Homework 3

Help Center

The **hard deadline** for this homework is **Sun 26 Jul 2015 6:01 PM EDT**.

In Module 3, we will consider two methods for clustering set of points in the plane. One of these methods (hierarchical clustering) will rely on a fast divide-and-conquer method for computing the closest pair of points from a set of points.

In doing the Homework and preparing for the Project and Application, we suggest that you review the class notes on closest pairs of points and clustering methods. You may also want to print out this short summary of the pseudo-code that we will use in this Module.

□ **In accordance with the Coursera Honor Code, I (Kumar Aakash) certify that the answers here are my own work.**

Question 1

Given an array $A[0 \ldots n-1]$, an *inversion* is a pair of indices (i, j) such that $0 \leq i < j \leq n-1$ and $A[i] > A[j]$. In Questions 1-5, we will consider the problem of counting the number of inversions in an array with n elements.

To begin, how many inversions are there in the array $A = [5, 4, 3, 6, 7]$? Enter your answer as a number in the box below.

3

Question 2

In an array with n elements, what is the maximum possible number of inversions? Enter your answer below as a math expression in terms of n.

(n*(n-1))/2

Preview Help

Question 3

What is the best case running time of a brute-force algorithm that counts the number of inversions in an array with n elements by checking every pair of elements in the array? Choose the tightest big-O bound for this best case time listed below.

○ $O(n^3)$

○ $O(n \log n)$

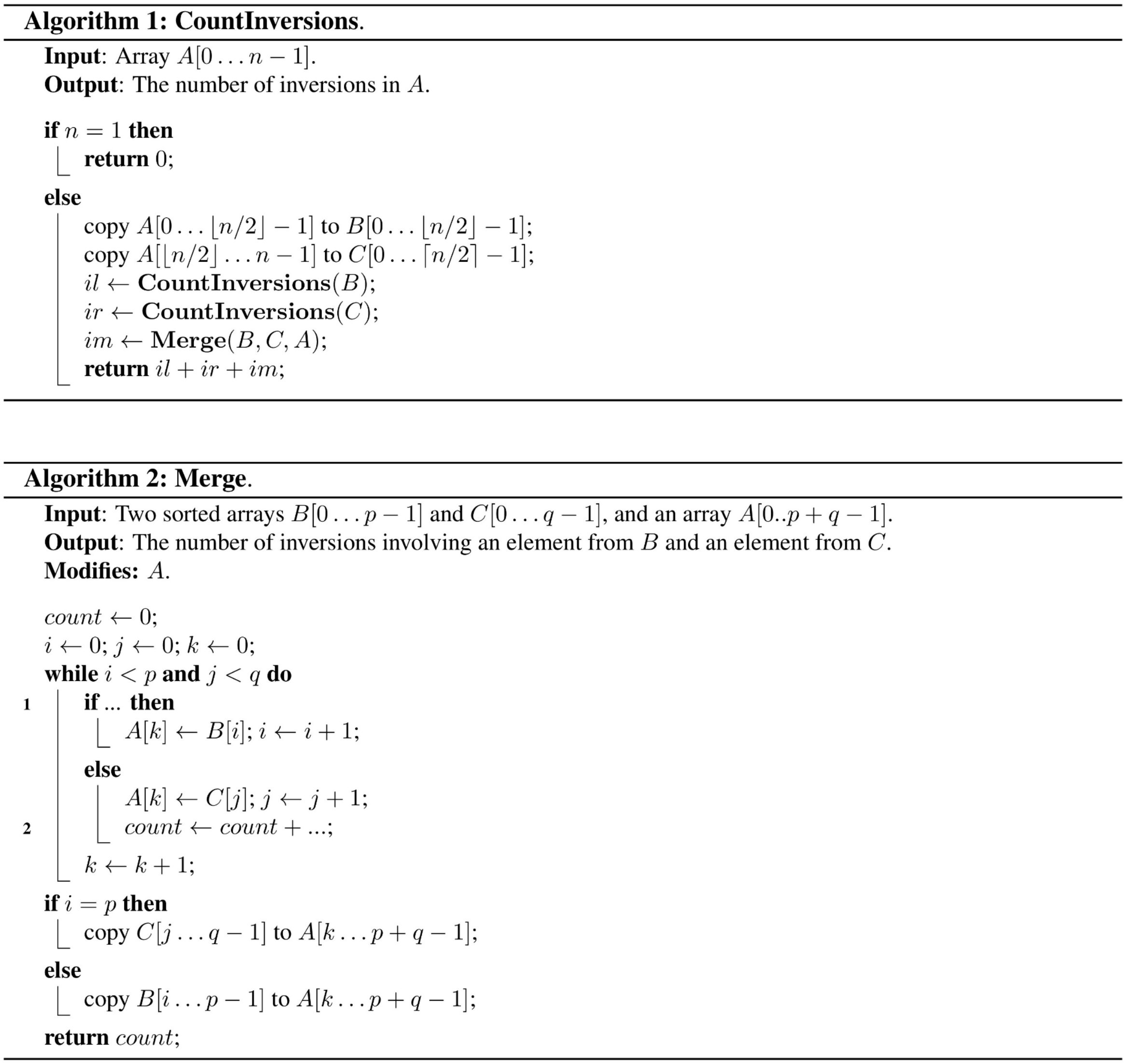
○ $O(1)$

● $O(n^2)$

○ $O(n)$

Question 4

In Questions 4 and 5, we will consider the following divide-and-conquer algorithm for counting the number of inversions in an array A.



```
Algorithm 1: CountInversions.
  Input: Array $A[0 \dots n-1]$.
  Output: The number of inversions in $A$.

  if $n = 1$ then
      return 0;
  else
      copy $A[0 \dots \lfloor n/2 \rfloor - 1]$ to $B[0 \dots \lfloor n/2 \rfloor - 1]$;
      copy $A[\lfloor n/2 \rfloor \dots n-1]$ to $C[0 \dots \lceil n/2 \rceil - 1]$;
      $il \leftarrow$ CountInversions($B$);
      $ir \leftarrow$ CountInversions($C$);
      $im \leftarrow$ Merge($B, C, A$);
      return $il + ir + im$;
```

```
Algorithm 2: Merge.
  Input: Two sorted arrays $B[0 \dots p-1]$ and $C[0 \dots q-1]$, and an array $A[0..p+q-1]$.
  Output: The number of inversions involving an element from $B$ and an element from $C$.
  Modifies: $A$.

  $count \leftarrow 0$;
  $i \leftarrow 0$; $j \leftarrow 0$; $k \leftarrow 0$;
  while $i < p$ and $j < q$ do
1     if ... then
          $A[k] \leftarrow B[i]$; $i \leftarrow i + 1$;
      else
          $A[k] \leftarrow C[j]$; $j \leftarrow j + 1$;
2         $count \leftarrow count +$ ...;
      $k \leftarrow k + 1$;
  if $i = p$ then
      copy $C[j \dots q-1]$ to $A[k \dots p+q-1]$;
  else
      copy $B[i \dots p-1]$ to $A[k \dots p+q-1]$;
  return $count$;
```

If you prefer, you can open this figure in a separate tab.

Note that lines 1 and 2 in the function **Merge** are incomplete. Which of the following options completes these two lines so that the algorithm is correct?

Hint: Note that **CountInversions** sorts A as a byproduct of counting the inversions.

○
- Line 1: $B[j] \leq C[i]$
- Line 2: $i - p$

●
- Line 1: $B[i] \leq C[j]$
- Line 2: $p - i$

○
- Line 1: $B[i] \geq C[j]$
- Line 2: $i - p$

○
- Line 1: $B[i] \leq C[j]$
- Line 2: $q - j$

○
- Line 1: $B[i] \geq C[j]$
- Line 2: $p - i$

Question 5

Which of the following gives the recurrence that results in the tightest running time for Algorithm **CountInversions**?

○ $T(n) = 2T(n/2) + O(n^3)$

○ $T(n) = 2T(n/2) + O(1).$

○ $T(n) = 2T(n/2) + O(n^2)$

○ $T(n) = T(n/2) + O(n).$

● $T(n) = 2T(n/2) + O(n)$

Question 6

Which of the following gives the tightest order of growth for the solution of the following recurrence?

- $T(n) = 4T(n/2) + n$
- $T(1) = 1$

The video lectures and slides cover the Master Theorem. If you want access to more material on the subject, you may wish to review the Wikipedia page on the Master Theorem.

● $O(n^2)$

○ $O(n)$

○ $O(n \log n)$

○ $O(\log n)$

○ $O(n^3)$

Question 7

Which of the following gives the tightest order of growth for the solution of the following recurrence?

- $T(n) = 4T(n/2) + n^3$
- $T(1) = 1$

○ $O(n)$

○ $O(\log n)$

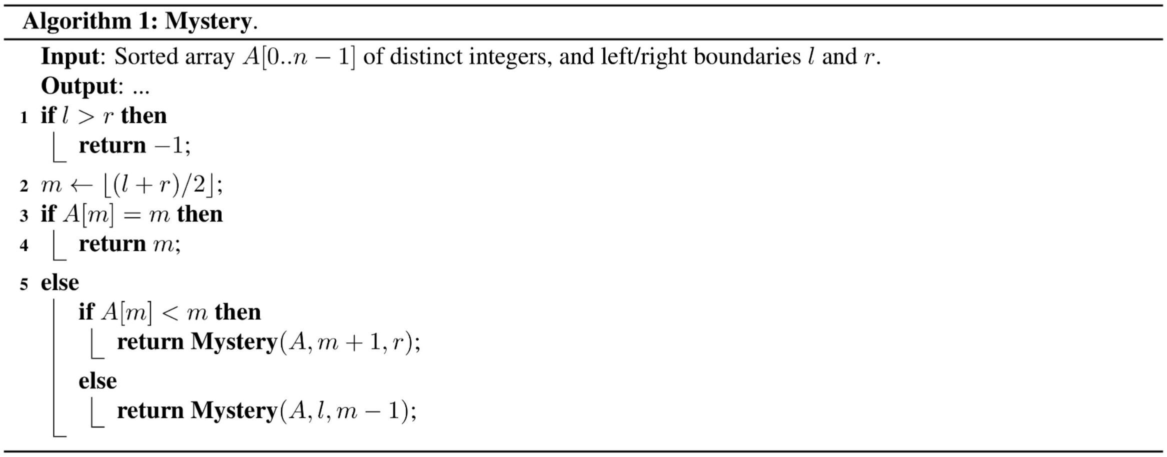
○ $O(n \log n)$

● $O(n^3)$

○ $O(n^2)$

Question 8

In Questions 8-10, we will consider the following mystery algorithm:



```
Algorithm 1: Mystery.
  Input: Sorted array $A[0..n-1]$ of distinct integers, and left/right boundaries $l$ and $r$.
  Output: ...
  if $l > r$ then
      return $-1$;
  $m \leftarrow \lfloor (l+r)/2 \rfloor$;
  if $A[m] = m$ then
      return $m$;
  else
      if $A[m] < m$ then
          return Mystery$(A, m+1, r)$;
      else
          return Mystery$(A, l, m-1)$;
```

If you prefer, you can open this figure in a separate tab.

What does **Mystery**([-2,0,1,3,7,12,15],0,6) compute? Enter your answer as a number in the box below.

3

Question 9

What does Algorithm **Mystery** compute when run on input $(A[0..n-1], 0, n-1)$?

● Returns i if there exists an i such that $A[i] = i$, and -1 otherwise.

○ Returns i if there exists an i such that $A[i] < A[\lfloor (n-1)/2 \rfloor]$, and -1 otherwise.

○ Returns -1, regardless of the content of A.

○ Returns i if there exists an i such that $A[i] > A[\lfloor (n-1)/2 \rfloor]$, and -1 otherwise.

Question 10

What are the best case and worst case running times of Algorithm **Mystery** as a function of the input size n (and assume $l \leq r$ in the input)?

○ Best case: $O(1)$
Worst case: $O(n \log n)$

○ Best case: $O(n)$
Worst case: $O(n \log n)$

○ Best case: $O(n)$
Worst case: $O(\log n)$

● Best case: $O(1)$
Worst case: $O(\log n)$

○ Best case: $O(1)$
Worst case: $O(n)$

Question 11

In Questions 11-14, we consider clusterings of points in preparation for the Project and Application.

Let $S(n,k)$ denote the number of ways that a set of n points can be clustered into k non-empty clusters. Which of the following is a correct recurrence for $S(n,k)$, for $n \geq 1$? Assume, for the base cases, that $S(n,n) = S(n,1) = 1$.

○ $S(n,k) = n\ S(n,k-1)$

○ $S(n,k) = k\ S(n,k-1)$

● $S(n,k) = k\ S(n-1,k) + S(n-1,k-1)$

○ $S(n,k) = k\ S(n-1,k)$

○ $S(n,k) = k\ S(n-1,k-1)$

Question 12

Which of the following formulas gives the number of ways of clustering a set of n points into 2 non-empty clusters; that is, a solution to the recurrence from the previous question for $k = 2$?

● $2^n - 2$

○ $n + 1$

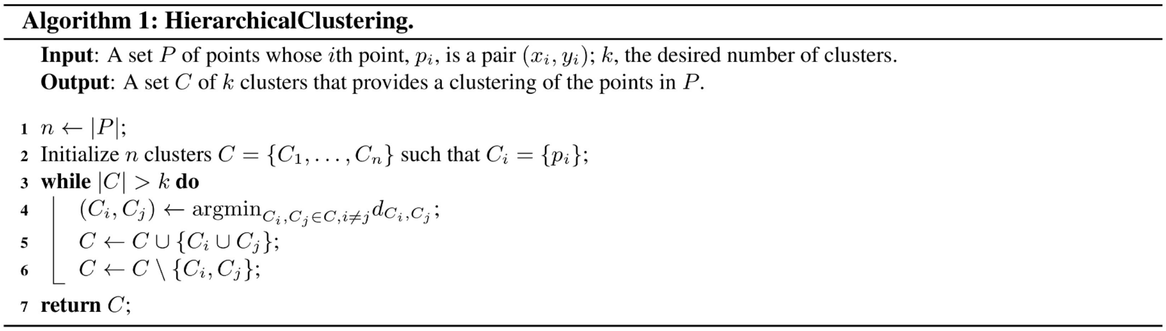
○ $2^{n-1} - 1$

○ $2^n - 1$

○ $n - 1$

Question 13

Consider the following pseudo-code of the hierarchical clustering algorithm:



Algorithm 1: HierarchicalClustering.

Input: A set P of points whose ith point, p_i, is a pair (x_i, y_i); k, the desired number of clusters.
Output: A set C of k clusters that provides a clustering of the points in P.

```
1 $n \leftarrow |P|$;
2 Initialize $n$ clusters $C = \{C_1, \ldots, C_n\}$ such that $C_i = \{p_i\}$;
3 while $|C| > k$ do
4     $(C_i, C_j) \leftarrow \text{argmin}_{C_i, C_j \in C, i \neq j} d_{C_i, C_j}$;
5     $C \leftarrow C \cup \{C_i \cup C_j\}$;
6     $C \leftarrow C \setminus \{C_i, C_j\}$;
7 return $C$;
```

If you prefer, you can open this figure in a separate tab.

Assuming that Line 4 takes $h(n)$ time in each iteration, for some function h, which of the following gives the tightest worst-case running time of the algorithm as a function of the number of points, n, when k is one? Assume that the union and difference of two sets A and B takes $O(|A| + |B|)$ time to compute.

● $O(n^2 + h(n)\, n)$

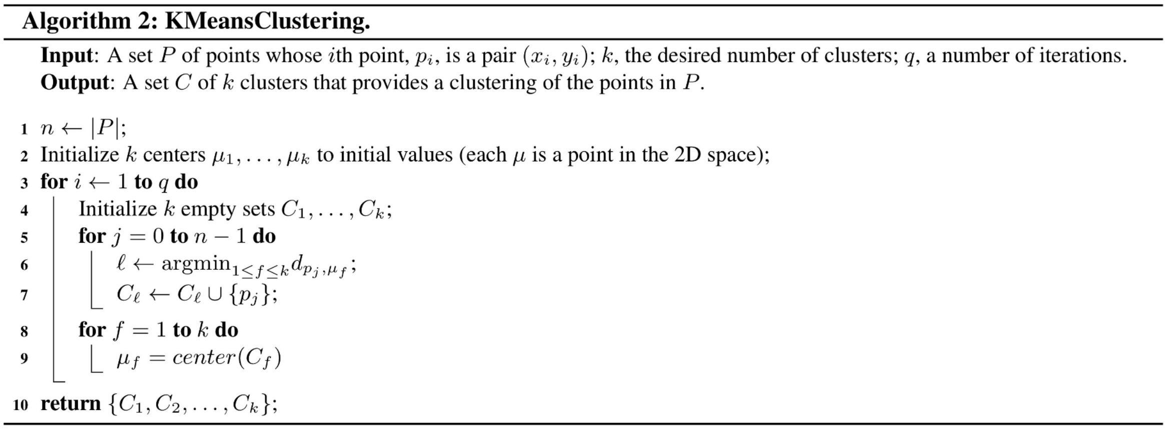
○ $O(n + h(n))$

○ $O(n^3 + h(n)\, n^2)$

○ $O(n \log n)$

Question 14

Consider the following pseudo-code of the k-means clustering algorithm:



```
Algorithm 2: KMeansClustering.
Input: A set $P$ of points whose $i$th point, $p_i$, is a pair $(x_i, y_i)$; $k$, the desired number of clusters; $q$, a number of iterations.
Output: A set $C$ of $k$ clusters that provides a clustering of the points in $P$.

1  $n \leftarrow |P|$;
2  Initialize $k$ centers $\mu_1, \ldots, \mu_k$ to initial values (each $\mu$ is a point in the 2D space);
3  for $i \leftarrow 1$ to $q$ do
4      Initialize $k$ empty sets $C_1, \ldots, C_k$;
5      for $j = 0$ to $n - 1$ do
6          $\ell \leftarrow \text{argmin}_{1 \leq f \leq k} d_{p_j, \mu_f}$;
7          $C_\ell \leftarrow C_\ell \cup \{p_j\}$;
8      for $f = 1$ to $k$ do
9          $\mu_f = center(C_f)$
10 return $\{C_1, C_2, \ldots, C_k\}$;
```

If you prefer, you can open this figure in a separate tab.

Which of the following gives the tightest worst-case running time of the algorithm as a function of the number of points, n, and the number of clusters k, and the number of iterations q? Assume that adding $\{p_j\}$ to C_l in line 7 takes $O(1)$ time.

● $O(q\,k\,n)$

○ $O(q\,k\,n^2)$

○ $O(q\,k\,n \log n)$

○ $O(k\,n)$

Question 15

Consider a list of n numbers sorted in ascending order. Which of the following gives the worst-case running time of the most efficient algorithm for finding a closest pair of numbers in this list?

○ $O(n \log^2 n)$

○ $O(n)$

○ $O(\log^2 n)$

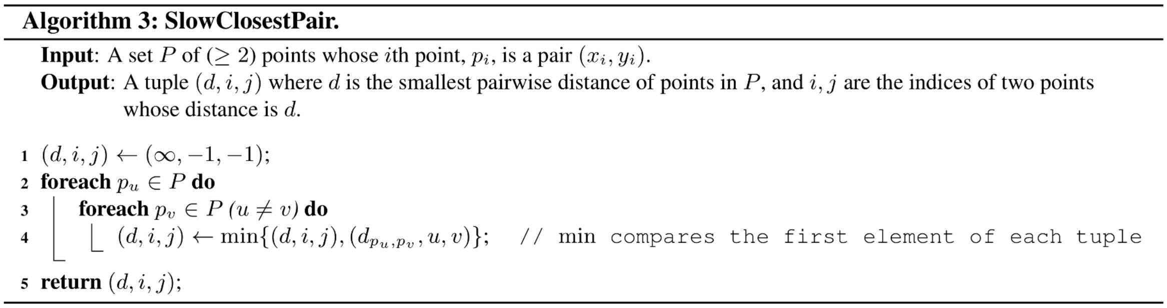
○ $O(n \log n)$

● $O(\log n)$

Question 16

In Question 16-20, we will consider methods for computing a closest pair of 2D points in the plane from a set of n points.

To begin, consider the pseudo-code of the brute-force algorithm for solving the closest pair problem:



Algorithm 3: SlowClosestPair.

Input: A set P of (≥ 2) points whose ith point, p_i, is a pair (x_i, y_i).
Output: A tuple (d, i, j) where d is the smallest pairwise distance of points in P, and i, j are the indices of two points whose distance is d.

```
$(d, i, j) \leftarrow (\infty, -1, -1)$;
foreach $p_u \in P$ do
    foreach $p_v \in P$ $(u \neq v)$ do
        $(d, i, j) \leftarrow \min\{(d, i, j), (d_{p_u, p_v}, u, v)\}$;   // min compares the first element of each tuple
return $(d, i, j)$;
```

If you prefer, you can open this figure in a separate tab.

Which of the following gives the tightest worst-case running time of the algorithm in terms of the number of points n?

○ $O(n^3)$

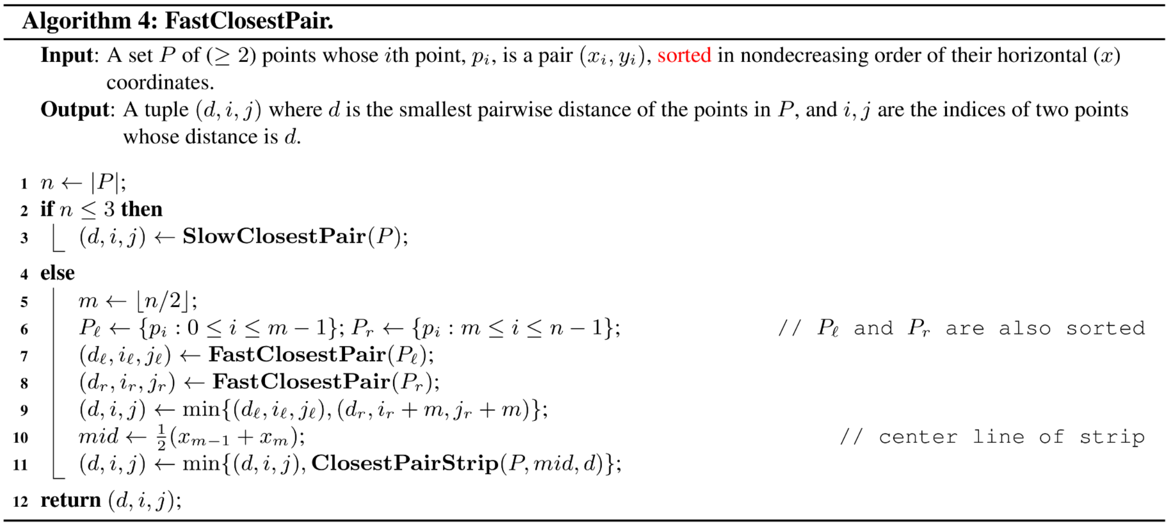
○ $O(n \log n)$

○ $O(n)$

● $O(n^2)$

Question 17

Consider the following pseudo-code of a divide-and-conquer algorithm for solving the closest pair problem:



Algorithm 4: FastClosestPair.

Input: A set P of (≥ 2) points whose ith point, p_i, is a pair (x_i, y_i), sorted in nondecreasing order of their horizontal (x) coordinates.

Output: A tuple (d, i, j) where d is the smallest pairwise distance of the points in P, and i, j are the indices of two points whose distance is d.

```
1  $n \leftarrow |P|$;
2  if $n \leq 3$ then
3      $(d, i, j) \leftarrow$ SlowClosestPair($P$);
4  else
5      $m \leftarrow \lfloor n/2 \rfloor$;
6      $P_\ell \leftarrow \{p_i : 0 \leq i \leq m-1\}$; $P_r \leftarrow \{p_i : m \leq i \leq n-1\}$;          // $P_\ell$ and $P_r$ are also sorted
7      $(d_\ell, i_\ell, j_\ell) \leftarrow$ FastClosestPair($P_\ell$);
8      $(d_r, i_r, j_r) \leftarrow$ FastClosestPair($P_r$);
9      $(d, i, j) \leftarrow \min\{(d_\ell, i_\ell, j_\ell), (d_r, i_r + m, j_r + m)\}$;
10     $mid \leftarrow \frac{1}{2}(x_{m-1} + x_m)$;                                              // center line of strip
11     $(d, i, j) \leftarrow \min\{(d, i, j),$ ClosestPairStrip($P, mid, d$)$\}$;
12 return $(d, i, j)$;
```

If you prefer, you can open this figure in a separate tab.

If the helper function **ClosestPairStrip** used in the conquer step in line 11 has a worst case running time that is $O(f(n))$, which of the following recurrences models the running time of **FastClosestPair**? You may assume that **ClosestPairStrip** examines all of its input and, therefore, $f(n)$ grows at least as fast as n. (Here, d is a constant.)

○
- $T(n) = 4\,T(n/2) + f(n)$
- $T(2) = d$

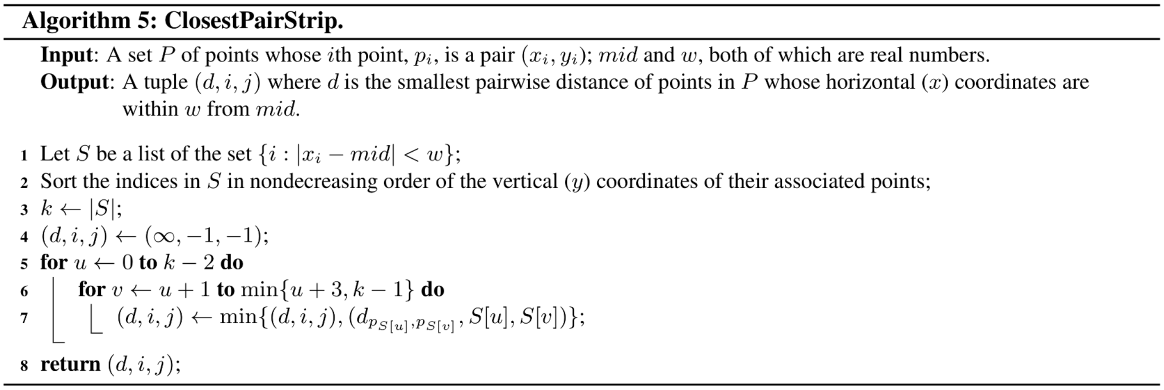
○
- $T(n) = 2\,T(n/2) + f(n)$
- $T(2) = d$

○
- $T(n) = T(n/2) + f(n)$
- $T(2) = d$

●
- $T(n) = 2\,T(n/2) + n$
- $T(2) = d$

Question 18

Consider the following pseudo-code for **ClosestPairStrip**:



Algorithm 5: ClosestPairStrip.

Input: A set P of points whose ith point, p_i, is a pair (x_i, y_i); mid and w, both of which are real numbers.
Output: A tuple (d, i, j) where d is the smallest pairwise distance of points in P whose horizontal (x) coordinates are within w from mid.

```
Let $S$ be a list of the set $\{i : |x_i - mid| < w\}$;
Sort the indices in $S$ in nondecreasing order of the vertical $(y)$ coordinates of their associated points;
$k \leftarrow |S|$;
$(d, i, j) \leftarrow (\infty, -1, -1)$;
for $u \leftarrow 0$ to $k - 2$ do
    for $v \leftarrow u + 1$ to $\min\{u + 3, k - 1\}$ do
        $(d, i, j) \leftarrow \min\{(d, i, j), (d_{p_{S[u]}, p_{S[v]}}, S[u], S[v])\}$;
return $(d, i, j)$;
```

If you prefer, you can open this figure in a separate tab.

If n is the size of the input P, what is the **worst case** running time of **ClosestPairStrip**?

● $O(n \log n)$

○ $O(n)$

○ $O(n^2)$

○ $O(n^2 \log n)$

Question 19

Based on your answers to Questions 17 and 18, what is the worst case running time of **FastClosestPair**?

To answer this question, you will need to review the full version of the Master Theorem that is available here.

● $O(n \log^2 n)$

○ $O(n \log n)$

○ $O(n^2 \log n)$

○ $O(n^2 \log^2 n)$

Question 20

If the algorithm for **ClosestPairStrip** could be modified such that its running time $f(n)$ is $O(n)$, what would be the worst case running time of **FastClosestPair**? Again, use your answer from

Question 17 and consult the Master Theorem.

● $O(n \log n)$

○ $O(n \log^2 n)$

○ $O(n^2 \log n)$

○ $O(n^2 \log^2 n)$

Question 21

Challenge problem

Think about how to modify **FastClosestPair** and **ClosestPairStrip** such that **ClosestPairStrip** runs in $O(n)$ time in the worst case. The key will be to pass two copies of the list of input points, one sorted in horizontal order and one sorted in vertical order, to each call of **FastClosestPair** and **ClosestPairStrip**.

Note that this question is ungraded (counts for zero points) and does not require the input of any type of answer. We suggest that you attempt this challenge only after you have finished the entire Module since this modification is tricky.

□ **In accordance with the Coursera Honor Code, I (Kumar Aakash) certify that the answers here are my own work.**

Submit Answers Save Answers

You cannot submit your work until you agree to the Honor Code. Thanks!