Name: Andrew Black

CS 350 Practice Midterm Examination

Time: One hour and fifty minutes
Closed Book — one 8.5 × 11 double-sided crib-sheet is allowed.

21^{st} October 2015

Instructions

This is a closed book exam. All you need is a writing instrument, an eraser, and your brain. One "crib sheet" up to 8.5″ × 11″ (double-sided) is allowed. Please take cell phones, PDAs, calculators, books, and computers off of your desk.

Write your answers on the exam paper. You can use the back of this sheet for scratch space. A supply of plain paper is available for scratch work; please hand in your scratch paper and your crib sheet with your exam.

There are **five** questions —attempt them all. If you get stuck, go on to the next question; mark the unfinished question so that you can come back to it if you have time. Keep on iterating through the paper until you have attempted all of the questions. In case you don't have enough time to answer all of the questions, I advise you to spend the first few minutes reading through the whole paper, and strategizing where to spend your time. The number of points for each question is given.

DO NOT TAKE ANY WRITTEN MATERIALS WITH YOU OUT OF THE EXAMINATION ROOM. This is because some students may be taking the exam at other times. I am relying on your academic integrity; do not discuss the examination questions with them.

Do not turn this page until you are instructed to do so.

Q1	Q2	Q3	Q4	Q5	Total
14	16	24	18	30	108

This page is intentionally blank

Question 1. [14 pts.; 2 pts each]

Circle **true** or **false**. Justify your answer briefly.

2 points

a. $n! \in O(2^n)$.

true false

$$\lim_{n\to\infty} \frac{n!}{2^n} = \frac{\overbrace{1\cdot 2\cdot 3\cdot \cdots\cdot n}^{n \text{ terms}}}{\underbrace{2\cdot 2\cdot 2\cdot \cdots\cdot 2}_{n \text{ terms}}} \to \infty$$

2 points

b. $n! \in \Omega(2^n)$.

true false

Some reasoning as above

2 points

c. $n! \in \Theta\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right)$

true false

rhs is Sterling's formula

2 points

d. Euclid's algorithm for finding the greatest common divisor of two numbers uses the decrease-by-a-constant-and-conquer algorithm design technique.

true false

decrease is by variable amount

2 points

e. $n^n \in \Omega(n!)$.

true false

2 points

f. $\lg n^2 \in \Theta(\lg n)$.

true false

$\lg n^2 = 2 \lg n \in \Theta(2\lg n) = \Theta(\lg n)$

2 points

g. $e^n \in \Theta(2^n)$.

true false

the functions differ by a factor of $\left(\frac{e}{2}\right)^n$, which is not a constant

Question 2. [Exponentiation; 22 pts.]

A programmer develops the following algorithm for computing a^n, using the formula

$$a^n = a^{\lfloor n/2 \rfloor} \times a^{\lceil n/2 \rceil}$$

function POWER(a, n)
 || Computes a^n by a divide-and-conquer algorithm
 || **Input**: a number a, and a positive integer n
 || **Output**: a^n
 if $n = 1$ **then**
  return a
 else
  return POWER(a, $\lfloor n/2 \rfloor$) × POWER(a, $\lceil n/2 \rceil$)

1. Write a recurrence relation for $M(n)$, the number of multiplications that occur when applying this algorithm to a and n.

3 points

Recurrence Relation: $M(n) = \begin{cases} M(\lfloor \frac{n}{2} \rfloor) + M(\lceil \frac{n}{2} \rceil) + 1 & n > 1 \\ 0 & n = 1 \end{cases}$

2. Solve the recurrence— you may assume that $n = 2^k$ for some $k \geq 0$.

3 points

$M(2^k) = 2M(2^{k-1}) + 1$
$= 2[2M(2^{k-2}) + 1] + 1 = 4M(2^{k-2}) + 2 + 1$
$= 4[2M(2^{k-3}) + 1] + 2 + 1 = 8M(2^{k-3}) + 2^2 + 2^1 +$
$= 2^i M(2^{k-i}) + \sum_{j=0}^{i-1} 2^j$ put $i = k$
$= 2^k M(2^0) + \sum_{j=0}^{k-1} 2^j = 0 + 2^k - 1 = n - 1$

5 points

3. What is the asymptotic efficiency class of this algorithm?

Θ efficiency of algorithm: $\Theta(n)$

4. Can you see a way to improve the asymptotic efficiency of this pseudocode? If so, write the improved pseudocode (which should work for all $n > 0$, not just for $n = 2^k$), and calculate the asymptotic efficiency of your improved algorithm. If not, explain why the pseudocode cannot be improved.

change the else branch to

if n.isEven then ~~return~~ def semi ← POWER(a, $\lfloor \frac{n}{2} \rfloor$)
 return semi × semi
else
 return semi × semi × a

5 points

efficiency so now is $\Theta(\lg n)$ because recursion tree is depth $\lg n$, and there are $\Theta(1)$ multiplications at each le

Question 3. [Basic Algorithms; 24 pts]

For each of the following pieces of pseudocode, name the problem that it solves, and name the algorithm used to solve the problem. If you do not know the name of the algorithm, give a name that you think captures the idea of the algorithm. Write down the basic operation and the cost equation (either a summation or a recurrence) for the running time of the algorithm. Solve the cost equation (show your steps) and write down the worst case asymptotic complexity of the algorithm. Finally, list the algorithmic paradigm to which the algorithm belongs, e.g, Brute force, decrease by a constant. Put your answers in the boxes provided. You may wish to work out the answers on scrap paper first.

a. Unknown1

function UNKNOWN1(*Point* $P[1..n]$)
|| Input P is an array of $n \geq 2$ points, each with an x and y component.
$d \leftarrow \infty$
for $i \leftarrow 1$ **to** $n-1$ **do**
  for $j \leftarrow i+1$ **to** n **do**
    $d \leftarrow min(d, (P[i].x - P[j].x)^2 + (P[i].y - P[j].y)^2)$
return $sqrt(d)$

Problem name	Closest Pair
Algorithm name	Brute Force Closest Pair
Basic Operation	computing the sq of dist. between 2 points
Running time cost equation	$T(n) \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1$
Solution of cost equation	$T(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1 = \sum_{i=1}^{n-1} n-i$ $= \frac{(n-1)(n)}{2}$
Worst case asymptotic complexity	$\Theta(n^2)$
Algorithmic Paradigm	Brute Force

8 points

b. Unknown2

$\boldsymbol{ALGORITHM}$: $unknown2(integer: A[0..n-1], integer: K)$

$l \leftarrow 0;\ \ r \leftarrow n-1$

$\boldsymbol{while}\ l \le r\ \ \mathbf{do}\ \{$

$\quad m \leftarrow \left\lfloor \frac{(l+r)}{2} \right\rfloor$

$\quad \boldsymbol{if}\ K = A[m]\ \ \boldsymbol{return}\ m$

$\quad \boldsymbol{else\ if}\ K < A[m]\ \ r \leftarrow m-1$

$\quad \boldsymbol{else}\ l \leftarrow m+1$

$\quad \}$

$\boldsymbol{return} - 1$

Problem name	Search in a sorted array
Algorithm name	Binary Search
Basic Operation	key comparison
Running time cost equation	$T(n) = \sum_{i=1}^{\lg n} 2$
Solution of cost equation	$T(n) = \sum_{i=1}^{\lg n} 2$ $= 2\lg n$
Worst case asymptotic complexity	$\Theta(\lg n)$
Algorithmic Paradigm	Divide & conquer or

Decrease by a constant factor (2) and conquer

8 points

c. Unknown3

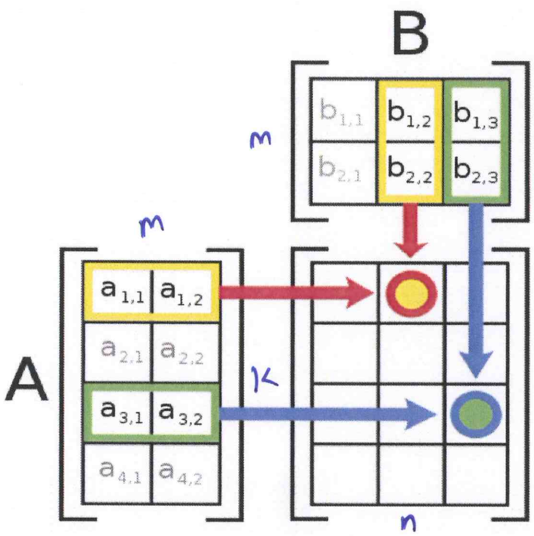
$\boldsymbol{ALGORITHM}$: $unknown3(integer: A[0..n-1])$
$\quad$ $\boldsymbol{for}\ i \leftarrow 0\ to\ n-2\ \boldsymbol{do}$ {
$\quad\quad$ $min \leftarrow i$
$\quad\quad$ $\boldsymbol{for}\ j \leftarrow i+1\ to\ n-1\ \boldsymbol{do}$ {
$\quad\quad\quad$ $if\ A[j] < A[min]\quad min \leftarrow j$
$\quad\quad\quad$ }
$\quad\quad$ $swap\ A[i] and\ A[min]$
$\quad\quad$ }

Problem name	Sorting an Array
Algorithm name	Selection sort
Basic Operation	key comparison
Running time cost equation	$T(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$
Solution of cost equation	$T(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$ $= \sum_{i=0}^{n-2} n-1-i$ $= (n-1)(n-1) - \sum_{i=0}^{n-2} i$ $= n^2 - 2n + 1 - \left(\frac{(n-2)(n-1)}{2} \right)$ $= n^2 - 2n + 1 - \frac{n^2 - 3n + 2}{2}$ $= \frac{n^2}{2} + \frac{n}{2}$
Worst case asymptotic complexity	$\Theta(n^2)$
Algorithmic Paradigm	Brute force or Decrease-by-one

8 points

Question 4. [18 pts.]

The figure below (from Wikipedia) illustrates the product of two matrices A and B, showing how each element of the product matrix corresponds to a row of A and a column of B. Let the dimensions of matrix A be $k \times m$ and the dimensions of matrix B be $m \times n$. Then the product matrix AB has size $k \times n$ and consists of all combinations of dot products of rows of A and columns of B.



For example, the values in the product matrix at the intersections marked with circles are:

$$
\begin{aligned}
x_{1,2} &= (a_{1,1} \quad a_{1,2}) \cdot (b_{1,2} \quad b_{2,2}) \\
&= a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\
x_{3,3} &= (a_{3,1} \quad a_{3,2}) \cdot (b_{1,3} \quad b_{2,3}) \\
&= a_{3,1}b_{1,3} + a_{3,2}b_{2,3}.
\end{aligned}
$$

(a) Write down, in pseudocode, an algorithm based on this definition that computes the product matrix.

```
MATMULT (A[1..k, 1..m], B[1..m, 1..n])
    for i ← 1..k do:
        for j ← 1..n do:
            c[i,j] ← DOTPRODUCT (A[i,*], B[*,j])

return c


DOTPRODUCT (V1 [1..m], V2 [1..m])
    result ← V1[1] × V2[1]
    for i ← 2..n do:
        result ← result + V1[i] × V2[i]
return result
```

6 points

(b) Analyze the complexity of the algorithm in terms of the number of *dot product* operations.

$$D(k, m, n) = \sum_{i=1}^{k} \sum_{j=1}^{n} 1$$

$$= \sum_{i=1}^{k} n$$

$$= kn$$

6 points

(c) Analyze the complexity of the algorithm in terms of the number of *scalar multiplication* operations.

Each dot product is between vectors of length m, and so involves m multiplications (and $(m-1)$ additions)

Hence $M(k, m, n) = m \cdot D(k, m, n)$

$$= mkn$$

6 points

Question 5. [25 pts.]

The median of a list of k numbers is defined to be the $\lceil (k/2) \rceil^{\text{th}}$ number in the list. Given two arrays $L_1, \dots L_n$ and $R_1, \dots R_n$, each of which is **sorted** in increasing order:

(a) design an algorithm that will find the median of the combined $2n$ elements in *sublinear* time. Hence, any method that looks at $O(n)$ elements is too slow!.

Median (L [1..n], R[1..n])

if n = 1 return min(L[1], R[1])

mid = $\lceil \frac{n}{2} \rceil$

M_L = L [mid]

M_R = R [mid]

if $M_L = M_R$ then return M_L

elseif $M_L > M_R$ then return median (L[1.. mid], R [n- mid -1 .. n])

elseif $M_L < M_R$ then return median (L [n- mid-1 .. n], R[1.. mid])

How did I solve this? By doing part (b) first!! It often helps to start with an example

10 points

(b) describe how your algorithm would work on the following input:
$L = 3, 6, 8, 12, 27, 40, 45$ and $R = 2, 15, 19, 25, 31, 37, 59$

compare the medians $M(L) = 12$
$M(R) = 25$

$M(L) < M(R)$, so discard upper part of $M(R)$ (3 elements) and lower part of $M(L)$ (3 elements)

$L = 12, 27, 40, 45$ $\quad R = \cancel{25\ 31\ 37\ 59}$ $2\ 15\ 19\ 25$

Because we have discarded 3 elements from each end, we need the median of the remaining lists — Recurse

$27 > 15$ — find median of $[\cancel{40}\ 12, \cancel{45}\ 27]$ and $[19\ 25]$

5 points

$\cancel{40}\ 12 < \cancel{x}\ 19$ — find median of $[27\cancel{40}]$ $[19]$

median of $n=2$ elements is $\lceil \frac{n}{2} \rceil = 1^{st}$, hence 19

(c) give an informal justification of your algorithm's correctness, and

Let M_L and M_R be the medians of L and R
The median of the combined list cannot be
$>$ ~~the higher~~ $\max(M_L, M_R)$ or $< \min(M_L, M_R)$
why? Because 1/2 the elements have to be > the median and 1/2 < the median, and there aren't enough elements > max or < min.

5 points

At each recursion, we must discard the same number of elements from each list, to keep the property that the median of the reduced list is the median of the original list.

(d) analyze its complexity.

number of comparisons, c, is 1 per recursion,

There are $\theta(\lg n)$ recursions, because we halve the size of n on each (± 1)

In the case where $n = 1$, there is 1 comparison

So complexity is $\theta(\lg n)$

10 points