Final Exam
Winter 2024

Name ____

Net ID ______________________ (@uw.edu)

Academic Integrity: You may not use any resources on this exam except for writing instruments, your own brain, and the exam packet itself. This exam is closed notes, closed neighbor, closed electronic devices, etc.. The last two pages of this exam provide a list of potentially helpful identities as well as room for scratch work (respectively). Please detach those last two pages from the exam packet. No markings on these last two pages will be graded. Your answer for each question must fit in the answer box provided.

Instructions: Before you begin, **Put your name and UW Net ID at the top of this page.** Make sure that your name and ID are LEGIBLE. Please ensure that all of your answers appear within the boxed area provided.

Section	**Max Points**
Asymptotic Analysis	14
Pre-Midterm Data Structures	13
Hash Tables	14
Sorting	9
Graphs	18
Parallelism	12
Concurrency	15
P vs NP	5
Extra Credit	(+2)
Total	100

Section 1: Asymptotic Analysis

(4 pts)**Question 1: Asymptotic Analysis of Code**
Give a simplified Θ bound on the best and worst case running times for the given code. (By simplified we mean it should contain no constant coefficients or non-dominant terms.)

```
int doStuff(List<Integer> numbers){
    int n = numbers.size();
    int count = 0;
    if(n < 150){
        for (int i = 0; i < n; i++){
            for (int j = i; j < n; j++){
                count++;
            }
        }
    }
    else{
        for (int i = 0; i < n; i++){
            count += n;
        }
    }
    return count;
}
```

(a) Best Case: $\Theta\Big(\boxed{}\Big)$

(b) Worst Case: $\Theta\Big(\boxed{}\Big)$

(6 pts)**Question 2: Which is larger?**
For each pair of functions $f(n)$ and $g(n)$ below, select the choice which characterizes the asymptotic relationship between f and g. Write the letter corresponding with your answer in the box provided.

1. $f(n) = n^2 \log(4n)$, $g(n) = n \log_2(4^n)$
   A. $f(n) \in \Theta(g(n))$
   B. $f(n) \in O(g(n))$ and $f(n) \notin \Theta(g(n))$
   C. $f(n) \in \Omega(g(n))$ and $f(n) \notin \Theta(g(n))$
2. $f(n) = n^{1.5}$, $g(n) = n \log(n)$
   A. $f(n) \in \Theta(g(n))$
   B. $f(n) \in O(g(n))$ and $f(n) \notin \Theta(g(n))$
   C. $f(n) \in \Omega(g(n))$ and $f(n) \notin \Theta(g(n))$
3. $f(n) = n$, $g(n) = \sum_{i=0}^{n} \frac{i}{2}$
   A. $f(n) \in \Theta(g(n))$
   B. $f(n) \in O(g(n))$ and $f(n) \notin \Theta(g(n))$
   C. $f(n) \in \Omega(g(n))$ and $f(n) \notin \Theta(g(n))$

(4 pts)**Question 3: Recurrence Relation**

The method given below determines whether the sum of values in a given list is odd. First, express the worst case running time of the code as a recurrence relation. Next solve that recurrence relation using any method of your choice and express the running time as a simplified Θ bound.

```
boolean oddSum(int[] nums){
    return oddSumRec(nums, 0, nums.length);
}
boolean oddSumRec(int[] nums, int lo, int hi){
    if(hi-lo <= 0){return false;}
    if(hi-lo == 1){return nums[lo]%2 == 1;}
    if(hi-lo == 2){return (nums[lo] + nums[lo+1])%2 == 1;}
    int third = (hi-lo)/3;
    boolean left = oddSumRec(nums, lo, lo+third);
    boolean middle = oddSumRec(nums, lo+third, hi-third);
    boolean right = oddSumRec(nums, hi-third, hi);
    boolean odd = False;
    odd = (odd != left); // Note: != is the same as XOR
    odd = (odd != right);
    odd = (odd != middle);
    return odd;
}
```

1. Recurrence Relation:

2. Solved and simplified Θ bound:

Section 2: Pre-Midterm Data Structures

(5 pts)**Question 4: Heap**
Answer each question below as it relates to a 0-indexed binary min heap containing 65 items. 0-indexed means the root of the tree is at index 0 in its array representation.

1. What is the height of the tree (recall that a one-node tree has height 0)?
2. If we call percolate up on index 15, which index will we compare to?
3. If we call percolate down on index 15, which two indices will we compare to?
4. What is the smallest index which contains a leaf?
5. What index contains the smallest value?

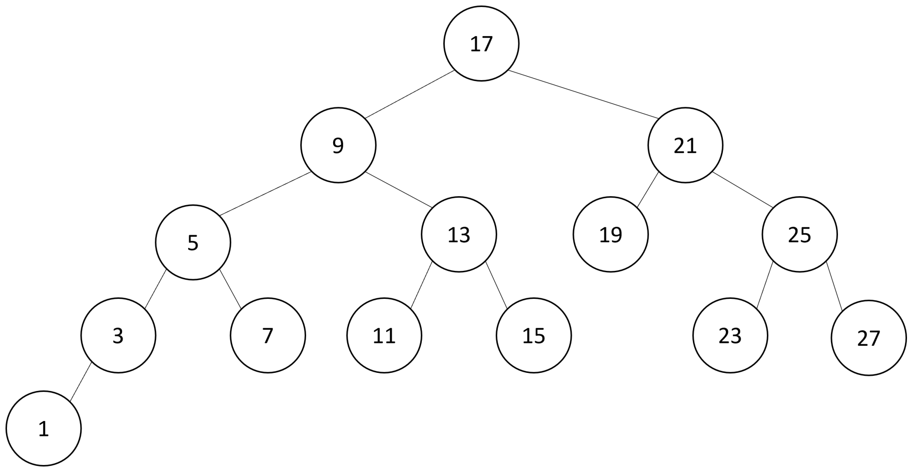
(4 pts)**Question 5: B Tree**
Suppose we had a B Tree with $L = 5$ and $M = 20$ that has height 3.

1. What is the minimum number of items in the data structure?
2. What is the maximum number of items in the data structure?

(4 pts)**Question 6: AVL Tree**

Using the AVL Tree below, complete the table by indicating the type of AVL Tree rotation that would occur for each key inserted ("single", "double", or "none").

Each row should be considered completely independently (i.e. "reset" to the image between rows).



Key Inserted	Rotation Type (write "single", "double", or "none")
0	
2	
22	
26	

Section 3: Hash Tables

(2 pts)**Question 7: Prime length**
In 1-2 sentences, explain why it is generally preferable to have a separate chaining hash table with a prime-numbered length.

(5 pts)**Question 8: Quadratic Probing**
Insert 18, 9, 29, 48, 12, 28, 19 (in that order) into the open addressing hash table below. You should use the primary hash function $h(k) = k\%10$. In the case of collisions, use quadratic probing for collision resolution. If an item cannot be inserted into the table, indicate this and continue inserting the remaining values.

Items that could not be inserted:

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

(5 pts)**Question 9: Double Hashing**
Insert 55, 15, 25, 28, 8, 48 (in that order) into the open addressing hash table below. You should use the primary hash function $h(k) = k\%10$. In the case of collisions, use double hashing for collision resolution where the secondary hash function is $g(k) = 1 + (k\%9)$. If an item cannot be inserted into the table, indicate this and continue inserting the remaining values.

Items that could not be inserted:

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

(2 pts)**Question 10: Quadratic v. Linear Probing**
In 1-2 sentences, explain why one might chose to use Quadratic probing over linear probing.

Section 4: Sorting

(4 pts)**Question 11: Sort Suggestions Hotline**
For each scenario below, select the sorting algorithm property the situation most needs, then select the *fastest* sorting algorithm with that property. Select only from the options provided.

1. I am running election simulations of the voters in the Oklahoma governor's race. There are about 4 million voters in the state, and I will be running over 100 million simulations. I want to sort those outcomes by the number votes cast for the current governor.

   Property Options: In Place, Stable, Adaptive, Online, Non-Comparison-Based.

   Property Needed:

   Algorithm Options: Quick Sort, Insertion Sort, Heap Sort, Radix Sort.

   Algorithm Suggestion:

2. I need to maintain a list of a grocery store's inventory items sorted by price. Every time we get a new delivery, though, some small number of the items may have had price increases or decreases, and so I'll need to completely re-sort the list.

   Property Options: In Place, Stable, Adaptive, Online, Non-Comparison-Based.

   Property Needed:

   Algorithm Options: Quick Sort, Insertion Sort, Merge Sort.

   Algorithm Suggestion:

(2 pts)**Question 12: Stable Sort**
Which of the following best matches the definition of Stable Sort? Write the letter of your choice in the box.

A. Items are re-ordered by swapping within the given list data structure

B. There is no randomness used in the sorting algorithm

C. The worst case running time matches the best case running time

D. Only non-equal elements may change relative order

E. If you re-run the algorithm on an already-sorted list, no items will change order

(3 pts)**Question 13: Quick Sort Runtime**

Using 1-2 sentences, explain why it might be a bad idea to always use the item at the left-most index as the pivot for Quick Sort.

Section 5: Graphs

(4 pts)**Question 14: Minimum number of edges**
Below we will give several descriptions of a graph. In the box provided, indicate the *minimum* number of edges that graph could have.

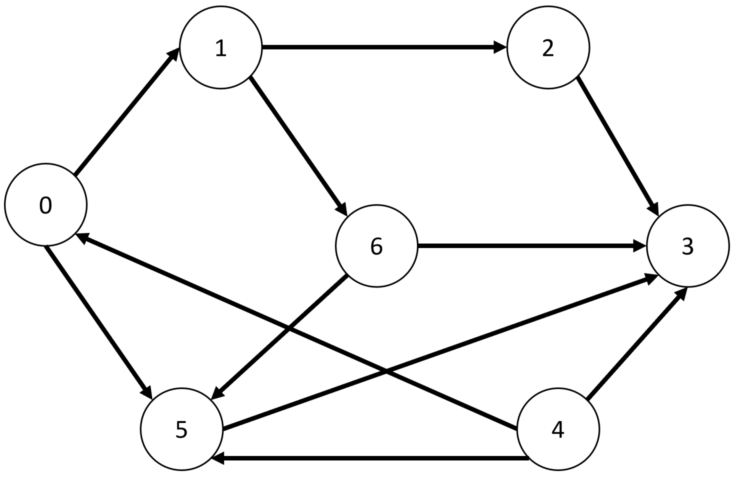
1. A connected undirected graph with 10 nodes

2. A weakly-connected directed graph with 10 nodes

3. A strongly-connected directed graph with 10 nodes

4. A Complete undirected graph with 10 nodes

(4 pts)**Question 15: Topological Sort**
List the nodes in the graph below such that they are in two different topologically sorted orderings.

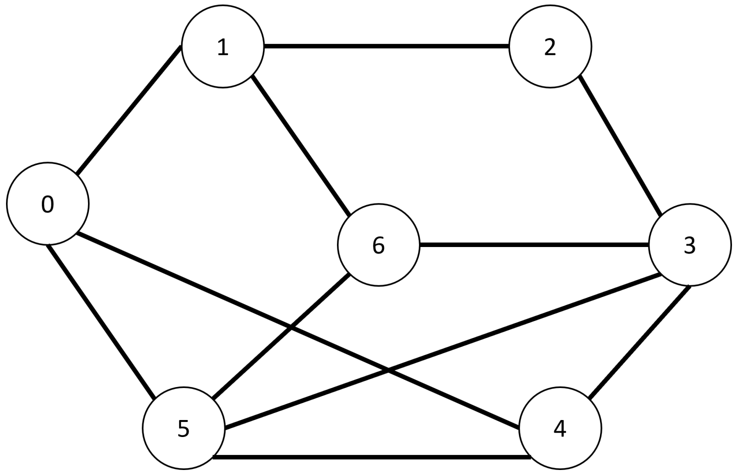


Topological Order 1:

Topological Order 2:

(2 pts)**Question 16: BFS**

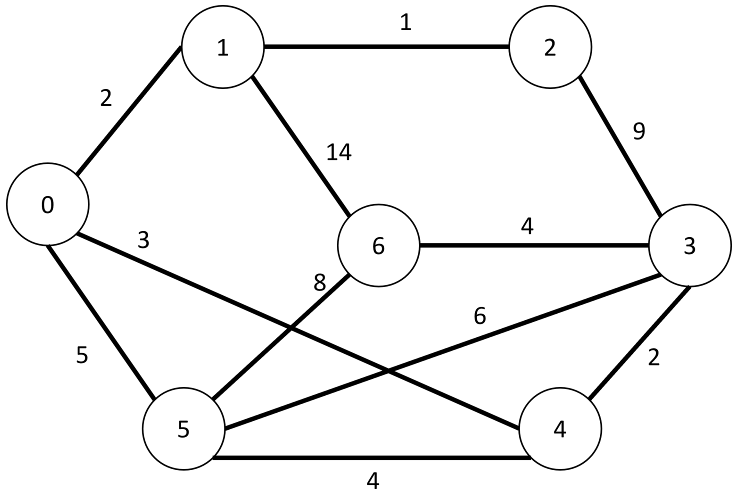
For the graph below, list the nodes in an order that they might be removed from the queue in a BFS starting from node 6 (note that this is the same as the previous graph, but now undirected).



BFS Order:

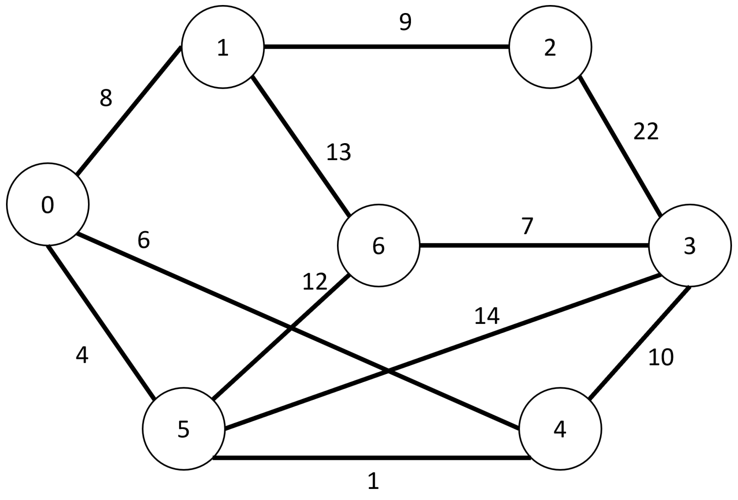
(2 pts)**Question 17: Dijkstras**

For the graph below, list the nodes an order that they might be removed from the priority queue when running Dijkstra's algorithm starting from node 6 (note that his is the same as the previous graph, but now with weights).



Dijkstra's Order:

(6 pts)**Question 18: MSTs**
The next 2 questions will relate to running minimum spanning tree algorithms (Kruskal's and Prim's) on the graph below (which is the same as the previous graph, but with different weights):



1. What are the weights of the first three edges added to the minimum spanning tree when running Kruskal's algorithm?

   First edge's weight:

   Second edge's weight:

   Third edge's weight:

2. What are the first three edges added to the minimum spanning tree when running Prims's algorithm starting with node 0?

   First edge's weight:

   Second edge's weight:

   Third edge's weight:

Section 6: Parallelism

(8 pts)**Question 19: ForkJoin**
For this question you will complete a parallel implementation of the following sequential method using the Java ForkJoin Framework.

The sequential method determines whether the given array's sum is an odd number.

A list has an odd sum if and only if there is an odd number of odd values. The provided algorithm maintains a boolean variable `oddOdds` to indicate whether an odd number of odd values have been seen thus far. It is initially `false`, then it inverts `oddOdds` once per each odd value in the input.

```
boolean oddSum(int[] arr){
    boolean oddOdds = False;
    for(int i = 0; i < arr.length; i++{
        if (arr[i]%2 == 1){
            oddOdds = !oddOdds;
        }
    }
    return oddOdds;
}
```

On the next page we have provided the majority of the code to implement `oddSum` in parallel using ForkJoin and RecursiveTask. In particular, we have provided:

– a main class which invokes the ForkJoin Pool

– the constructor for the class which extends RecursiveTask

– the sequential code within `compute` that runs when the array length is below the sequential cutoff (we chose 100 as the cutoff)

Complete our implementation by finishing the compute method.

(Hint: If you get stuck on coming up with an algorithm, question 3 provides a divide-and-conquer algorithm for this problem.)

```
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveTask;

public class Main {
    public static final ForkJoinPool fjPool = new ForkJoinPool();
    public static Boolean OddSum (int[] input) {
        return fjPool.invoke(new OddSumTask(input, 0, input.length));
    }
}

public class OddSumTask extends RecursiveTask<Boolean> {
    int[] arr;
    int hi;
    int lo;

    public OddSumTask(int[] arr, int lo, int hi){
        this.arr = arr;
        this.hi = hi;
        this.lo = lo;
    }

    public Boolean compute(){
        if (hi-lo < 100){
            boolean oddOdds = false;
            for(int i = lo; i < hi; i++){
                if (arr[i]%2==1)
                    oddOdds = !oddOdds;
            }
            return oddOdds;
        }
        // finish the compute method here









    }
```

(2 pts)**Question 20: Map,Reduce,Pack**
Each option below describes a combination of map, reduce/fold, and pack/filter. Write "True" or "False" in each box to indicate whether that option would also compute `oddSum`?

A. Map all values of the array to -1 if even and 1 if odd, then use a reduction to find the sum of the array, returning true if the sum is positive and false otherwise.

B. Pack all of the values using the boolean function `x%2 == 1`, then return true if the length of the resulting array is odd (and false otherwise).

(2 pts)**Question 21: Parallel Pack**
Suppose after the first stage of a parallel pack (i.e. filter) operation (that is, the map applying a boolean function to each index of the input array), you find that the value at index i is 1, at index $i+1$ is 1, and at index $i+2$ is 0. That is, the parallel map result appears like this:

...	1	1	0	...
	i	$i+1$	$i+2$	

Next, suppose that after the second stage of the same parallel pack operation (that is, the parallel prefix sum), you find that the value at index i is 10.

For each question below we give an index from the input array. If that item appears in the final output array, give its index. If does not appear, write "None".

1. At what index of the output array will you find the value that was at index $i+1$ of the input array?

2. At what index of the output array will you find the value that was at index $i+2$ of the input array?

Section 7: Concurrency

The questions on this page and the next use the following three java classes below being used in a parallel implementation. All three relate to writing to and erasing a whiteboard.

```java
public class WhiteBoard{
    public String contents = "";
    synchronized public void writeTo(String text){
        this.contents += text;
        System.out.println(contents);
    }
}
public class Marker{
    public int inkRemaining;
    public Marker(){
        this.inkRemaining = 10;
    }
    public void writeWith(Whiteboard wb, String text){
        int written = 0;
        while(this.inkRemaining > 0 && written < text.length()){
            wb.writeTo(text.charAt(written++));
            inkRemaining--;
        }
    }
}
public class Eraser{
    public void erase(Whiteboard wb){
        synchronized(wb){
            wb.contents = "";
            wb.writeTo("");
        }
    }
}
```

(1 pt)**Question 22: Erase Deadlock?**
The `erase` method in the `Eraser` class does not cause deadlock even though both it and `writeTo` require the same lock. Using 1 sentence, explain why.

(2 pts)**Question 23: Negative Ink**

If no marker objects are shared among threads then it is impossible for a marker to have negative ink. Using 1-2 English sentences, describe how three threads using the same marker could cause that marker to have the value `-2` in the `inkRemaining` field.

(2 pts)**Question 24: Nonsense Text**

Suppose two threads have a reference to the same `Whiteboard` object in a variable called `wb`. The first thread calls `new Marker().writeWith(wb,"one")`. The second thread calls `new Marker().writeWith(wb,"two")`. Using 1-2 English sentences, describe how it could be that the final value of `wb.contents` is `"otwneo"`?

(2 pts)**Question 25: Adding Syncronized**

Suppose we made the `writeWith` method synchronized, i.e., we change its signature to:

```
synchronized public void writeWith(Whiteboard wb, String text)
```

Write "True" or "False" to indicate whether the previous two scenarios above are still possible.

1. It is still possible for a marker to have negative ink when shared among multiple threads.

2. It is still possible for two threads (with one calling `new Marker().writeWith(wb,"one")` and the other calling `new Marker().writeWith(wb,"two")`) to cause `wb.contents` to become `"otwneo"`.

(8 pts)**Question 26: Deadlock and/or Race Condition**
Below we provide a partial implementation of a doubly-linked list. It has a pointer to the first node (called `front`) and the last node (called `back`). Each node has an integer value, and a reference to the next and previous nodes. The `add` method appends a node with the given value to the end of the list. The `set` method changes the value of the node at the given index.

For this question you will consider potential implementations of a `setBackwards` method, which does the same thing as `set`, but working from the back of the list instead of the front.

For each `setBackwards` implementation, indicate whether a thread invoking the `set` method above and a parallel thread invoking the `setBackwards` method on a shared `LinkedList` instance has a potential for deadlock and whether it as a race condition by writing "yes" or "no" in the corresponding box.

```
public class LinkedList{
    private LLNode front;
    private LLNode back;
    private class LLNode{
        public LLNode prev;
        public LLNode next;
        public int value;
    }
    public LinkedList(){...} //constructor

    synchronized public void add(int val){...} //add a node to the back of the list

    public void set(int index, int newVal){ //change the value at the given index
        synchronized(this.front){
            LLNode curr = this.front;
            for(int i = 0; i < index; i++)
                curr = curr.next;
            synchronized(curr){
                curr.value = newVal;
            }
        }
    }
    //change the value at the given index from the back
    public void setBackwards(int index, int newVal){...}
}
```

1.

```
public void setBackwards(int index, int newVal){
    synchronized{this.back}{
        LLNode curr = this.back;
        for(int i = 0; i < index; i++){
            curr = curr.prev;
        }
        synchronized{curr}{
            curr.value = newVal;
        }
    }
}
```

Deadlock?

Race Condition?

2.

```
public void setBackwards(int index, int newVal){
    synchronized{this.front}{
        LLNode curr = this.back;
        for(int i = 0; i < index; i++){
            curr = curr.prev;
        }
        synchronized{curr}{
            curr.value = newVal;
        }
    }
}
```

Deadlock?

Race Condition?

3.

```
public void setBackwards(int index, int newVal){
    synchronized{this.back}{
        LLNode curr = this.back;
        for(int i = 0; i < index; i++){
            synchronized(curr){
                curr = curr.prev;
            }
        }
        curr.value = newVal;
    }
}
```

Deadlock?

Race Condition?

4.

```
synchronized public void setBackwards(int index, int newVal){
    LLNode curr = this.back;
    for(int i = 0; i < index; i++){
        curr = curr.prev;
    }
    curr.value = newVal;
}
```

Deadlock?

Race Condition?

Section 8: P vs NP

(5 pts)**Question 27: Complexity Classes**

For each statement below indicate whether it is known to be true, known to be false, or whether the truth of the statement is not yet known by writing "True", "False", or "unknown" in the corresponding box.

1. If a problem can be ***solved*** by an algorithm with running time $\Theta(n^7)$ then it belongs to class P.

2. If a problem can be ***solved*** by an algorithm with running time $\Theta(n^7)$ then it belongs to class NP.

3. If a problem can be ***verified*** by an algorithm with running time $\Theta(n^7)$ then it belongs to class P.

4. If a problem can be ***verified*** by an algorithm with running time $\Theta(n^7)$ then it belongs to class NP.

5. If a problem can be ***verified*** by an algorithm with running time $\Theta(n^7)$ then it belongs to class EXP.

Extra Credit

(2 pts)**Question Extra Credit: What did you learn this quarter?**
Nathan's grandmother, Elouise, is 94 years old and has never used a computer before in her life. Summarize what you learned this quarter in a way that Elouise would appreciate.

Scratch Work

Nothing written on this page will be graded.

Identities

Nothing written on this page will be graded.

Summations

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x} \text{ for } |x|< 1$$

$$\sum_{i=0}^{n-1} = \sum_{n}^{i=1} = n$$

$$\sum_{i=0}^{n} i = 0 + \sum_{n}^{i=1} i = \frac{n(n+1)}{2}$$

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

$$\sum_{i=0}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$$

$$\sum_{i=0}^{n-1} x^i = \frac{1-x^n}{1-x}$$

$$\sum_{i=0}^{n-1} \frac{1}{2^i} = 2 - \frac{1}{2^{n-1}}$$

Logs

$$x^{\log_x(n)} = n$$

$$\log_a(b^c) = c\log_a(b)$$

$$a^{\log_b(c)} = c^{log_b(a)}$$

$$\log_b(a) = \frac{\log_d(a)}{\log_d(b)}$$