Feedback — Homework 3

Help

You submitted this homework on **Tue 30 Sep 2014 12:04 PM CST**. You got a score of **100.00** out of **100.00**.

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.

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.

You entered:

3

Your Answer		Score	Explanation
3	✔	5.00	Correct.
Total		5.00 / 5.00	

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.

You entered:

0.5*n^2-0.5*n

Preview Help

Your Answer		Score	Explanation
0.5*n^2-0.5*n	✔	5.00	
Total		5.00 / 5.00	

Question Explanation

Consider which class of examples generated the maximal number of inversions and derive a formula for the number of inversions as an arithmetic sum.

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.

Your Answer		Score	Explanation
$O(n^2)$	✔	5.00	Correct. The brute force method compares every element to every other element so it has to be $O(n^2)$.
$O(n^3)$			
$O(n)$			
$O(1)$			
$O(n\log(n))$			
Total		5.00 / 5.00	

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 $A[0 \ldots n-1]$.
Output: The number of inversions in A.

```
if n = 1 then
    return 0;
else
    copy A[0...⌊n/2⌋ − 1] to B[0...⌊n/2⌋ − 1];
    copy A[⌊n/2⌋...n − 1] to C[0...⌈n/2⌉ − 1];
    il ← CountInversions(B);
    ir ← CountInversions(C);
    im ← Merge(B, C, A);
    return il + ir + im;
```

Algorithm 2: Merge.

Input: Two sorted arrays $B[0 \ldots p-1]$ and $C[0 \ldots q-1]$, 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 ← 0;
  i ← 0; j ← 0; k ← 0;
  while i < p and j < q do
1     if ... then
          A[k] ← B[i]; i ← i + 1;
      else
          A[k] ← C[j]; j ← j + 1;
2         count ← count + ...;
      k ← k + 1;
  if i = p then
      copy C[j...q − 1] to A[k...p + q − 1];
  else
      copy B[i...p − 1] to A[k...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.

Your Answer		Score	Explanation
○ • Line 1: $B[i] \geq C[j]$ • Line 2: $q - j$			
○ • Line 1: $B[i] \leq C[j]$ • Line 2: p			
○ • Line 1: $B[i] \geq C[j]$ • Line 2: i			
◉ • Line 1: $B[i] \leq C[j]$ • Line 2: $p - i$	✔	5.00	Correct.

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

Total	5.00 / 5.00

Question Explanation

Note that line 1 should be consistent with sorting A and that line 2 should count the number of inversions that are detected when $C[j]$ is moved to $A[k]$.

Question 5

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

Your Answer		Score	Explanation
$T(n) = 2T(n/2) + O(n)$	✔	5.00	Correct.
$T(n) = 2T(n/2) + O(\log n)$			
$T(n) = 2T(n/2) + O(1).$			
$T(n) = 2T(n/2) + O(n^2)$			
$T(n) = 2T(n/2) + O(n^3)$			
Total		5.00 / 5.00	

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.

Your Answer	Score	Explanation

$O(n^2)$	✔	5.00	Correct.
$O(n)$			
$O(n\log(n))$			
$O(\log(n))$			
$O(n^3)$			
Total		5.00 / 5.00	

Question Explanation

Review the Master theorem if necessary.

Question 7

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

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

Your Answer		Score	Explanation
$O(n)$			
$O(\log(n))$			
$O(n^2)$			
$O(n^3)$	✔	5.00	Correct.
$O(n\log(n))$			
Total		5.00 / 5.00	

Question Explanation

Review the Master theorem if necessary.

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 ← ⌊(l + r)/2⌋;
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.

You entered:

3

Your Answer		Score	Explanation
3	✔	5.00	
Total		5.00 / 5.00	

Question Explanation

You may wish to implement the mystery function in Python.

Question 9

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

Your Answer		Score	Explanation
○ Returns i if there exists an i such that $A[i] < A[\lfloor (n-1)/2 \rfloor]$, and -1 otherwise.			
○ Returns i if there exists an i such that $A[i] > A[\lfloor (n-1)/2 \rfloor]$, and -1 otherwise.			
◉ Returns i if there exists an i such that $A[i] = i$, and -1 otherwise.	✔	5.00	Correct.
○ Returns -1, regardless of the content of A.			
Total		5.00 /	

5.00

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)?

Your Answer		Score	Explanation
Best case: $O(1)$ Worst case: $O(n \log n)$			
Best case: $O(n)$ Worst case: $O(n \log n)$			
Best case: $O(1)$ Worst case: $O(n)$			
Best case: $O(n)$ Worst case: $O(n^2)$			
Best case: $O(1)$ Worst case: $O(\log n)$	✔	5.00	Correct.
Total		5.00 / 5.00	

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 $S(n, n) = S(n, 1) =.1$

Your Answer		Score	Explanation
$S(n, k) = k\ S(n-1, k-1)$			
$S(n, k) = n\ S(n, k-1)$			
$S(n, k) = k\ S(n-1, k) + S(n-1, k-1)$	✔	5.00	Correct.

$S(n, k) = k\, S(n-1, k)$			
$S(n, k) = k\, S(n, k-1)$			
Total		5.00 / 5.00	

Question Explanation

When deriving the recurrence, consider the case when the nth point goes into an existing non-empty cluster or when it creates a new cluster.

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 $k = 2$?

Your Answer		Score	Explanation
$n+1$			
$2^n - 1$			
$2^n - 2$			
$2^{n-1} - 2$			
$2^{n-1} - 1$	✔	5.00	Correct.
Total		5.00 / 5.00	

Question Explanation

Remember that there are 2^n subsets of the n points. Try to think of one cluster as a subset of the n points and the other cluster as the complement of that subset.

Alternatively, consider implementing the recurrence in Python, computing the value of the recurrence for small values of n, and deriving a pattern.

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.

Your Answer		Score	Explanation
$O(n^2 + h(n)\ n)$	✔	5.00	Correct.
$O(n + h(n))$			
$O(n^3 + h(n)\ n^2)$			
$O(n \log(n))$			
Total		5.00 / 5.00	

Question 14

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

Algorithm 2: KMeansClustering.

Input: A set P of points whose ith 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.

$n \leftarrow |P|$;
Initialize k centers $\mu_1, \ldots, \mu_k$ to initial values (each μ is a point in the 2D space);
for $i \leftarrow 1$ **to** q **do**
    Initialize k empty sets $C_1, \ldots, C_k$;
    for $j = 0$ **to** $n - 1$ **do**
        $\ell \leftarrow \text{argmin}_{1 \leq f \leq k} d_{p_j, \mu_f}$;
        $C_\ell \leftarrow C_\ell \cup \{p_j\}$;
    for $f = 1$ **to** k **do**
        $\mu_f = center(C_f)$
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.

Your Answer		Score	Explanation
◉ $O(q\ k\ n)$	✔	5.00	Correct.
○ $O(q\ k\ n^2)$			
○ $O(q\ k\ n \log(n))$			
○ $O(k\ n)$			
Total		5.00 / 5.00	

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?

Your Answer		Score	Explanation
○ $O(n \log(n))$			
○ $O(n \log^2(n))$			
○ $O(n^2)$			
○ $O(\log(n))$			
◉ $O(n)$	✔	5.00	Correct. Scan through the list and compute the minimum difference between consecutive numbers.
Total		5.00 / 5.00	

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: BFClosestPair.

Input: A set P of (≥ 2) points.
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**
    $(d, i, j) \leftarrow \min\{(d, i, j), (d_{p_u, p_v}, u, v)\}$
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?

Your Answer		Score	Explanation
○ $O(n)$			
○ $O(n^3)$			
○ $O(n \log(n))$			
◉ $O(n^2)$	✔	5.00	Correct.
Total		5.00 / 5.00	

Question 17

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

Algorithm 4: SlowDCClosestPair.

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 the points in P, and i, j are the indices of two points whose distance is d.

$n \leftarrow |P|$;
if $n \leq 3$ **then**
  return BFClosestPair(P);
else
  Let H be a sorted list of the points in P in nondecreasing order of their horizontal (x) coordinates;
  $m \leftarrow \lceil n/2 \rceil$; `// the number of points in each half`
  $mid \leftarrow \frac{1}{2}(x_{H[m-1]} + x_{H[m]})$; `// the horizontal coordinate of the vertical dividing line`
  $P_\ell \leftarrow \{H[i] : 0 \leq i \leq m-1\}$; $P_r \leftarrow \{H[i] : m \leq i \leq n-1\}$;
  $(d_\ell, i_\ell, j_\ell) \leftarrow$ **SlowDCClosestPair**(P_ℓ);
  $(d_r, i_r, j_r) \leftarrow$ **SlowDCClosestPair**(P_r);
  $(d, i, j) \leftarrow \min\{(d_\ell, i_\ell, j_\ell), (d_r, i_r, j_r)\}$; `// min is based on the first element of the tuple`
  Let S be a list of the set $\{p_i : |x_i - mid| < d\}$ **sorted** in nondecreasing order of their vertical (y) coordinates;
  Let k be the number of elements in S;
  for $u \leftarrow 0$ **to** $k-2$ **do**
    for $v \leftarrow u+1$ **to** $\min\{u+3, k-1\}$ **do**
      $(d, i, j) \leftarrow \min\{(d, i, j), (d_{S[u],S[v]}, S[u], S[v])\}$;
return (d, i, j);

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

Which of the following is a recurrence of the running time of the algorithm in terms of the size of the input, n? (c and d are constants)

Your Answer		Score	Explanation
• $T(n) = 2\,T(n/2) + n^2$ • $T(2) = d$			
• $T(n) = 2\,T(n/2) + n$ • $T(2) = d$			
• $T(n) = 2\,T(n/2) + c$ • $T(2) = d$			
• $T(n) = T(n/2) + n$ • $T(2) = d$			
• $T(n) = 2\,T(n/2) + n\log(n)$ • $T(2) = d$	✔	5.00	Correct.
Total		5.00 / 5.00	

Question Explanation

Consider the running times of lines 5 and 12 carefully.

Question 18

Which of the following gives the tightest worst-case running time of Algorithm **SlowDCClosestPair**?

Your Answer		Score	Explanation
$O(n\log(n))$			
$O(n^2\log(n))$			
$O(n^2\log^2(n))$			
$O(n\log^2(n))$	✔	5.00	Correct.
Total		5.00 / 5.00	

Question 19

In **SlowDCClosestPair**, each recursive call sorted the points in horizontal and vertical order. One way to improve the performance of this method is to compute horizontal/vertical orderings for the original points once and then pass portions of these orderings to the various recursive calls.

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

Algorithm 5: FastDCClosestPair.

Input: A set P of (≥ 2) points whose ith point, p_i, is a pair (x_i, y_i); two lists H and V such that H contains the indices of the points sorted in nondecreasing order of their horizontal (x) coordinates, and V contains the indices of the points sorted in nondecreasing order of their vertical (y) 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.

Let n be the number of elements in H;
if $n \leq 3$ **then**
    $Q \leftarrow \{p_{H[i]} : 0 \leq i \leq n-1\}$;
    return BFClosestPair(Q);
else
    $m \leftarrow \lceil n/2 \rceil$; `// the number of points in each half`
    $mid \leftarrow \frac{1}{2}(x_{H[m-1]} + x_{H[m]})$; `// the horizontal coordinate of the vertical dividing line`
    $H_\ell \leftarrow H[0..m-1]$; $H_r \leftarrow H[m..n-1]$;
    Copy to V_ℓ, in order, the elements $V[i]$ that are elements of H_ℓ;
    Copy to V_r, in order, the elements $V[i]$ that are elements of H_r;
    $(d_\ell, i_\ell, j_\ell) \leftarrow$ **FastDCClosestPair**(P, H_ℓ, V_ℓ); `// Use the original P`
    $(d_r, i_r, j_r) \leftarrow$ **FastDCClosestPair**(P, H_r, V_r); `// Use the original P`
    $(d, i, j) \leftarrow \min\{(d_\ell, i_\ell, j_\ell), (d_r, i_r, j_r)\}$; `// min is based on the first element of the tuple`
    Copy to S, in order, every $V[i]$ for which $|x_{V[i]} - mid| < d$;
    Let k be the number of elements in S;
    for $u \leftarrow 0$ **to** $k-2$ **do**
        for $v \leftarrow u+1$ **to** $\min\{u+3, k-1\}$ **do**
            $(d, i, j) \leftarrow \min\{(d, i, j), (d_{S[u],S[v]}, S[u], S[v])\}$;
return (d, i, j);

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

Which of the following is a recurrence of the running time of the algorithm in terms of the size of the input, n? (c and d are constants)

Your Answer		Score	Explanation
○ $T(n) = T(n/2) + n$; $T(1) = d$			
◉ $T(n) = 2T(n/2) + n$; $T(1) = d$	✔	5.00	Correct. Both the divide steps and the conquer steps are $O(n)$.
○ $T(n) = 2T(n/2) + n^2$; $T(1) = d$			
○ $T(n) = 2T(n/2) + n\log(n)$; $T(1) = d$			
○ $T(n) = 2T(n/2) + c$; $T(1) = d$			
Total		5.00 /	

5.00

Question 20

Which of the following gives the tightest worst-case running time of Algorithm **FastDCClosestPair**?

Your Answer		Score	Explanation
$O(n^2 \log^2(n))$			
$O(n^2 \log(n))$			
$O(n \log^2(n))$			
$O(n \log(n))$	✔	5.00	Correct.
Total		5.00 / 5.00	