

Week 1 Flipped Activities

1. Setting up and Testing Your Programming Environment

- (a) Setup algs4 from percolation assignment page. Be sure to uninstall DrJava from 126 and reinstall.
- (b) Copy the code Maze.java from flipped page, compile and run. This code uses algs4 library WeightedQuickUnionUF.
- (c) Try the code with different values of N including invalid values and edge cases.

2. Analysis of runtime

The runtime of an algorithm can be estimated using experimental values.

- (a) Build a table of values, N versus runtime T using the Maze (given above). Consider only the values greater than 1 second. Why is that we would not consider times under 1 second?
- (b) Assuming the Maze code runs in polynomial time, we will use the formula $T = aN^b$ to estimate runtime T for a data set of size N . Compute the values of a and b . Estimate the values of a and b up to two decimal places.

3. **Social Network Connectivity** Given a social network containing N members and a log file containing M timestamps at which times pairs of members formed friendships, design an algorithm to determine the earliest time at which all members are connected (i.e., every member is a friend of a friend of a friend ... of a friend). Assume that the log file is sorted by timestamp and that friendship is an equivalence relation. The running time of your algorithm should be $M \log N$ or better and use extra space proportional to N .

4. **Bitonic max** An array is bitonic if it consists of a strictly increasing sequence of keys immediately followed by a strictly decreasing sequence of keys. Design an algorithm that determines the maximum key in a bitonic array of size N in time proportional to $\log_2 N$.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a[i]	10	34	56	76	87	80	70	66	56	30	28	25	20	15	11

- State an algorithm that finds any element of an array in $\log_2 N$ time. What assumptions are made about the array in your algorithm?
 - How do we modify the algorithm (or design a new one) to solve the problem stated above?
 - Estimate the runtime of your algorithm as a function of N .
5. **Runtime Analysis** Consider the following three algorithms:
- Algorithm 1 solves problems of size N by recursively dividing them into 2 sub-problems of size $N/2$ and combining the results in time c (where c is some constant).
 - Algorithm 2 solves problems of size N by solving one sub-problem of size $N/2$ and performing some processing taking some constant time c .
 - Algorithm 3 solves problems of size N by solving two sub-problems of size $N/2$ and performing a linear amount (i.e., cN where c is some constant) of extra work

For each algorithm, write down a recurrence relation showing how $T(N)$, the running time on an instance of size N , depends on the running time of a smaller instance. Solve $T(N)$ to obtain a closed formula.

Algorithm 1: $T(N) =$

Algorithm 2: $T(N) =$

Algorithm 3: $T(N) =$

6. Memory Analysis

Suppose that a Java library `NodeList` is implemented using an array of `Nodes`.

```
public class NodeList<Item> {  
    private Node<Item>[] list;  
    private int N; // number of items in the list  
    private class Node {  
private int count;  
        private Item item; // the item  
        private Node next, prev; // the next and previous nodes  
    }  
    ...  
}
```

Using the 64-bit memory cost model from the textbook, how much memory (in bytes) does a `Node` object use and how much does a `LinkedList` object use to store N items? Do not include the memory for the items themselves but do include the memory for the references to them.

(a) Memory of a node

(b) Memory of a `LinkedList` with N nodes.

7. Percolation Assignment

The first programming assignment is to write a program to estimate the value of the percolation threshold via Monte Carlo simulation.

- (a) What is percolation and how can Union-Find be used to simulate a percolating system?
- (b) One of the expensive operations in percolation assignment is to see if a bottom site is connected to the top site. Suggest a way to this efficiently.
- (c) Study the methods to be implemented in the Percolation class

```
public class Percolation {  
    public Percolation(int N)  
    public void open(int row, int col)  
    public boolean isOpen(int row, int col)  
    public boolean isFull(int row, int col)  
    public int numberOfOpenSites()  
    public boolean percolates()  
}
```

- (d) WeightedQuickUnionUF is a given class. What is the runtime complexity of WeightedQuickUnionUF methods, union and find?
- (e) Discuss the assignment deliverables, Percolation.java and PercolationStats.java and readme.txt files. More specifically discuss readme.txt