Topics and Sample Problems for Exam I
Over Chapters 1-4, Project 1 and Handout

- Computability
  - Know what you can and cannot compute
  - Definition of an Algorithm
  - Complexity and "Hard" Problems
  - Efficiency and Interpretation
- Algorithm Design
  - Take conceptual design to establishment and maintenance of truth
  - Invariants (Invariant handout)
  - How to show informally that invariants hold and lead to postconditions
- Complexity Analysis
  - Given some code, find the best and worst case run times
  - Introduction to average case run times
  - Summation of simple series
- Divide and Conquer Design
  - Merge Sort
  - Binary Search
  - Correctness
  - Complexity
- Work problems developing both invariants and run time analysis
- Asymptotic Complexity – Ch3
  - Know definitions and conceptual meanings of big-O, Theta, and big-Omega) notation
  - Understand how to find appropriate constants and n0 to show asymptotic bounds for a given function
  - Be able to show the asymptotic relationship between expressions.
  - Understand how the leading constant and lesser terms can dominate run-time complexity for small values of n

- Recurrences—Ch 04
  - Given a proposed solution from the recursion tree method (or a guess), use the substitution method to verify the guess.
  - To review, do problems 4.1-1, 4.1-2, 4.1-4, 4.1-5

Chapter 1

The Dogbert computer corporation claims their new computer will run 100 times faster than that of their competitor, Apple, Inc. Given two algorithms A & B with running times as follows:

$t_A(n)=100n^3$

$t_B(n)=10^3n^4$

for what values of n (if any) will the Dogbert computer run algorithm B faster than the Apple computer will run algorithm A?

The SPAMCO computer corporation claims their new computer will run 100 times faster than that of their competitor, Dogbert, Inc. Given two algorithms A & B with running times as follows:

$t_A(n)=5000n$
$t_B(n)=10n^2$

for what values of n (if any) will the Dogbert computer run algorithm B faster than the SPAMCO computer will run algorithm A?

True or false

a. An algorithm can always be computed by a computer.
b. Hard problems, such as the traveling salesperson problem, in Computer Science are those that are difficult to understand and express easily.

Chapter 2

Problem: Search for v in A[1..n] if v is found find the index k so that v=a[k] otherwise k=0

Write the assertions for invariant in the following code for loop

```
i=0, k=0

for i=1 to n

        If A[i]=v
                k=i;
                break; // if you donot break you get last match if there are multiple
                matches,
                with break you get the first match.

end for
```

Post condition:

Problem: Sum n natural numbers. Write the invariant for the while loop

```
i=0;
sum=0;

while ++i<= n

        sum=sum+i;
end while
```

PostCondition:

Problem Insertion sort. Write invariant for the following code

```
i=1
for i=2 to length(A)
        Do      key=A[i]
                k=i-1
                while(k>0&& key < A[k])
                        A[k+1]=A[k]
                        k=k-1
                endwhile
                A[k+1]=key
endFor
```

Post condition:

Problem Bubble Sort

Write invariant for the following code

Note: A(1..i-1)≤ A(i-1..n) means every element in A(1..i-1) is less that or e□ual to every element in A(i-1..n)

```
for i=1 to length(A)-1
        for j=lengh(A)-1 to i
                if A[j] > A[j+1]
                        swap(A[j], A[j+1])
        endfor
endfor
```

What is $\sum_{i=0}^{n} (7/13)^i$

What is $\sum_{i=0}^{\log_2 n} cn$ where c is a constant?

What is $\sum_{i=0}^{n} c(i \square n)$ where c is a constant?

Given the following recursive function to determine of an element x is in an array a:

```
Search(x,a[],i)
  if (i==0) return FALSE;
  else if (a[i-1]==x) return TRUE;
  else return Search(x,a,i-1);
```

Formulate the running time bounds of Search as a recurrence to determine what are the best, worst, and average time and space complexities of Search(x,a,n) (20 pts)

True or False, explain your answer for each for full credit:

c. A loop invariant is true before the loop starts, during each execution of the loop, and right up until the moment when the loop is exited, and then the postcondition becomes true.

Chapter 3

Arrange the following expressions by growth rate from slowest to fastest. Show your work for full credit.

$4n^2$ $\log_3 n$ 3^n $20n$ 2 $\log_2 n$ $n^{2/3}$

Is $7n^2 + 3n + 4 = O(n^3)$? Is $7n^2 + 3n + 4 = \square(n^3)$? Prove your answers.

State whether the following are True or False and why.

a. (5 pts) $\log_2 n$ is $O(\log_3 n)$

b. (5 pts) $10 \cdot n \square_1^n i$ is $\square(n^3)$

d. If f(n) is $\square$(g(n)), then f(n) is O(g(n)) and f(n) is $\square$(g(n)).

Show the following are correct:

1. $5n^2-6n=\square(n^2)$
2. $n^3+10^6n^2=\square(n^3)$
3. $6(2^n) + n^2 = \square\ (2^n)$

Show the following are incorrect

4. $10n^2+9=\square(n)$
5. $n^2 \lg n=\square(n^2)$

What is the sum of $\sum_{i=0,n} 2^i$ for n=31?

Compute $\log_2 1024$ using your calculator in base e, or 10 (the answer is 10, so practice using the log base change formula in the book).

Show $a^{\log_b c} = c^{\log_b a}$ using the rules of logarithms and exponentials

Show that

$n^2/2 - 3n = Q(n^2)$

Find c_1, c_2, n_0 to show that f(n) = Q(g(n)) where

$f(n) = 2n^2 - 100$ $g(n)= n^2$

$f(n) = n^2/2 - 4n$ $g(n)= n^2$

State if the following are true or false. Justify your answers

$\square(n) + \square(n^2) = \square(n^2)$

$\square(2^{n-1}) = \square(2^{n+1}) = \square(2^n)$

$\square(1) + \square(2) + \square(3) + \ldots + \square(n) = \square(n^2)$

$\square(2^0) + \square(2^1) + \square(2^2) + \ldots + \ldots + \square(2^n) = \square(2^n)$

Chapter 4

Find the solution by Characteristic e□uation method

$F_0 = 0$, $F_1 = 1$

$F_n = F_{n-1} + F_{n-2} \qquad n \geq 2$

Find the solution by substitution method

$T_n = T_{n-1} + n$

$T_n = T_{n-1} + \lg n$

$T(n) = T(n-a) + T(a) + cn$

$T_n = 3T_{n/2} + n$ if $n>1$

$T_n = 3T_{n/4} + n^2$ if $n>1$

$T_n = 16T_{n/4} + n^2$

$T_n = 7T_{n/2} + n^2$

$T_n = 2T_{n/2} + \lg n$ if $n>1$

Find the solution by tree method

$T_n = T_{n-1} + n$

$T_n = T_{n-1} + \lg n$

$T_n = 3T_{n/2} + n$ if $n>1$

$T_n = 2T_{n/2} + n^2$ if $n>1$

$T_n = 2T_{n/2} + n^3$

$T_n = 2T_{n/2} + \lg n$ if $n>1$

Find the solution by Characteristic E□uation method

$T_n = T_{n-1} + n$

$T(n) = T(n-a) + T(a) + cn$

$T_n = T_{n/2} + n$ if $n>1$

$T_n = 2T_{n/2} + n$ if $n>1$

$T_n = 3T_{n/2} + n$ if $n>1$

$T_n = 2T_{n/2} + n^2$ if $n>1$

$T_n = 3T_{n/4} + n^2$ if $n>1$

$T_n = 16T_{n/4} + n^2$
$T_n = 7T_{n/2} + n^2$
$T_n = 2T_{n/2} + n^3$

$T_n = 2T_{n/2} + \lg n \quad \text{if } n>1$
$T_n = 2T_{\sqrt{n}} + \lg n$
$T_n = 3T_{n/2} + n \lg n$

Consider the recursive version of Binary Search that finds x in A[1..n] sorted ascending, if it exists and return its index. If x is not in A[1..n], the program returns 0. For simplicity, assume n is a power of 2.

```
Binsrch(A,left,right,x);

        if(left>right)
                return(0);
        mid=□(right-left+1)/2□;
        if (A[mid]==x)
                return(mid);
        else if (A[mid] <x)
                Binsrch(A,mid+1,right,x;
        else
                Binsrch(A,left,mid-1,x);
        end if;
```

a. (15 pts) Develop a recurrence for the running time T(n) of BinSrch.
b. (15 pts) Solve the recurrence from part a.

Show the solution of following recurrence for n a power of 2:

$$T(n)=2T(n/2) + n \qquad n>1$$
$$T(1)=1 \qquad \text{otherwise}$$

is $T(n)=O(n\log_2 n)$ using the substitution method.

Solve the following recurrence with the usual assumptions (15 pts):

$$T(n)=3T(n/2) + n \qquad n>4$$
$$T(4)=14 \qquad \text{otherwise}$$

Given merge sort as follows:

```
Procedure MergeSort(A[],left,right);
        if (left>=right) return;
        middle=(left+right)/2;
```

```
MergeSort(A,left,middle);
MergeSort(A,middle+1,right);
Merge(A[left..middle],A[middle+1..right]);
```

a. Set up a recurrence that describes the run-time complexity of merge sort (assume the size of A is a power of 2) (5 pts)
b. Solve the recurrence from (a) (15 pts).

c. Develop an invariant that describes what is true at the completion of each MergeSort (15 pts).

Find the sum of the heights of all nodes in a complete binary tree?

Write recursion relations in two forms and derive the result without solving them.

Find the sum of the depths of all nodes in a complete binary tree?

Write recursion relations in two forms and derive the result without solving them.