CSC-321 Design and Analysis of Algorithms
Section 401
Fall 2019-20

Instructor: Iyad Kanj
Office: CDM 832
Phone: (312) 362-5558
Email: `ikanj@cs.depaul.edu`
Office Hours: Monday & Wednesday 3:00 - 4:30
Course Website: https://d2l.depaul.edu/

Programming Assignment

(Due October 28)

Implement the algorithm for the closest pair of points (based on the lecture notes on D2L). Your program should input the coordinates of the points from (each of) the test files uploaded on D2L, and outputs: (1) the coordinates of a closest pair in the point-set, and (2) the distance between the two points in the closest pair.

Instructions

1. You can use any *standard* programming language (C, C^{++}, $C\#$, Java, Python, etc.).
2. Test your code on the test files in D2L (located in the same folder as the assignment), and compare your solutions to the provided solutions (same folder). The grader will test your program on the uploaded test files (text files). Make sure that your program runs on these test files.
3. Submit all the files containing your source code. Make sure that the files compile and run.
4. Please create a single ".zip" file containing all your submission files and upload it on D2L.

CSC-321 Design and Analysis of Algorithms
Section 401
Fall 2019-20

Instructor: Iyad Kanj
Office: CDM 832
Phone: (312) 362-5558
Email: `ikanj@cs.depaul.edu`
Office Hours: Monday & Wednesday 3:00 - 4:30
Course Website: https://d2l.depaul.edu/

Programming Assignment

(Due October 28)

Implement the algorithm for the closest pair of points (based on the lecture notes on D2L). Your program should input the coordinates of the points from (each of) the test files uploaded on D2L, and outputs: (1) the coordinates of a closest pair in the point-set, and (2) the distance between the two points in the closest pair.

Instructions

1. You can use any *standard* programming language (C, C^{++}, $C\#$, Java, Python, etc.).
2. Test your code on the test files in D2L (located in the same folder as the assignment), and compare your solutions to the provided solutions (same folder). The grader will test your program on the uploaded test files (text files). Make sure that your program runs on these test files.
3. Submit all the files containing your source code. Make sure that the files compile and run.
4. Please create a single ".zip" file containing all your submission files and upload it on D2L.

Closest Pair of Points Algorithm and Its Running Time

Let $S = \{p_1, \ldots, p_n\}$ be a set of points, where $p_i = (x_i, y_i)$, for $i = 1, \ldots, n$. Let X be a list containing the points in S sorted w.r.t. their x-coordinates, and Y a list containing the points in S sorted w.r.t. their y-coordinates. Clearly, X and Y can be obtained in $\mathcal{O}(n \lg n)$ time.

```
Closest-Pair-Algo

1. if $|S| \leq 3$ return a closes pair $(p_{min}, q_{min})$ in $S$ by brute force;
2. using $X$, compute a vertical line $D$ of equation $x = \ell$ that partitions $S$ into $S_L, S_R$ of roughly-equal size
   such that all points in $S_L$ are on $D$ or to the left of it, and all points in $S_R$ are on $D$ or to the right of it;
3. using $X$ and $Y$, create the arrays $X_L, Y_L$ and $X_R, Y_R$;
4. recurse on $S_L, X_L, Y_L$ to compute a closest pair $(p_L, q_L)$; let $\delta_L = |p_L q_L|$;
5. recurse on $S_R, X_R, Y_R$ to compute a closest pair $(p_R, q_R)$; let $\delta_R = |p_R q_R|$;
5. let $\delta = \min\{\delta_L, \delta_R\}$;
6. let $S_{mid}$ be the set of points in $S$ whose $x$-coordinate satisfies $\ell - \delta \leq x \leq x + \delta$;
7. using $Y$, compute the list of points in $S_{mid}$ sorted by their $y$-coordinates;
8. go over $Y_{mid}$ (in the sorted order), and for each point, compute its distance to the next (at most) 7 points
   in $Y_{mid}$ and keep track of the pair of points $(p_{mid}, q_{mid})$ of minimum distance;
9. return the closest pair $(p_{mid}, q_{min})$ among, $(p_L, q_L), (p_R, q_R)$, and $(p_{mid}, q_{mid})$;
```

Figure 1: The algorithm Closest Pair.

Let $T(n)$ be the running time of Closest-Pair in the worst case on n points. Step 1 takes constant time. Steps 2-3 take $\mathcal{O}(n)$ time. Steps 4-5 result on two calls to the same algorithm, but on $n/2$ points each. Steps 6-9 can be implemented in $\mathcal{O}(n)$ time. Therefore, $T(n)$ obeys the following recurrence relation:

$$T(n) = \begin{cases} \mathcal{O}(1) & \text{if } n \leq 3 \\ 2T(n/2) + \mathcal{O}(n) & \text{otherwise} \end{cases}$$

We can solve $T(n)$ using the master method to obtain $T(n) = \mathcal{O}(n \lg n)$.

CSC-321 Design and Analysis of Algorithms
Section 401
Fall 2019-20

Instructor: Iyad Kanj
Office: CDM 832
Phone: (312) 362-5558
Email: `ikanj@cs.depaul.edu`
Office Hours: Monday & Wednesday 3:00 - 4:30
Course Website: https://d2l.depaul.edu/

Programming Assignment

(Due October 28)

Implement the algorithm for the closest pair of points (based on the lecture notes on D2L). Your program should input the coordinates of the points from (each of) the test files uploaded on D2L, and outputs: (1) the coordinates of a closest pair in the point-set, and (2) the distance between the two points in the closest pair.

Instructions

1. You can use any *standard* programming language (C, C^{++}, $C\#$, Java, Python, etc.).
2. Test your code on the test files in D2L (located in the same folder as the assignment), and compare your solutions to the provided solutions (same folder). The grader will test your program on the uploaded test files (text files). Make sure that your program runs on these test files.
3. Submit all the files containing your source code. Make sure that the files compile and run.
4. Please create a single ".zip" file containing all your submission files and upload it on D2L.