a lower bound of $\Omega(nlogn)$ for any comparison-based algorithm.

Q1.1

5 Points

Give an efficient sorting algorithm for an array C[1,...,n] whose elements are taken from the set {1,2,3,4,5,6,7}.

Q_1.1.pdf

Download

1 / 1

Q1.2

5 Points

Give an efficient sorting algorithm for an array $D[1, \ldots, n]$ whose elements are distinct $(D[i] \neq D[j]$, for every $i \neq j \in \{1, \ldots, n\})$ and are taken from the set $1, 2, \ldots, 2n$.

▾ Q1.2.pdf Download

Q2 Linear Sorting

10 Points

In case you designed linear-time sorting algorithms for any subpart of problem 1, does it mean that the lower bound for sorting of $\Omega(nlogn)$ is wrong? Explain.

I'll have some additional explanations here. Because the O(nlog n) that the professor taught in class is an array for the general case, but the array like in Q1.1 is special, we can achieve the time complexity of O(n), so it is not wrong to say that the lower bound is Ω(nlog n).

Q2.pdf Download

Q3 Closest Pair

10 Points

We have learned an algorithm that solves the Closest pair problem in $2D$ in $\Theta(nlogn)$ time. (Closest pair problem in $2D$: Given n points in the 2D plane, find a pair with smallest Euclidean distance between them.)

Give an algorithm that solves the Closest pair problem in $3D$ in $\Theta(nlogn)$ time. (Closest pair problem in $3D$: Given n points in the $3D$ space, find a pair with smallest Euclidean distance between them.)

Q3.pdf Download

1 / 4

ClosestPair.java Download

```java
import util.Point3;

public class ClosestPair {
  private Point3[] px, pz;
  public Point3[] solve(Point3[] points) {
    int n = points.length;

    px = new Point3[n];
    pz = new Point3[n];
    System.arraycopy(points, 0, px, 0, n);
    System.arraycopy(points, 0, pz, 0, n);
    Arrays.sort(px, Comparator.comparingLong(o -> o.x));
    Arrays.sort(pz, Comparator.comparingLong(o -> o.z));

    for (int i = 0; i < n; i++) {
      px[i].idx = i;
    }

    Point3[] res = find(0, n - 1, pz);

    return getDis(res[0], res[1]);

  }

  private Point3[] find(int x1, int x2, Point3[] pz){
    switch (x2 - x1 + 1) {
      case 2:
        return new Point3[]{px[x1], px[x2]};
      case 3:
        double dis12 = getDis(px[x1], px[x1 + 1]);
        double dis23 = getDis(px[x1 + 1], px[x2]);
        double dis13 = getDis(px[x1], px[x2]);
        if (dis12 < dis23) {
          if (dis12 < dis13) {
            return new Point3[]{px[x1], px[x1 + 1]};
          } else {
            return new Point3[]{px[x1], px[x2]};
          }
        } else {
          if (dis23 < dis13) {
            return new Point3[]{px[x1 + 1], px[x2]};
          } else {
            return new Point3[]{px[x1], px[x2]};
          }
        }
    }

    int mi = (x1 + x2) / 2;
    int idx1 = 0;
    int idx2 = 0;
    Point3[] qz = new Point3[mi - x1 + 1];
    Point3[] rz = new Point3[x2 - mi];

    for (Point3 p : pz) {
      if (p.idx <= mi) {
        qz[idx1++] = p;
      } else {
        rz[idx2++] = p;
      }
    }

    Point3[] left = find(x1, mi, qz);
    Point3[] right = find(mi + 1, x2, rz);

    double dis1 = getDis(left[0], left[1]);
    double dis3 = getDis(right[0], right[1]);
    double delta = Math.min(dis1, dis3);

    long x = px[mi].x;

    ArrayList<Point3> s1 = new ArrayList<>();
    ArrayList<Long> segRef = new ArrayList<>();
    ArrayList<Integer> sl1 = new ArrayList<>();
    ArrayList<Integer> sl2 = new ArrayList<>();
    HashMap<Long, ArrayList<Point3>> s2 = new
HashMap<>();
    construct(pz, x, mi, delta, s1, segRef, sl1, sl2,
s2);

    Point3[] pairMin = new Point3[2];
    double disMin = delta;

    for (int i = 0; i < s1.size(); i++) {
      Point3 s = s1.get(i);
      long segIdx = segRef.get(i);
      int loc1 = sl1.get(i);
      int loc2 = sl2.get(i);

      ArrayList<Point3> seg;

      seg = s2.get(segIdx);
      if (!seg.isEmpty()){
        for (int j = Math.max(0, loc1 - 16); j <
seg.size() && j <= loc1 + 16; j++) {
          Point3 st = seg.get(j);
          double tmp = getDis(s, st);
          if (tmp < disMin) {
            disMin = tmp;
            pairMin[0] = s;
            pairMin[1] = st;
          }
        }
      }

      seg = s2.get(segIdx + 1);
      if (!seg.isEmpty()) {
        for (int j = Math.max(0, loc2 - 16); j < seg.size() && j <= loc2 + 16; j++) {
          Point3 st = seg.get(j);
          double tmp = getDis(s, st);
          if (tmp < disMin) {
            disMin = tmp;
            pairMin[0] = s;
            pairMin[1] = st;
          }
        }
      }
    }

    if (pairMin[0] != null) {
      return pairMin;
    } else if (dis1 < dis3) {
      return left;
    } else {
      return right;
    }
  }

  private static void construct(Point3[] pz, long x, int idx, double delta,
                                ArrayList<Point3> s1,
                                ArrayList<Long> segRef,
                                ArrayList<Integer> sl1,
                                ArrayList<Integer> sl2,
                                HashMap<Long, ArrayList<Point3>> s2) {

    long yMin = Long.MAX_VALUE;
    for (Point3 p : pz) {
      if (yMin > p.y) {
        yMin = p.y;
      }
    }

    for (Point3 p : pz) {
      if (Math.abs(x - p.x) > delta) {
        continue;
      }

      long no = (long)((p.y - yMin) / delta);
      if (p.idx <= idx) {
        s1.add(p);

        if (no == 0) {
          no = -1;
        } else {
          no = (no - 1) / 2;
        }
        segRef.add(no);

        ArrayList<Point3> arr;
        arr = s2.computeIfAbsent(no, k -> new ArrayList<>());
        sl1.add(Math.max(arr.size()-1, 0));
        arr = s2.computeIfAbsent(no+1, k -> new ArrayList<>());
        sl2.add(Math.max(arr.size()-1, 0));

      } else {
        no = no / 2;
        s2.computeIfAbsent(no, k -> new ArrayList<>()).add(p);
      }
    }
  }
}
```

Point3.java Download

```
public class Point3 {
        public int idx;
        public long x, y, z;

        public Point3(long x, long y, long z) {
                this.x = x;
                this.y = y;
                this.z = z;
        }

        public static double getDis(Point3 p1, Point3 p2) {
                long tmp1 = p1.x - p2.x;
                long tmp2 = p1.y - p2.y;
                long tmp3 = p1.z - p2.z;
                return Math.sqrt(tmp1 * tmp1 + tmp2 * tmp2 + tmp3 * tmp3);
        }

}
```

Q4 Programming

10 Points

For this question, you will solve the Median of Two Sorted Arrays problem in leetcode (leetcode problem No. 4). The overall run time complexity should be $O(log(m+n))$. Please see the link below.

https://leetcode.com/problems/median-of-two-sorted-arrays/

Q4.1

8 Points

Please submit your code here (note your code will be checked for correctness and overall quality.

Note that your source files should be in the programming language you are using e.g. .py, .java , .c etc)

Q4.1.java   Download

```java
class Solution {
    public double findMedianSortedArrays(int[] nums1, int[] nums2) {
        int m = nums1.length;
        int n = nums2.length;
        int start = 0;
        int end = m;

                if(m > n)
        {
                        return findMedianSortedArrays(nums2, nums1);
                }
        while(start <= end)
                {
            int i = start + (end - start) / 2;
            int j = (m + n + 1) / 2 - i;

                        if(i > start && nums1[i - 1] > nums2[j])
                        {
                end = i - 1;
            }
                        else if(i < end && nums1[i] < nums2[j - 1])
                        {
                start = i + 1;
            }
                        else
                        {
                    int maxLeft = Integer.MIN_VALUE;
                    int minRight = Integer.MAX_VALUE;

                                    if(i - 1 >= 0) maxLeft
= Math.max(maxLeft, nums1[i - 1]);
                    if(j - 1 >= 0) maxLeft  =
Math.max(maxLeft, nums2[j - 1]);
                    if(i <= m - 1) minRight =
Math.min(minRight, nums1[i]);
                    if(j <= n - 1) minRight =
Math.min(minRight, nums2[j]);

                    return (m + n) % 2 == 0 ? (maxLeft +
minRight) / 2.0 : maxLeft;
                }
            }
            return 0.0;
        }
}
```

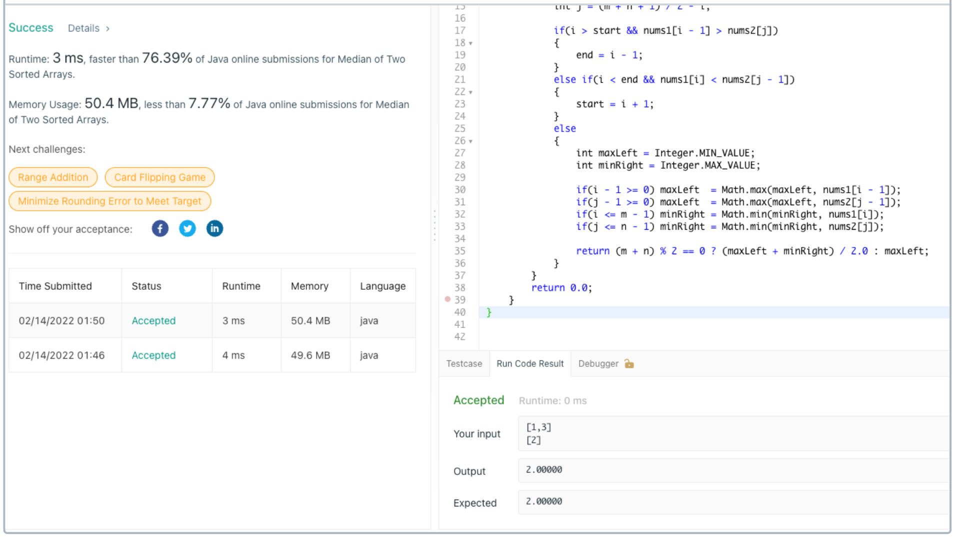
Q4.2

2 Points

Please submit an image from leetcode showing whether your solutions was accepted and your outputs.

Q4.2.png

Download



Success Details ›

Runtime: 3 ms, faster than 76.39% of Java online submissions for Median of Two Sorted Arrays.

Memory Usage: 50.4 MB, less than 7.77% of Java online submissions for Median of Two Sorted Arrays.

Next challenges:

Range Addition · Card Flipping Game · Minimize Rounding Error to Meet Target

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
02/14/2022 01:50	Accepted	3 ms	50.4 MB	java
02/14/2022 01:46	Accepted	4 ms	49.6 MB	java

```java
            if(i > start && nums1[i - 1] > nums2[j])
            {
                end = i - 1;
            }
            else if(i < end && nums1[i] < nums2[j - 1])
            {
                start = i + 1;
            }
            else
            {
                int maxLeft = Integer.MIN_VALUE;
                int minRight = Integer.MAX_VALUE;

                if(i - 1 >= 0) maxLeft  = Math.max(maxLeft, nums1[i - 1]);
                if(j - 1 >= 0) maxLeft  = Math.max(maxLeft, nums2[j - 1]);
                if(i <= m - 1) minRight = Math.min(minRight, nums1[i]);
                if(j <= n - 1) minRight = Math.min(minRight, nums2[j]);

                return (m + n) % 2 == 0 ? (maxLeft + minRight) / 2.0 : maxLeft;
            }
        }
        return 0.0;
    }
}
```

Accepted Runtime: 0 ms

```
Your input   [1,3]
             [2]
Output       2.00000
Expected     2.00000
```

Problem Set 4

● GRADED

STUDENT
Kejian Tong

TOTAL POINTS
40 / 40 pts

QUESTION 1

Special Sorting		**10** / 10 pts
1.1	(no title)	**5** / 5 pts
1.2	(no title)	**5** / 5 pts

QUESTION 2

Linear Sorting	**10** / 10 pts

QUESTION 3

Closest Pair	**10** / 10 pts

QUESTION 4

Programming		**10** / 10 pts
4.1	(no title)	**8** / 8 pts
4.2	(no title)	**2** / 2 pts