Feedback — Homework 2

Help Center

Thank you. Your submission for this homework was received.

You submitted this homework on **Fri 2 Oct 2015 10:32 AM PDT**. You got a score of **95.00** out of **100.00**.

This Homework covers the basics of algorithmic efficiency and discusses two methods for performing breadth-first search and computing connected components of grapns. We'll build on these topics further in the Project and Application components of Module 2.

Note that you have only two attempts on this and all subsequent Homeworks. So, please check your work carefully.

Please review the notes on algorithmic efficiency and graph exploration and BFS as needed.

Question 1

For Questions 1-3, we will consider the following pseudo-code of Algorithm **Mystery**:

Algorithm 1: Mystery.

Input: Undirected graph $g = (V = \{0, 1, \dots, n-1\}, E)$ given by its adjacency matrix A.
Output: Set X.

$X \leftarrow \emptyset$;
for $i \leftarrow 0$ **to** $n-1$ **do**
    $flag \leftarrow True$;
    for $j \leftarrow 0$ **to** $n-1$ **do**
        if $A[i, j] = 1$ **then**
            $flag \leftarrow False$;
            Break;
    if $flag = True$ **then**
        $X \leftarrow X \cup \{i\}$;
return X;

If you find it easier to refer to, you can open this figure in another window with this link: figure.

Given a graph g, what set X does Algorithm **Mystery** compute?

Your Answer		Score	Explanation
The set of all nodes that have degree 0.	✔	5.00	

The set of all nodes that are connected to every other node.

The set of all nodes that have degree 1.

The set of all nodes.

Total	5.00 / 5.00

Question 2

Which of the following terms gives the tightest possible bound on the best-case running time of Algorithm **Mystery**?

Your Answer		Score	Explanation
$O(1)$			
$O(n^2)$			
$O(n^3)$			
$O(n)$	✔	5.00	Correct. One best case situation would be when the graph is a complete graph with n nodes.
$O(n \log n)$			
Total		5.00 / 5.00	

Question 3

Which of the following terms gives the tightest possible bound on the worst-case running time of Algorithm **Mystery**?

Your	Score	Explanation

Answer			
$O(n^2)$	✔	5.00	Correct. One worst case situation would be when the graph consists of n nodes and no edges.
$O(1)$			
$O(n)$			
$O(n \log n)$			
$O(n^3)$			
Total		5.00 / 5.00	

Question 4

Which of the following choices is the tightest upper bound for the function $f(n) = \frac{1}{2} n(n+1)$?

Your Answer		Score	Explanation
$f(n)$ is $O(n^3)$.			
$f(n)$ is $O(n)$.			
$f(n)$ is $O(n^2)$.	✔	5.00	Correct.
$f(n)$ is $O(1)$.			
Total		5.00 / 5.00	

Question 5

Which of the following choices is the tightest upper bound for the function $f(n) = \frac{1}{2^n}$?

Your Answer		Score	Explanation
$f(n)$ is $O(n^3)$.			

$f(n)$ is $O(n^2)$.			
$f(n)$ is $O(n)$.			
$f(n)$ is $O(1)$.	✔	5.00	Correct.
Total		5.00 / 5.00	

Question 6

Which of the following choices is the tightest upper bound for the function $f(n) = \frac{n^2}{1+n}$?

Your Answer		Score	Explanation
$f(n)$ is $O(1)$.			
$f(n)$ is $O(n^2)$.			
$f(n)$ is $O(n)$.	✔	5.00	Correct.
$f(n)$ is $O(n^3)$.			
Total		5.00 / 5.00	

Question 7

Which of the following choices is the tightest upper bound for the function $f(n) = n^3 - n^2$?

Your Answer		Score	Explanation
$f(n)$ is $O(n^2)$.			
$f(n)$ is $O(n^3)$.	✔	5.00	Correct.
$f(n)$ is $O(n)$.			
$f(n)$ is $O(1)$.			
Total		5.00 / 5.00	

Question 8

Let $f(n) = a_0 + a_1 n + a_2 n^2 + a_3 n^3$. In proving that $f(n)$ is $O(n^3)$, what is the smallest value for the constant c consistent with $n_0 = 1$? (You should assume that the a_i are positive.)

Enter this value c as an expression in terms of the coefficients a_0, a_1, a_2, and a_3 below. You should enter a_0 as a0 and so on. Remember to preview your answer before submitting.

You entered:

a0 + a1 + a2 + a3

Preview Help

Your Answer		Score	Explanation
a0 + a1 + a2 + a3	✔	5.00	
Total		5.00 / 5.00	

Question 9

Let $f(n) = a_0 + a_1 n + a_2 n^2 + a_3 n^3$. In proving that $f(n)$ is $O(n^3)$, what is the smallest value for the constant c consistent with $n_0 = 2$? (You should assume that the a_i are positive.)

Enter this value c as an expression in terms of the coefficients a_0, a_1, a_2, and a_3 below. Again, you should enter a_0 as a0 and so on. Remember to preview your answer before submitting.

You entered:

((1/8)*a0) + ((1/4)*a1) + ((1/2)*a2)+ a3

Preview Help

Your Answer		Score	Explanation
((1/8)*a0) + ((1/4)*a1) + ((1/2)*a2)+ a3	✔	5.00	
Total		5.00 / 5.00	

Question 10

In proving that $3n + 5$ is $\Theta(n)$, what are the two associated constants for the lower and upper bounding functions (largest possible value for the lower bound, smallest possible value for upper bound), assuming that we wish to prove this bound for $n_0 = 1$?

Enter these two constants below (lower bound first, upper bound second), separated by a space. As a hint, both are small integers.

You entered:

3,8

Your Answer		Score	Explanation
3	✔	2.50	
8	✔	2.50	
Total		5.00 / 5.00	

Question 11

In this question, we will make use of the **BFS-Distance** algorithm as we saw it in the lecture. We've also included the pseudo-code for Algorithm **CC-Distance** for computing the connected components of a graph based on **BFS-Distance**. If you find it easier to refer to, you can open this figure in another window with this link: figure.

Algorithm 1: BFS-Distance.

Input: Undirected graph $g = (V, E)$; source node i.
Output: $d_j, \forall j \in V$: the distance between nodes i and j.

Initialize Q to an empty queue;
foreach $j \in V$ **do**
    $d_j \leftarrow \infty$;
$d_i \leftarrow 0$;
$enqueue(Q, i)$;
while Q *is not empty* **do**
    $j \leftarrow dequeue(Q)$;
    foreach *neighbor* h *of* j **do**
        if $d_h = \infty$ **then**
            $d_h \leftarrow d_j + 1$;
            $enqueue(Q, h)$;
return d;

Algorithm 2: CC-Distance.

Input: Undirected graph $g = (V, E)$.
Output: CC: the set of connected components of graph g.

1 $RemainingNodes \leftarrow V$;
2 $CC \leftarrow \emptyset$;
3 **while** $RemainingNodes \neq \emptyset$ **do**
4   Let i be an arbitrary node in $RemainingNodes$;
5     $d \leftarrow$ **BFS-Distance**(....);
6     $W \leftarrow \emptyset$;
7   **foreach** $u \in RemainingNodes$ **do**
8   **if** .... **then**
9             $W \leftarrow W \cup \{u\}$;   // W stores the nodes of a connected component
10     $CC \leftarrow$..........;
11     $RemainingNodes \leftarrow$..........;
12 **return** CC;

The skeleton is missing information on Lines 5, 8, 10, and 11. Which of the following options completes the pseudo-code so that it correctly computes the set of all CCs of a graph?

Your Answer		Score	Explanation
◉ Line 5: g, i Line 8: $d_u \neq \infty$ Line 10: $CC \cup \{W\}$ Line 11: $RemainingNodes - W$	✔	10.00	Correct.
○ Line 5: g, i Line 8: $d_u \neq \infty$ Line 10: $CC \cup W$ Line 11: $RemainingNodes - W$			
○ Line 5: g, i Line 8: $d_u \neq \infty$ Line 10: $CC \cup W$ Line 11: $RemainingNodes - \{i\}$			

- Line 5: g
  Line 8: $d_u \neq \infty$
  Line 10: $CC \cup \{W\}$
  Line 11: $RemainingNodes - \{i\}$

- Line 5: g, i
  Line 8: $d_u \neq \infty$
  Line 10: $CC \cup \{u\}$
  Line 11: $RemainingNodes - W$

Total	10.00 / 10.00

Question Explanation

Check the set operations in your resulting pseudo-code carefully.

Question 12

If a graph g is given by its adjacency list, has n nodes and m edges, which of the following expressions gives the tightest upper bound on the worst-case running time of **CC-Distance** (as given by the pseudo-code in the previous question) on the graph g?

You may assume that the running times of the set operations in **CC-Distance** are proportional to the number of elements being added to or removed from the set. **Note that you should analyze the running time of the pseudo-code as is and not perform any optimizations on it.**

Your Answer		Score	Explanation
$O(m)$			
$O(m+n)$			
$O(n)$			
$O(n^2)$	✔	5.00	Correct. Each call to **BFS-Distance** in line 5 of **CC-Distance** requires time proportional to n plus the number of edges in the connect component. Summing over all calls to **BFS-Distance** yields a running time proportional to n^2 plus the number of edges in the graph

$O(mn^2 + n^3)$

Total	5.00 / 5.00

Question Explanation

Think about the running time of line 5 in **CC-Distance**.

Question 13

Let us now consider a slightly modified breadth-first search, Algorithm **BFS-Visited**, and an algorithm for computing the set of all CCs of a graph based on it, Algorithm **CC-Visited**.

Algorithm 3: BFS-Visited

Input: Undirected graph $g = (V, E)$; source node i.
Output: $Visited$: the set of all nodes visited by the algorithm.

Initialize Q to an empty queue;
$Visited \leftarrow \{i\}$;
$enqueue(Q, i)$;
while Q *is not empty* **do**
    $j \leftarrow dequeue(Q)$;
    foreach *neighbor* h *of* j **do**
        if $h \notin Visited$ **then**
            $Visited \leftarrow Visited \cup \{h\}$;
            $enqueue(Q, h)$;
return $Visited$;

Algorithm 4: CC-Visited.

Input: Undirected graph $g = (V, E)$.
Output: CC: the set of connected components of graph g.

1 $RemainingNodes \leftarrow V$;
2 $CC \leftarrow \emptyset$;
3 **while** $RemainingNodes \neq \emptyset$ **do**
4   Let i be an arbitrary node in $RemainingNodes$;
5     $W \leftarrow$ **BFS-Visited**(....);
6     $CC \leftarrow$..........;
7     $RemainingNodes \leftarrow$..........;
8 **return** CC;

If you find it easier to refer to, you can open this figure in another window with this link: figure.

The skeleton of Algorithm **CC-Visited** is missing information on Lines 5, 6, and 7. Which of the following options completes the pseudo-code so that it correctly computes the set of all CCs of a graph?

Your Answer		Score	Explanation
Line 5: g, i Line 6: $CC \cup W$ Line 7: $RemainingNodes - \{W\}$			
Line 5: $g, RemainingNodes$ Line 6: $CC \cup RemainingNodes$ Line 7: $RemainingNodes - W$			
Line 5: g Line 6: $CC \cup d$ Line 7: $RemainingNodes - d$			
Line 5: g, i Line 6: $CC \cup \{W\}$ Line 7: $RemainingNodes - \{W\}$			
Line 5: g, i Line 6: $CC \cup \{W\}$ Line 7: $RemainingNodes - W$	✔	10.00	Correct.
Total		10.00 / 10.00	

Question Explanation

Remember to double check that your set operations are well-defined.

Question 14

If a graph g is given by its adjacency list, has n nodes and m edges, which of the following terms gives the tightest upper bound on the worst-case running time of **CC-Visited** (as given by the pseudo-code in the previous question) on the graph g?

You may assume that the running times of the set operations in **BFS-Visited** and **CC-Visited** are proportional to the number of elements being added to or removed from the set. **Note that you should analyze the running time of the pseudo-code as is and not perform any optimizations on it.**

Your Answer		Score	Explanation
$O(n)$	✖	0.00	Incorrect. All m of the edges must be examined.

- $O(m)$
- $O(m+n)$
- $O(n^2)$
- $O(mn^2+n^3)$

Total	0.00 / 5.00

Question Explanation

Think about the running time of line 5 in **CC-Visited**.

Question 15

Which of the following statements is true concerning the running times for Algorithm **CC-Distance** and Algorithm **CC-Visited**?

Your Answer		Score	Explanation
The running time of **CC-Distance** is asymptotically slower than that of **CC-Visited** due to the initialization of d in lines 2-3 of **BFS-Distance**.	✔	5.00	Correct. Initializing the dictionary requires $O(n)$ work as given in the pseudo-code.
The running times of **CC-Distance** and **CC-Visited** can't be compared since they return different types of answers.			
The running time of **CC-Visited** is asymptotically slower than that of **CC-Distance** due to the use of a set in line 8 of **BFS-Visited**.			
The running times of **CC-Distance** and **CC-Visited** are asymptotically the same.			
Total		5.00 / 5.00	

Question 16

For the last three questions, we use powers of adjacency matrices to compute some simple properties of paths in graphs. If your linear algebra background needs supplementing, we suggest that you review the class notes on matrices and then review this handout on paths and matrices in graphs.

If A is the adjacency matrix for a graph g, what is the number of paths of length k from node i to node j in g? Note that this answer should include simple paths (no cycles allowed) and non-simple paths (cycles allowed).

Your Answer		Score	Explanation
$(A^k)[i,j]$	✔	5.00	Correct. We will also write this expression as $A^k[i,j]$.
$(A[i,j])^k$			
$(A^{i+j})[k,k]$			
$A[i+k, j-k]$			
Total		5.00 / 5.00	

Question Explanation

Remember to review the handouts linked in the question.

Question 17

If A is the adjacency matrix for a graph g, what is the length of the shortest path from node i to node j in g?

Your Answer		Score	Explanation
The length of the shortest path is the smallest non-negative integer k such that $(A^k)[i,j]$ is non-zero.	✔	5.00	Correct. All powers less than k have zero paths from node i to node j.

- The length of the shortest path is $A[i, j]$.
- The length of the shortest path is the smallest non-negative integer k such that $(A[i, j])^k$ is non-zero.
- The length of the shortest path is the largest non-negative integer k such that $(A^k)[i, j]$ is zero.

Total	5.00 / 5.00

Question 18

One measure of the "importance" of an edge e in a graph g is its *edge centrality*, which is defined as the number of shortest paths in g that include e. Edges with high centrality are usual good targets for removal when one is seeking to split a graph into several connected components by removing an edge.

If A is the adjacency matrix for g and $\hat{A}$ is the adjacency matrix for the graph $\hat{g}$ created by removing e (but not its endpoints) from g, which of the following expressions corresponds to the number of shortest paths from node i to node j that include the edge e? You may assume that there exists at least one path from node i to node j in g.

Your Answer		Score	Explanation
$(A^k)[i, j] - (\hat{A}^k)[i, j]$ where k is the length of the shortest path from node i to node j in g.	✔	5.00	Correct. $(\hat{A}^k)[i, j]$ corresponds to those shortest paths of length k that don't include e.
$(A^k)[i, j]$ where k is the length of the shortest path from node i to node j in g.			
$(\hat{A}^k)[i, j]$ where k is the length of the shortest path from node i to node j in g.			

$(A^k)[i, j] - (\hat{A}^k)[i, j]$ where k is the length of the shortest path from node i to node j in $\hat{g}$.

Total	5.00 / 5.00