

CS 141, Winter 2005

Practice Midterm

Name (first last): SOLUTIONS

- This exam is open book, open notes.
- No electronic equipment allowed (cell phones, PDA's, computers...).
- Write legibly. What can't be read will not be graded.
- Use pseudo-code or English to describe your algorithms.
- When designing an algorithm, you are allowed to use any algorithm or data structure we have covered in cs141 or in cs14 without giving its details, unless the question specifically asks for such details.
- *You may open the exam and start working at the start of class (11:10am) at the start of class. You must stop working on your exam and turn it in the end of class (12:30pm).*
- If you have a question about the meaning of a question, please raise your hand.

for grader use only:

1.	<input type="text"/>	/30
3.	<input type="text"/>	/10
5.	<input type="text"/>	/10
7.	<input type="text"/>	/10

2.	<input type="text"/>	/10
4.	<input type="text"/>	/10
6.	<input type="text"/>	/10

total:

<input type="text"/>	/90
----------------------	-----

1. +3 points for each correct answer, -3 points for each incorrect answer, 0 points for each question not answered.

If the worst-case running time of an algorithm is $O(n^2)$, then the algorithm can finish in time $O(n)$ on some inputs of size n . _____ True False

If the worst-case running time of an algorithm is $O(n)$, then the algorithm can finish in time $\Omega(n^2)$ on some inputs of size n . _____ True False

$1 + 2 + 3 + \dots + (n - 1) + n = O(n)$ _____ True False

$1^{10} + 2^{10} + 3^{10} + \dots + (n - 1)^{10} + n^{10} = \Theta(n^{11})$ _____ True False

If $\alpha = 1.001$, then $\alpha^0 + \alpha^1 + \alpha^2 + \dots + \alpha^n = \Omega(2^n)$ _____ True False

Suppose $T(0) = 0$ and $T(n) = n + T(n/2)$ for $n > 0$. Then $T(n) = O(n)$. _____ True False

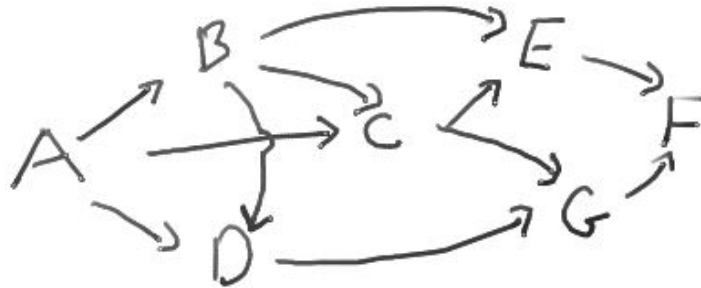
Suppose $T(0) = 1$ and $T(n) = n + 2T(n/2)$ for $n > 0$. The recursion tree for T has depth $O(\log n)$. _____ True False

Suppose $T(0) = 1$ and $T(n) = n + 2T(n - 1)$ for $n > 0$. The recursion tree for T has $O(n^2)$ nodes. _____ True False

To determine whether an algorithm is correct, it is enough to try it on a few thousand inputs and see if it works on those inputs. _____ True False

Let $\text{fib}(n)$ denote the n th Fibonacci number. For all n , the greatest common divisor of $\text{fib}(n)$ and $\text{fib}(n - 1)$ equals 1. (Hint: recall the lower bound on the running time of Euclid's algorithm.) _____ True False

3. In the directed graph below, label each vertex with the number of distinct paths from A to the vertex.



B=1 E=3
 A=1 C=2 F=7
 D=2 G=4

How many distinct paths are there that go from A to F *and* go through C on the way? Give a value and explain how you got it.

There are 4. There are two paths from A to C, and two paths from C to F. Each path from A to F through C consists of a path from A to C then a path from C to F. Thus, the number of such paths is the product $2 \cdot 2$.

4. Let $T(n)$ be the *time taken* by the following subroutine:

```

int mystery3(int n) {
    if (n <= 1) return 2;
    else return (2*mystery3(n-1) + 3*mystery3(n-2)) % 10;
}
  
```

Give a recurrence relation for $T(n)$ (ignoring constant factors).

$T(n) = T(n-1) + T(n-2) + 1$ for $n \geq 1$. $T(0) = T(1) = 1$.

What is the depth of the deepest node in the recursion tree (in terms of n)?

$n-t = O(n)$

Precisely describe an algorithm that runs in $O(n)$ time and returns the same *value* as `mystery3(n)` returns.

```

int mystery3(int n) {
    if (n <= 1) return 2;
    Array<int> A;
  
```

```

A[0] = 2;
A[1] = 2;
for (int i = 2; i <= n; ++i)
    A[i] = (2*A[i-1] + 3*A[n-2]) % 10;
return A[n];
}

```

5. Prove or disprove the following claim:

For any digraph G and any edge (u, w) on a cycle in G , it is *always* possible to run DFS on G in such a way that (u, w) is classified as a back edge.

The claim is true. If the DFS starts at w , u will be a descendant of w in the DFS tree, so the edge (w, u) will have to be a back edge.

6. Describe a linear-time algorithm for the following problem:

Given a digraph G with no cycles, and two vertices s and t , count the number of even-length paths from s to t .

(An “even-length” path is one with an even number of edges.)

For each vertex v ,

Define $EVEN[v] = \#$ even-length paths from s to v .

Define $ODD[v] = \#$ odd-length paths from s to v .

These satisfy the recurrences

$$EVEN[s] = 1, ODD[s] = 0$$

$$EVEN[v] = \sum_{w:(w,v) \in E} ODD[w]$$

$$ODD[v] = \sum_{w:(w,v) \in E} EVEN[w]$$

Here is the algorithm:

1. Set $EVEN[s] = 1$, $ODD[s] = 0$
2. Initialize $EVEN[v] = ODD[v] = 0$ for all other vertices v .
3. Topologically sort the vertices using DFS.
4. Consider the vertices in topological order.
5. For each vertex v other than s ,
6. Set $EVEN[v]$ and $ODD[v]$ according to the recurrences above.
7. Return $EVEN[t]$

7. Run the dynamic programming algorithm for longest ascending subsequence on the sequence below. Fill in the empty box below each number with the value of the subproblem for that number.

3	2	5	3	4	7	1	10	8	9
1	1	2	2	3	4	1	5	5	6

What is the longest ascending subsequence?

2 3 4 7 8 9

Give pseudo-code for the dynamic programming algorithm.

(Assuming the input is in an array $A[1..n]$)

```

For i = 1 to n
  Set table[i] = 1
  For j = 1 to i-1
    if  $A[j] < A[i]$  and  $table[j]+1 > table[i]$ 
      set  $table[i] = table[j]+1$ 
return  $\max(table[1], table[2], \dots, table[n])$ 

```

What is the running time as a function of n , the length of the sequence?

Proportional to n^2 .