Computer Science Foundation Exam

August 14, 2015

Section I A

COMPUTER SCIENCE

NO books, notes, or calculators may be used, and you must work entirely on your own.

SOLUTION

Question #	Max Pts	Category	Passing	Score
1	10	DSN	7	
2	10	ANL	7	
3	10	ALG	7	
4	10	ALG	7	
5	10	ALG	7	
TOTAL	50			

You must do all 5 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat.

1) (10 pts) DSN (Recursive Functions)

Consider the problem of transforming a positive integer X into a positive integer Y when the only two operations you are allowed are adding 1 to the current number or multiplying the current number by 2. Write a recursive function that returns the minimum number of steps necessary to transform X to Y. If X > Y, return 1000000000, to indicate that no solution exists. (For example, if X = 13 and Y = 28, the correct response would be 2 - first you would add 1 to 13 to obtain 14, then multiply 14 by 2 to obtain 28.) Feel free to call the provided function. *Note: don't worry about the run time of your function - assume that the inputs are such that the run time is relatively small, even when written using straight-forward recursion. There is a clever, efficient solution without recursion but please write the slower recursive solution since the goal of this question is to test recursive thinking.*

```
#define NO_SOLUTION 1000000000

int min(int x, int y) {
    if (x < y) return x;
    return y;
}

// Returns the minimum number of steps to transform x into y, or
// 100000000 to indicate no solution.
int minSteps(int x, int y) {

    if (x > y) return NO_SOLUTION;          // 2 pts
    if (x == y) return 0;                   // 2 pts

    int mult = 1 + minSteps(2*x, y);        // 2 pts
    int add = 1 + minSteps(x+1, y);         // 2 pts

    return min(add, mult);                  // 2 pts
}
```

2) (10 pts) ANL (Summations and Algorithm Analysis)

Consider the following segment of code, assuming that n has been previously declared and initialized to some positive value:

```
int i, j, k;
for (i = 1; i <= n; i++){
    for(k =1; k <= i; k++){
        j = k;
        while(j > 0)
            j--;
    }
}
```

(a) (3 pts) Write a summation (3 nested sums) equal to the number of times the statement `j--;` executes, in terms of n.

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

Grading: 1/2 pt for outer sum, 1 pt for each inner sum, 1/2 for 1 inside, round down. Note - variable names used in sums are independent of those in code…

(b) (7 pts) Determine a closed form solution for the summation above in terms of n.

$$\sum_{i=1}^{n}\sum_{k=1}^{i}\sum_{j=1}^{k}1=\sum_{i=1}^{n}\sum_{k=1}^{i}k=\sum_{i=1}^{n}\frac{i(i+1)}{2}$$
$$=\frac{1}{2}(\sum_{i=1}^{n}i^2+\sum_{i=1}^{n}i)$$
$$=\frac{1}{2}(\frac{n(n+1)(2n+1)}{6}+\frac{n(n+1)}{2})$$
$$=\frac{n(n+1)}{12}((2n+1)+3)$$
$$=\frac{n(n+1)}{12}(2n+4)$$
$$=\frac{n(n+1)(n+2)}{6}$$

Grading: 1 pt solving inner sum, 1 pt solving middle sum, 2 pts for i^2 sum, 1 pt for i sum, 2 pts for algebra - accept either factored or poly form)

3) (10 pts) ALG (Stacks)

Use a stack to evaluate the postfix expression below. Please show the state of the stack at the exact point in time during the algorithm that the marked (A, B, C) locations are reached while processing the expression. Also, write down the equivalent infix expression, placing parentheses when necessary.

				A					**B**					**C**		
5	**8**	**3**	*****		**+**	**6**	**+**	**7**		**/**	**4**	**2**	**–**		*****	

24	7	2
5	35	5
A	B	C

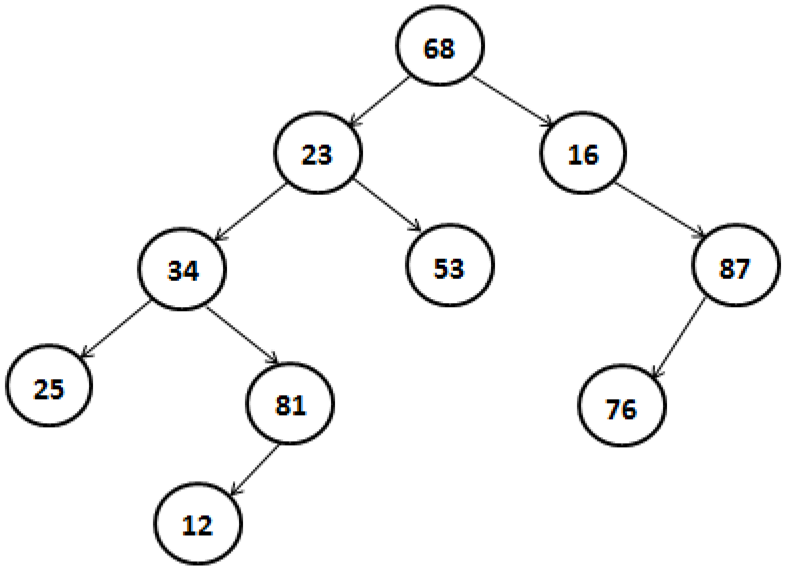
Value of the Expression: 10

Equivalent Infix Expression: (5 + (8 * 3) + 6) / 7 * (4 – 2)

Grading: 2 pts per stack, 1 pt expression, 3 pts expression - extra parens allowed, assign partial as needed.

4) (10 pts) ALG (Binary Trees)

Please give the preorder, inorder and postorder traversal of the binary tree shown below. In addition, determine whether or not the tree is a valid binary search tree.



Preorder:

Preorder: **68, 23, 34, 25, 81, 12, 53, 16, 87, 76**

Inorder: **25, 34, 12 81, 23, 53, 68, 16, 76, 87**

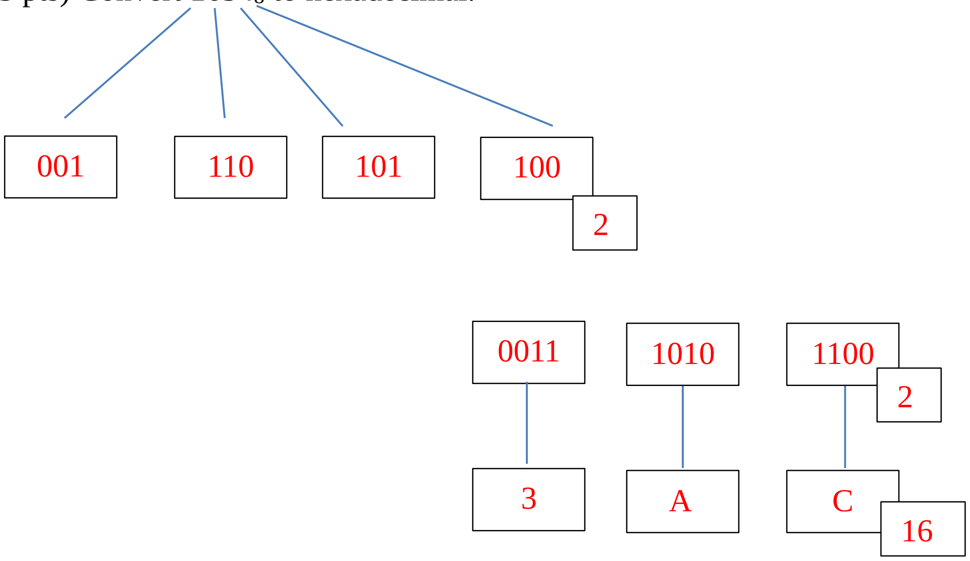
Postorder: **25, 12, 81, 34, 53, 23, 76, 87, 16, 68**

Is this valid binary search tree? (Circle your answer.) YES **NO**

Grading: 3 pts each traversal - give a total of 5 out of 9 if the traversal names are switched but the answers are the three distinct traversals. Otherwise, give partial as follows - 2 pts if 6-9 items correct, 1 pt if 3-5 items correct, 0 otherwise.

5) (10 pts) ALG (Base Conversion)

(a) (5 pts) Convert 1654_8 to hexadecimal.



$3AA_{16}$

Grading: 2 pts for converting from base 8 to base 2, 1 pt for regrouping from sets of three values to sets of four values, 2 pts for converting regrouped base 2 to base 16.

(b) (5 pts) Convert 1925_{10} to octal.

1925/8	=	240	with remainder	5
240/8	=	30	with remainder	0
30/8	=	3	with remainder	6
3/8	=	0	with remainder	3

Answer: $1928_{10} = 3605_8$

Grading: 1 pt for each division by 8 with remainder, 1 pt for final solution in correct order

Computer Science Foundation Exam

August 14, 2015

Section I B

COMPUTER SCIENCE

NO books, notes, or calculators may be used,
and you must work entirely on your own.

SOLUTION

Question #	Max Pts	Category	Passing	Score
1	10	ANL	7	
2	10	ANL	7	
3	10	DSN	7	
4	10	DSN	7	
5	10	ALG	7	
TOTAL	50		35	

You must do all 5 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat.

1) (10pts) ANL (Algorithm Analysis)

Consider the recursive function `diminish` shown below:

```
double diminish(int m, int n){
    if (n == 0)
        return m;
    return  1.0/2*diminish(m,n-1)
}
```

(a) (3 pts) Let T(n) represent the run time of the function `diminish`. Write a recurrence relation that T(n) satisfies.

$$T(n) = T(n-1) + O(1)$$

Grading: 1 pt for each component. Note that the last component may be any positive integer constant and still receive full credit.

(b) (6 pts) Using the iteration method, determine a closed-form solution (Big-Oh bound) for T(n).

$$T(n) = T(n-1) + O(1)$$
$$T(n) = T(n-2) + O(1) + O(1)$$
$$T(n) = T(n-3) + O(1) + O(1) + O(1)$$
$$T(n) = T(n-k) + kO(1)$$

Plugging in k = (n-1), we find:

$$T(n) = T(n-(n-1)) + (n-1)O(1)$$
$$T(n) = T(1) + (n-1)O(1)$$
$$T(n) = 1 + (n-1)O(1)$$
$$T(n) = O(n)$$

Grading: 3 pts for couple iterations and generalization, 3 pts for rest - allow with or without Big-Oh notation. Give full credit to any function that is O(n), most likely ones are n, 2n, 3n, with a plus or minus 1, potentially.

(c) (1 pt) In terms of the values of `m` and `n`, respectively, what does the function call `diminish(m,n)` return? (You may assume that `m` and `n` are both positive.)

$$diminish(m,n) = \frac{m}{2^n}$$

Grading: 1 pt for correct answer, 0 otherwise

2) (10 pts) ANL (Algorithm Analysis)

(a) (5 pts) An algorithm for searching for a housing contract in a database of n records takes *O(lg n)* time. When $n = 2^{20}$, one million searches can be performed in one fifth of a second. If we increase the database to size $n = 2^{25}$, how long will 500,000 searches take?

Let T(n) represent the time one search takes. Thus, $T(n) = clgn$, for some constant c. Using the given information, we have:

$$10^6 T(n) = .2sec = 10^6 clg(2^{20})$$
$$.2sec = 10^6(20)c$$
$$c = 10^{-8}sec$$

We are being asked to find $500000T(2^{25})$:

$$500000T(2^{25}) = 5(10^5)(10^{-8}sec)\lg(2^{25})$$
$$= 5(10^5)(10^{-8}sec)25$$
$$= 125(10^{-3}sec)$$
$$= 125ms, or\ .125\ sec$$

Grading: 1 pt setting up valid equation, 2 pts solving for constant, 2 pts for plugging into second part and getting the answer - can be represented in any unit of time, though sec and ms are probably going to be the most common.

(b) (5 pts) A shortest distance algorithm on an $n \times m$ street grid runs in *O(nm)* time. If the algorithm takes 2 seconds to run on a 4000 × 3000 sized grid, how long will it take on a grid of size 2000 × 18000 sized grid?

Let T(n, m) represent the time algorithm takes . Thus, $T(n,m) = cnm$, for some constant c. Using the given information, we have:

$$T(4000,3000) = 2sec = c(4000)(3000)$$
$$c = \frac{1}{6}10^{-6}sec$$

Now we solve for T(2000, 18000):

$$T(2000,18000) = \left(\frac{1}{6}10^{-6}\text{sec}\right)(2000)(18000) = \frac{36}{6}sec = 6\ sec$$

Grading: 1 pt setting up valid equation, 2 pts solving for constant, 2 pts for plugging into second part and getting the answer - can be represented in any unit of time, though sec is probably going to be the most common.

3) (10 pts) DSN (Linked Lists)

Write a function, `moveFrontToBack`, that takes in a pointer to the front of a ***doubly*** linked list storing an integer, moves the first node of the list to the back of the list and returns a pointer to the new front of the list. If the list contains fewer than two elements, the function should just return the list as it is. (Note: prev points to the previous node in the list and next points to the next node in the list.)

Use the struct definition provided below.

```
typedef struct dllnode {
    int value;
    struct dllnode* prev;
    struct dllnode* next;
} dllnode;

dllnode* moveFrontToBack(dllnode* front) {

    if (front == NULL || front->next == NULL)     // 2 pts
        return front;

    dllnode* newfront = front->next;

    dllnode* back = newfront;                  // 3 pts iterating to back
    while (back->next != NULL)
        back = back->next;

    back->next = front;                        // 1 pt
    front->prev = back;                        // 1 pt
    front->next = NULL;                        // 1 pt
    newfront->prev = NULL;                     // 1 pt

    return newfront;                           // 1 pt
}
```

Grading conceptually: 2 pts base cases, 3 pts iterating to back, 5 pts reattaching things and returning.

4) (10 pts) DSN (Binary Trees)

Mark and his buddy Travis have devised a password scheme to secure files that they send among themselves. Their scheme hides the password in a string of English letters. The password is the alphabetically ordered sequence of the consonants in the string. So as not to have to compute the password each time, Mark has written a function called `printPassword`, which takes the letters of the original string stored in a binary search tree and prints out the password. For example, if the string in the message is **mental**, the password printed out would be **lmnt**. Or if the string was ***fragile***, then the password would be ***fglr.*** You may call the following function in your solution:

```
// Returns 1 if c is a consonant, 0 otherwise.
int isConsonant(char c)
```

Using the struct definition given below, complete the function in the space provided.

```
typedef struct treenode {
    char ch;
    struct treenode *left;
    struct treenode *right;
} treenode;

void printPassword(treenode* root) {

    if (root != NULL) {                              // 2 pts
        printPassword(root->left);                   // 2 pts
        if (isConsonant(root->ch))                   // 2 pts
            printf("%c", root->ch);                  // 2 pts
        printPassword(root->right);                  // 2 pts
    }
}
```

5) (10 pts) ALG (Sorting)

(a) (4 pts) Consider sorting the array below in ascending order using Bubble Sort. Show the contents of the array after each iteration of the outer loop.

Original	6	12	1	9	4	2
1st iteration	**6**	**1**	**9**	**4**	**2**	**12**
2nd iteration	**1**	**6**	**4**	**2**	**9**	**12**
3rd iteration	**1**	**4**	**2**	**6**	**9**	**12**
4th iteration	**1**	**2**	**4**	**6**	**9**	**12**
5th iteration	**1**	**2**	**4**	**6**	**9**	**12**

Grading: 1 pt for each of the first four lines, only award the point if the whole line is correct.

(b) (6 pts) Please provide the best case and worst case run times (Big-O) for each of the following three sorting algorithms, in terms of n, the number of elements being sorted.

Sort	**Best Case**	**Worst Case**
Merge Sort	O(nlgn)	O(nlgn)
Quick Sort	O(nlgn)	$O(n^2)$
Insertion Sort	O(n)	$O(n^2)$

Grading: 1 pt for each. Each is either correct or not correct. Accept if O() is left out.

Computer Science Foundation Exam

August 14, 2015

Section II A

DISCRETE STRUCTURES

NO books, notes, or calculators may be used, and you must work entirely on your own.

SOLUTION

Question	Max Pts	Category	Passing	Score
1	15	PRF (Induction)	10	
2	10	PRF (Logic)	6	
3	15	PRF (Sets)	10	
4	10	NTH (Number Theory)	6	
ALL	50		32	

You must do all 4 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat.

1) (15 pts) PRF (Induction)

Use mathematical induction to prove that, for every positive integer n,

$$3 \mid (5^n + (-1)^{n+1})$$

Let P(n) be an open statement equal to the proposition above.

Base Case (n=1):
$5^1 + (-1)^2 = 5 + 1 = 6 = 3(2)$
$3 \mid 3(2)$ by definition of divisible. **(Grading: 2 pts)**

Inductive Hypothesis:
Assume $P(k)$ is true for all $k, 2 \leq k$ In other words, assume the following: **(Grading: 2 pts)**

$$3 \mid (5^k + (-1)^{k+1})$$

Inductive step (show true for n=k+1): **(Grading: 2 pts)**
Let $a = 5^{k+1} + (-1)^{k+1+1} = 5^{k+1} + (-1)^{k+2}$

Our goal is to show that $3 \mid a$. **(Grading: 6 pts - 2 pts per step, roughly)**

$$a = 5^{k+1} + (-1)^{k+2} = 5 \cdot 5^k + (-1)(-1)^{k+1} = (6-1)(5^k) + (-1)(-1)^{k+1}$$
$$= 6(5^k) + (-1)(5^k + (-1)^{k+1})$$

To simplify things, we can use two variables b and c:

$$b = 6(5^k), c = (-1)(5^k + (-1)^{k+1})$$

By manipulating b, it follows $b = 6(5^k) = 3 \cdot 2 \cdot (5^k)$, By definition of divisible, we know $3 \mid b$. **(Grading: 1 pt)**

By the inductive hypothesis and the multiplicative divisibility law, it follows that $3 \mid c$.
(Grading: 2 pts use of IH)

By the additive divisibility law it follows that $3 \mid (b + c)$. **(No need to state the law…)**

$$\therefore 3 \mid a$$

By the principle of mathematical induction, our conjecture holds.

$$Q.E.D.$$

2) (15 pts) PRF (Logic)

Validate the following argument using the laws of logic, substitution rules or rules of inference. List the rule used in each step and label the steps used in each derivation.

$$\begin{array}{l} (p \vee r) \rightarrow q \\ \neg s \\ (p \wedge r) \vee s \\ \neg q \vee t \\ \hline \therefore t \end{array}$$

1. $\neg s$	Premise
2. $(p \wedge r) \vee s$	Premise
3. $p \wedge r$	Disjunctive Syllogism on (1) and (2)
4. $(p \vee r) \rightarrow q$	Premise
5. p	Simplification on (3)
6. $p \vee r$	Disjunctive Amplification on (5)
7. q	Modus Ponens on (4) and (6)
8. $\neg q \vee t$	Premise
9. t	Disjunctive Syllogism on (7) and (8)

$Q.E.D.$

Grading: -1 per incorrect step, -1/2 per incorrect reason, give 0 if left blank.

3) (10 pts) PRF (Sets)

Prove the following statement is true:

Let A and B be two finite sets. Prove that $A \nsubseteq B \Rightarrow A \neq \emptyset \wedge \left(\exists x \left(x \in (A \cup B)\right)\right)$.

Proving the contrapositive is equivalent to proving the original statement:

$$\neg\left(A \neq \emptyset \wedge \left(\exists x \left(x \in (A \cup B)\right)\right)\right) \rightarrow A \subseteq B$$

Next rework the premise using DeMorgan's law for quantifiers and Logical DeMorgan's law:

$$A = \emptyset \vee \left(\forall x \left(x \notin (A \cup B)\right)\right) \rightarrow A \subseteq B$$

Use DeMorgan's law to simplify the above statement:

$$A = \emptyset \vee \left(\forall x \left(x \notin A \wedge x \notin B\right)\right) \rightarrow A \subseteq B$$

The Universal quantified statement is the same as the definition of empty set. So we can rework this statement further:

$$A = \emptyset \vee (A = \emptyset \wedge B = \emptyset) \rightarrow A \subseteq B$$

As everything here is a proposition, we can use the Absorption Law from the Laws of Logic to obtain the following:

$$A = \emptyset \rightarrow A \subseteq B$$

As we only used logical equivalences to rework our statement, proving this statement is equivalent to the original.

Our goal is to prove $A \subseteq B$. To show this we need to show the following is true:

$$\forall x\ (x \in A \rightarrow x \in B)$$

This statement holds vacuously as there are no elements in A. This is determined from the premise $A = \emptyset$.

$$Q.E.D.$$

Alternate solution:

Use direct proof. If A isn't a subset of B, there must exist some element of x such that $x \in A$ and $x \notin B$. Thus, it follows that A is non-empty (since it contains x). By definition of union it also follows that $x \in (A \cup B)$, completing the proof.

Grading: Many, many ways to do this, conceptually, grade as follows:
5 pts for showing that A is non-empty,
5 pts for showing that A union B is non-empty.

4) (10 pts) NTH (Number Theory)

Prove for an arbitrary prime number p that there always exists some composite number q where $\gcd(p, p+q) > 1$.

First we will convert the English statement into a Quantified Statement:

$$\forall p\ \big(p\ is\ prime \rightarrow\ \exists q\ (q\ is\ composite\ \wedge \gcd(p, p+q) > 1)\big)$$

So for an arbitrary p, we must find a specific q that is composite and the gcd statement above holds. Let q be set by the below equation:

$$q = 2p$$

This value of q is not prime as it contains the divisor 2 in addition to 1 and $2p$. A composite number is a positive integer > 1 that is not prime. The established q meets that criteria.

Now we merely need to show that q will satisfy our gcd inequality:

$$\gcd(p, p+q) = \gcd(p, p+2p) = \gcd(p, 3p)$$

As p is prime, the gcd of the above statement must be either 1 or p as these are the only options for the divisor. We know $p \mid 3p$ by definition of divisible. As $p > 1$, due to p being prime, it follows:

$$\gcd(p, 3p) = \mathrm{p}$$

As we have shown there always exists a satisfactory q for all p, our proof is complete.

$$Q.E.D.$$

Grading: 5 pts for picking a value of q. 5 pts for showing that the corresponding gcd is greater than 1.

Computer Science Foundation Exam

August 14, 2015

Section II B

DISCRETE STRUCTURES

NO books, notes, or calculators may be used,
and you must work entirely on your own.

SOLUTION

Question	Max Pts	Category	Passing	Score
1	15	CTG (Counting)	10	
2	15	PRB (Probability)	10	
3	10	PRF (Functions)	6	
4	10	PRF (Relations)	6	
ALL	50		32	

You must do all 4 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat.

1) (15 pts) CTG (Counting)

Please leave your answers in factorials, permutations, combinations and powers. Do not calculate out the actual numerical value for any of the questions. Justify your answers.

Each of the following questions concerns a classroom with five girls and five boys. For each part below, treat each of these 10 students as being distinguishable from one another.

(a) (3 pts) The teacher wants to create five teams, each with one girl and one boy. How many different sets of teams can the teacher create? Note: two sets of teams are different if at least one pair of students on the same team in the first set of teams is on different teams in the second set of teams.

Label the girls g_1, g_2, g_3, g_4 and g_5. Label the boys b_1, b_2, b_3, b_4 and b_5. We have 5 choices of boys to pair with g_1, followed by four remaining choices of boys to pair with g_2, etc. Thus, there are 5! sets of teams that the teacher can create. **(Grading: 3 points all or nothing.)**

(b) (5 pts) In how many ways can the class line up such that no girl is next to another girl and no boy is next to another boy?

To satisfy the criterion, we must alternate genders, but the first person in line may be either a girl or a boy. Additionally, there are 5! ways to order the girls, 5! ways to order the boys, the choices of which are independent of one another. Using the multiplication principle, our final answer is $2(5!)^2$. **(Grading: 2 pts for starting girl and starting boy, 1 pt for each 5!, 1 pt for multiplying)**

(c) (7 pts) How many subsets of students in the class contain more girls than boys? (Please give an exact numeric answer. Note that the 5th row of Pascal's Triangle is 1, 5, 10, 10, 5 and 1.)

Since the number of boys and girls is equal in the class, the number of subsets containing more boys than girls is equal to the number of subsets containing more girls than boys. Let X equal the number of subsets with more girls than boys, our desired answer. Let Y equal the number of subsets with the same number of boys and girls. There are 2^{10} number of subsets in the class total. Thus, we get the equation:

$2X + Y = 2^{10} = 1024$.

We determine Y by breaking up our counting into six cases: subsets with k girls and k boys, for $0 \le k \le 5$. Since these choices are independent, we have $Y = \sum_{k=0}^{5} \binom{5}{k}^2 = 1 + 25 + 100 + 100 + 25 + 1 = 252$.

It follows that $X = \frac{1024-252}{2} = 386$.

Grading: There are <u>many ways</u> to solve this. Give partial credit as necessary. Make sure you clearly understand the student's approach to solving the problem before grading.

2) (15 pts) PRB (Probability)

(a) (7 pts) 2% of the population has disease A. Given that a patient has disease A, a test, T, correctly identifies that the patient has the disease 95% of the time. given that a patient does NOT have disease A, T correctly identifies that the patient doesn't have the disease 80% of the time. You've taken test T and have tested positive for disease A. What is the actual probability that you have the disease?

Let a be the event that a randomly selected person has disease A and let t be the event that a randomly selected person tests positive for disease A using test T. We know the following:

$$p(a) = .02, p(t|a) = .95, p(\bar{t}|\bar{a}) = .8$$

It naturally follows that, $p(\bar{t}|a) = 1 - p(t|a) = 1 - .95 = .05$ and similarly, $p(t|\bar{a}) = .2$.

We desire to find $p(a|t) = \frac{p(a\cap t)}{p(t)}$. Using a tree diagram, we can solved for both of these quantities:

$$p(a \cap t) = p(a) \times p(t|a) = .02 \times .95 = .019$$

$$p(t) = p(a)p(t|a) + p(\bar{a})p(t|\bar{a}) = .02 \times .95 + .98 \times .2 = .019 + .196 = .215$$

It follows that $p(a|t) = \frac{p(a\cap t)}{p(t)} = \frac{.019}{.215} \sim .0884(8.84\%)$.

Grading: 2 pts for filling in opposite probabilities, 1 pt for identifying what needs to be found (includes division), 2 pt for p(t), 2 pts for p(a and t)

(b) (8 pts) Consider a partition of an array of 2n+1 distinct integers where the partition element is randomly chosen. (Note: this is where we place all integers less than the partition element on the left of it and all the integers greater than the partition element on the right of it.) Let L equal the number of values less than the partition element and R equal the number of values greater than the paritition element. (Thus, L + R = 2n.) Calculate the expected value of |L - R|.

Each of the 2n+1 integers is chosen as the partition element with probability $\frac{1}{2n+1}$. Each of the ordered pairs (L, R) associated with the partition elements are (2n, 0), (2n-1, 1), ..., (n,n), ..., (1, 2n-1), (0, 2n). The corresponding values of |L - R| are 2n, 2n-2, ..., 0, 2, 4, ..., 2n-2 and 2n. Basically, each positive even value from 2 to 2n appears on the list twice while 0 appears once. Plugging into the formula for expectation, we find:

$$\boldsymbol{Exp(|L-R|) = \frac{2\sum_{k=1}^{n} 2k}{2n+1} = \frac{4\sum_{k=1}^{n} k}{2n+1} = \frac{4n(n+1)}{2(2n+1)} = \frac{2n(n+1)}{2n+1}}.$$

Grading: 2 pts for definition of expectation, 2 pts for 1/(2n+1) probability, 4 pts for algebra.

3) (10 pts) PRF (Functions)

Let $f(x) = \sqrt{3x^2 + 5x + 7}$ and $g(x) = 2^{x^2}$, both with the domain $x \geq 0$.

(a) (7 pts) What is $g(f(x))$? (Simplify your answer for full credit.)

$$g\big(f(x)\big) = g\left(\sqrt{3x^2 + 5x + 7}\right) = 2^{\sqrt{3x^2+5x+7}^2} = 2^{3x^2+5x+7}$$

Grading: 4 pts for plugging into definition, 3 pts for sqrt/sqr simplification, other simplifications aren't necessary.

(b) (3 pts) What is the range of $g(f(x))$, given that its domain is all reals values of $x \geq 0$.

Note that $f(x)$ is a continuously increasing function and that $f(0) = \sqrt{7}$. Since $g\big(f(x)\big)$ is also continuously increasing, it follows that its minimum value is $g\big(f(0)\big) = 2^7 = 128$. Thus, the range of $g\big(f(x)\big)$ is [128, ∞).

Grading: 2 pts for low bound, 1 pt for high bound, no explanation necessary.

4) (10 pts) PRF (Relations)

(a) (5 pts) Let A = {1, 2, 3, 4}, B = {x, y, z} and C = {m, n}. Let R = {(1, x), (1, z), (2, y), (4, x), (4, y), (4, z)} and S = {(x, m), (y, m), (y, n)}. What is $S \circ R$?

$$S \circ R = \{(1, m), (2, m), (2, n), (4, m), (4, n)\}$$

Grading: +1 for each correctly included order pair, -1 penalty for listing an extra item not on the real list, cap grade at 0.

(b) (5 pts) Let A = {1, 2, 3}. There are 5 equivalence relations over $A \times A$. Explicitly list all five.

Here is the proof that there are 5 such relations:

All reflexive relations over the set A must contain (1, 1), (2, 2), (3, 3). Furthermore, due to symmetry, we know that (1, 2) is in the relation iff (2, 1) is, (1, 3) is in the relation iff (3, 1) is, and (2, 3) is in the relation iff (3, 2) is. Thus, we only have freedom in selecting whether or not (1, 2), (1, 3) and (2, 3) are in the relation. We must determine how many of these $2^3 = 8$ possible relations are transitive. We are safe in including none, one or three of these ordered pairs. Any choice of exactly two of these will break transitivity. (For example, if we choose (1, 2) and (1, 3), that means our relation also has (3, 1). But note that it has (3, 1) and (1, 2) but doesn't contian (2, 3). Thus, out of the 8 possible relations, only three are not transitive.

It follows that there are 5 relations over A that are equivalence relations:

{(1, 1), (2, 2), (3, 3)}

{(1, 1), (2, 2), (3, 3), (1, 2), (2, 1)}

{(1, 1), (2, 2), (3, 3), (1, 3), (3, 1)}

{(1, 1), (2, 2), (3, 3), (2, 3), (3, 2)}

{(1, 1), (2, 2), (3, 3), (1, 2), (2, 1), (1, 3), (3, 1), (2, 3), (3, 2)}

Grading: +1 for each correctly listed relation, -1 penalty for listing a relation that isn't an equivalence relation, cap grade at 0.