ComS 311 PA #2 Report

By Christian Shinkle, Alec Harrison, and Benjamin Trettin

1. Data Structures used for Q and visited
    a. We used a linked list to implement our queue. This will give us constant time for removing elements from the queue and constant time for inserting elements into the queue.
    b. We used a hash map for our visited because we needed constant time for checking if it already contained a vertex and constant time for insertion.
2. Number of edges and vertices in the graph WikiCS.txt
    a. 949 Edges
    b. 100 Vertices
3. Top 10 nodes by mostInfluentialDegree
    a. /wiki/Computer_Science
    b. /wiki/Timeline_of_computing_hardware_before_1950
    c. /wiki/History_of_computing_hardware
    d. /wiki/History_of_Unix
    e. /wiki/History_of_the_World_Wide_Web
    f. /wiki/History_of_computer_hardware_in_Yugoslavia
    g. /wiki/Digital_computer
    h. /wiki/Computer
    i. /wiki/Computing_technology
    j. /wiki/Computing
    k. Their influence is: 55.0
4. Top 10 nodes by mostInfluentialModular
    a. /wiki/Computer_Science
    b. /wiki/Timeline_of_computing_hardware_before_1950
    c. /wiki/History_of_computing_hardware
    d. /wiki/History_of_Unix
    e. /wiki/Computing_technology
    f. /wiki/History_of_the_World_Wide_Web
    g. /wiki/Computing
    h. /wiki/List_of_pioneers_in_computer_science
    i. /wiki/History_of_computer_hardware_in_Yugoslavia
    j. /wiki/Computer_graphics_(computer_science)
    k. Their influence is: 55.0

5. Top 10 nodes by mostInfluentialSubModular
    a. /wiki/Computer_Science
    b. /wiki/Computer
    c. /wiki/Computation
    d. /wiki/Procedure_(computer_science)
    e. /wiki/Algorithm
    f. /wiki/Computer_scientist
    g. /wiki/Glossary_of_computer_science
    h. /wiki/Practical_disciplines
    i. /wiki/Computational_complexity_theory
    j. /wiki/Computational_problem
    k. Their influence is: 55.0
6. Pseudo code for constructor and public method
    a. Constructor

```
Reads from the file
Sets the first part to number of nodes
While(file.has nextline) {
        Scans the line parsing by spaces to find the to and from node
        Adds both to hashmap if they're unique
                Keeps track of edges from each node puts them
                in a list and name of node
}
Close the file
```

    b. Out degree

```
Checks the hashmap for the given string and returns the arraylist's size
Return size
```

    c. Shortest path

```
Starts at the first given node
        While(Has outgoing node){
                Finds the distance from first string to second by checking all out
                vertex combinations
                Adds every node in that optimum path to an arraylist
Return arraylist
```

    d. Distance(string, string)

```
Calls shortest path with the 2 given strings
```

```
Sets int  distance to shortestpath.size - 1
If( distance ==0)
    Return -1; //Path doesn’t exist
Else
    Return distance;
```

e. Distance(ArrayList, string)

```
Checks the distance for every string of Arraylist[i] to string
If(arraylist[i]<CurrentLowest)
Update CurrentLowest

If(minDist ==Int max) // No path found
Return -1
Else
Return minDist
```

f. Influence(string)

```
Make a MST
Check every node off of the main node
Keep a running total and does (1/(2^distanceawayfromS)*number of nodes at the
Distance)
Returns running total;
```

g. Influence(ArrayList)

```
Constructs a hashmap of every node in the given array list

For every node read in from the file that is not in the given array list
    Call distance giving the given arraylist and the current node not in the list
    Uses that distance to calculate (1/(2^distanceawayfromS)
    Adds to running total of influence
Adds given arraylist.size() to running total
Returns running total
```

h. MostinfluentalDegree(int k)

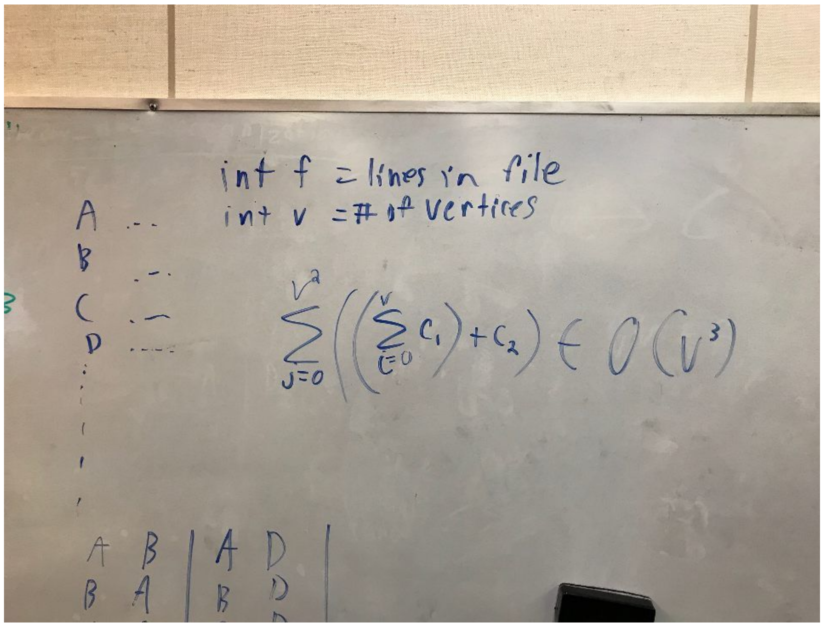
```
For every node given find the number of outdegrees for each
Use data and our tuple to store the outdegree and construct a max heap of tuples
Take off the top k elements (Fixing the tree between grab and removes) and
adding them to an arraylist

Return that arraylist
```

i. MostinfluentalModular(int k)

    j. MostinfluentalSubModular(int k)

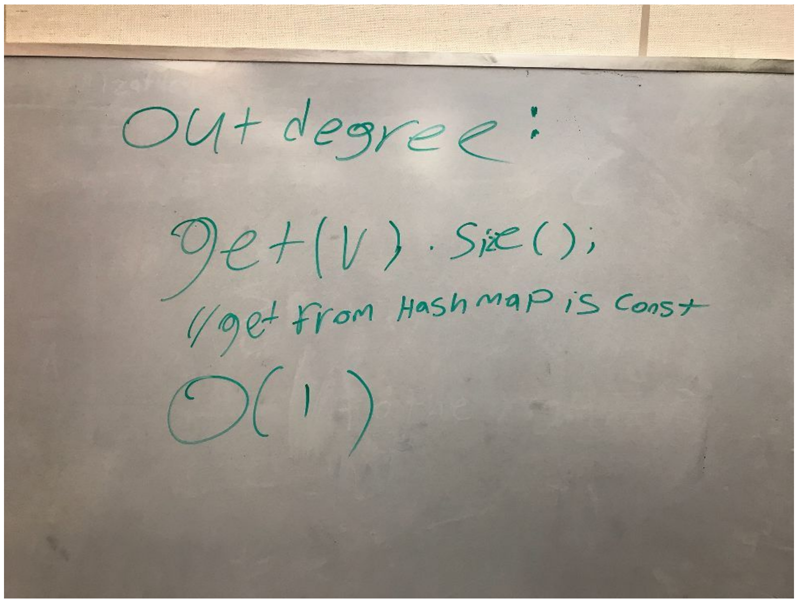
7. Analyze and report runtimes for constructor and each method
    a. Constructor



int f = lines in file
int v = # of vertices

$$\sum_{j=0}^{v^2}\left(\left(\sum_{i=0}^{v} c_1\right)+c_2\right) \in O(v^3)$$

    b. Out degree



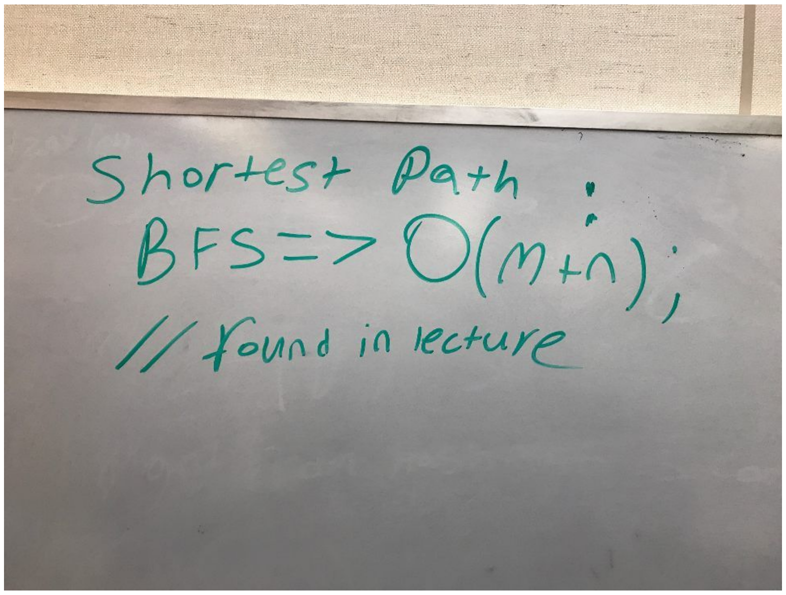
out degree:

get(V).Size();

//get from HashMap is Const

$O(1)$

    c. Shortest path

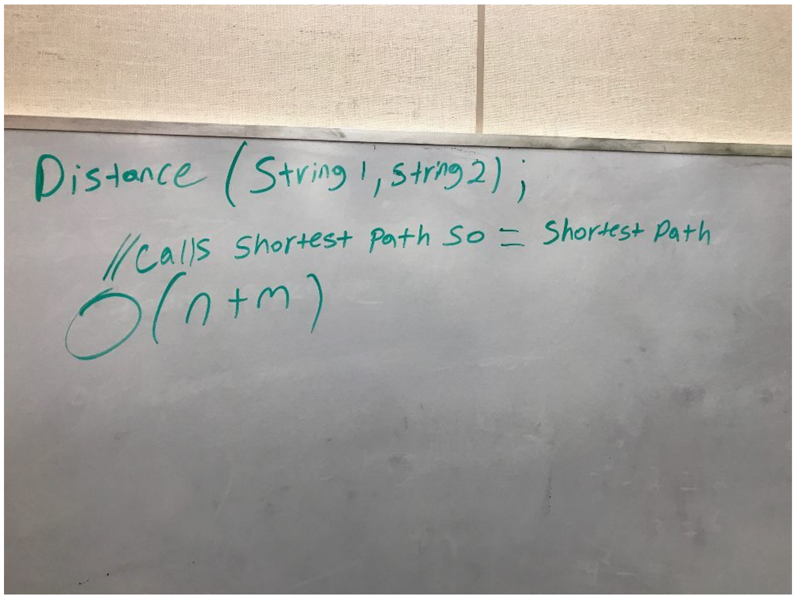


Shortest Path :

BFS => $O(m+n)$;

// found in lecture

d. Distance(string, string)

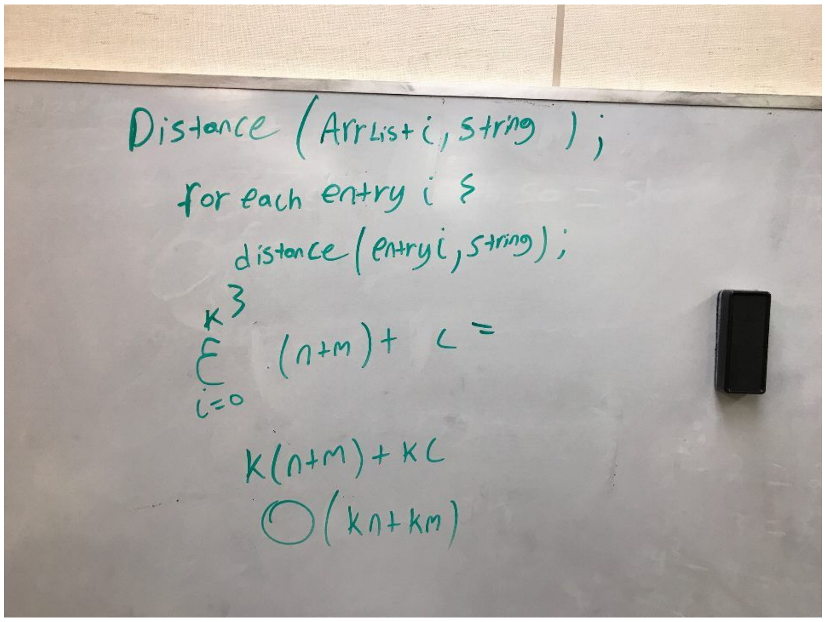


Distance (String1, String2);

//calls Shortest Path so = Shortest Path

$O(n+m)$

e. Distance(ArrayList, string)



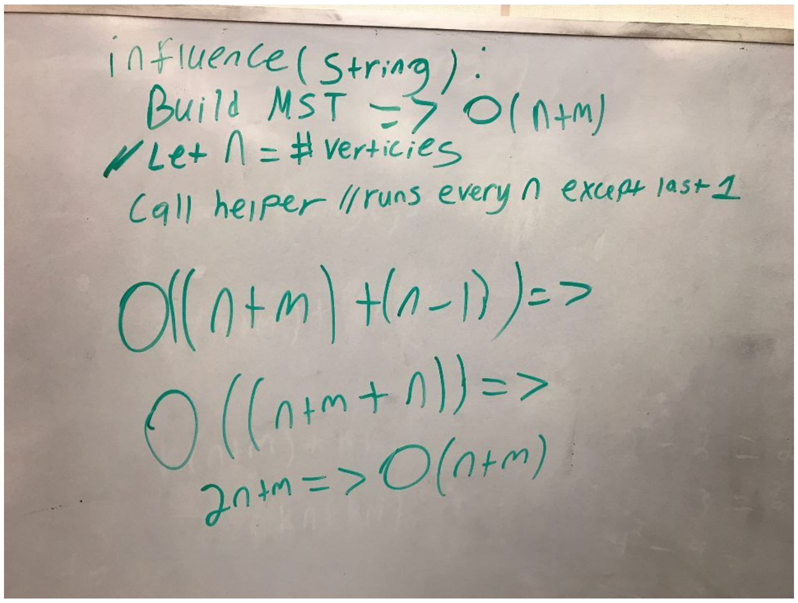
```
Distance (ArrList i, String );
  for each entry i {
    distance(entry i, String);
  }
```

$$\sum_{i=0}^{k} (n+m) + c =$$

$$k(n+m) + kc$$

$$O(kn+km)$$

f. Influence(string)



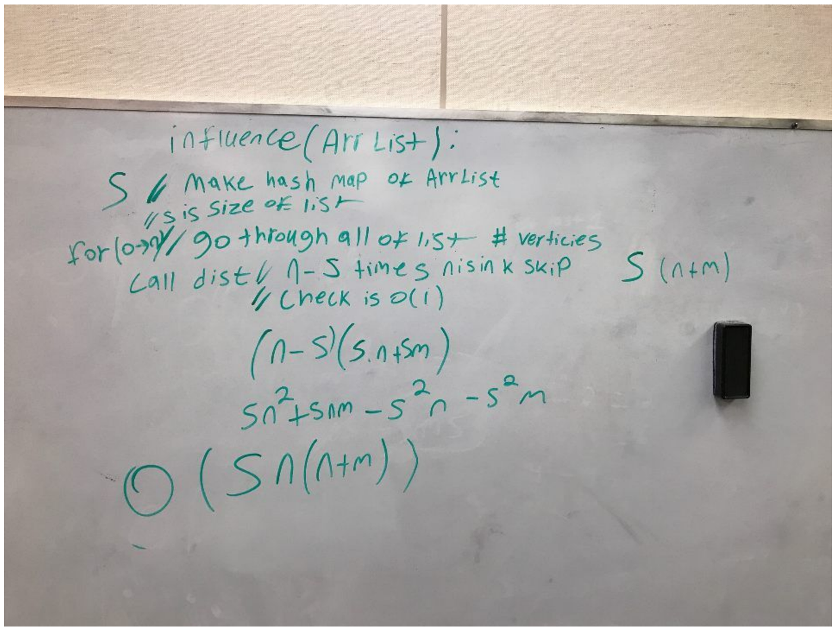
```
influence(String):
  Build MST => O(n+m)
  // Let n = # verticies
  Call helper //runs every n except last 1
```

$$O((n+m)+(n-1))=>$$

$$O((n+m+n))=>$$

$$2n+m => O(n+m)$$

g. Influence(Arraylist)



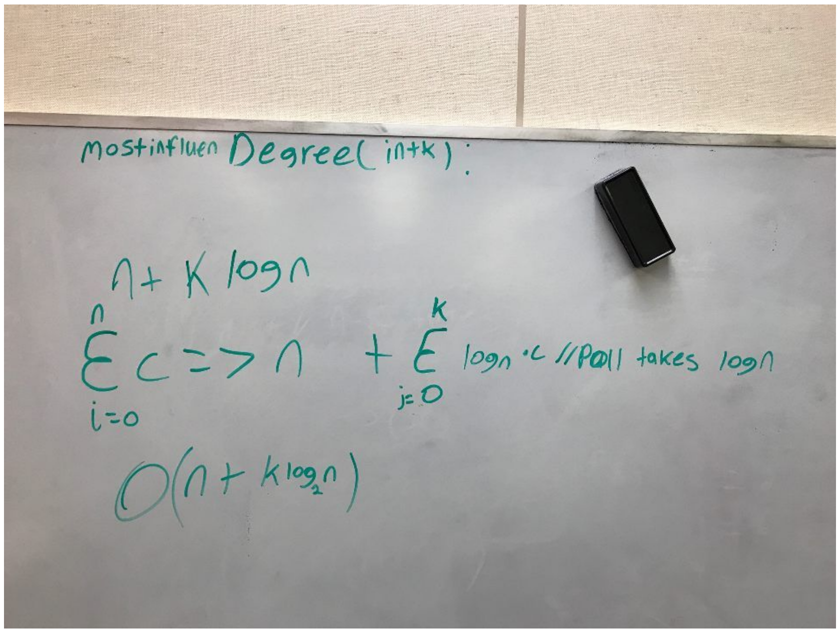
```
influence(ArrList):
S // make hash map of ArrList
  //s is size of list
for(0->n)// go through all of list  # verticies
  call dist// n-S times n is in k skip        S(n+m)
    // check is o(1)
```

$(n-S)(S\cdot n+Sm)$

$Sn^2+Snm-S^2n-S^2m$

$O(Sn(n+m))$

h. MostinfluentalDegree(int k)



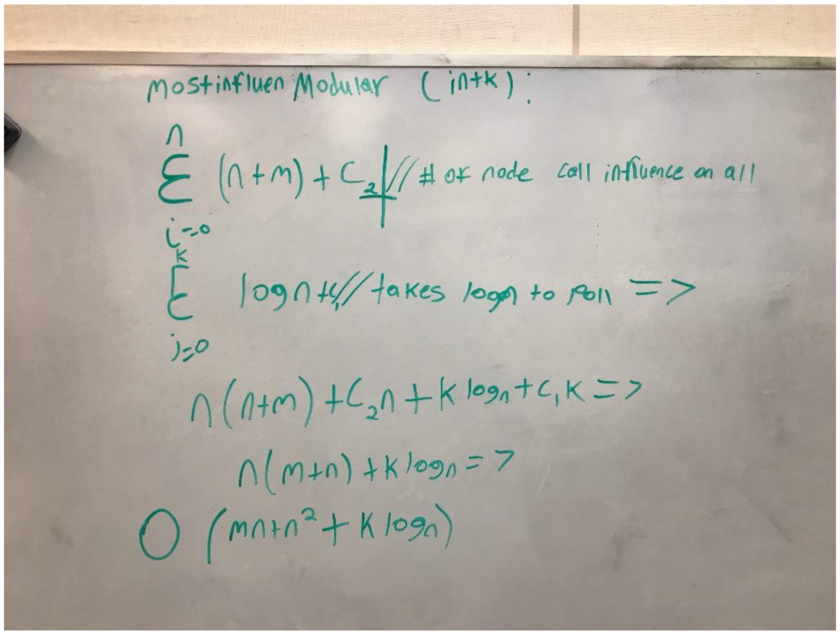
mostinfluen Degree(int k):

$n+K\log n$

$\sum_{i=0}^{n} c => n + \sum_{j=0}^{k} \log n \cdot c$ //Poll takes logn

$O(n+k\log_2 n)$

i. MostinfluentalModular(int k)



mostinfluen Modular (int k):

$\sum_{i=0}^{n} (n+m) + C_2$ // # of node call influence on all

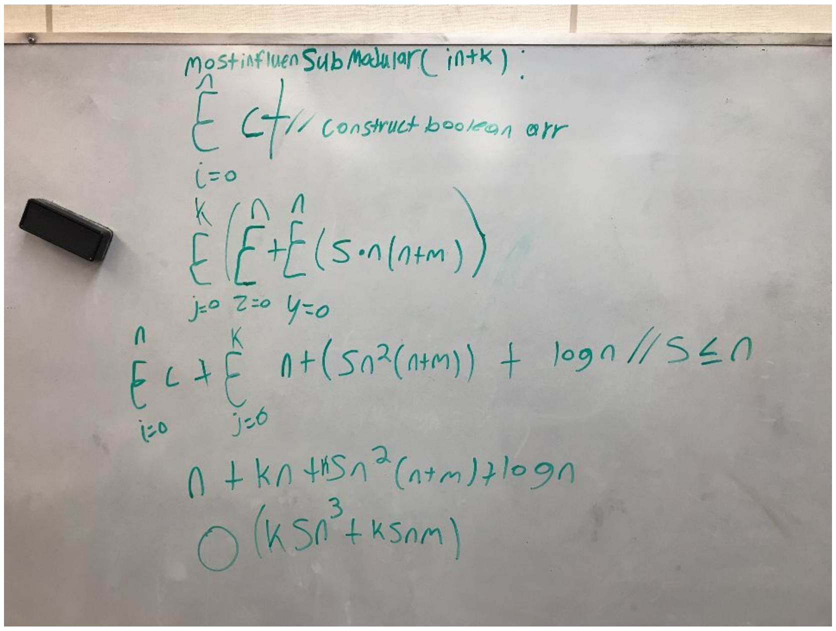
$\sum_{j=0}^{k} \log n + c_1$ // takes logn to poll $\Rightarrow$

$n(n+m) + C_2 n + k \log n + c_1 k \Rightarrow$

$n(m+n) + k \log n \Rightarrow$

$O(mn + n^2 + k \log n)$

j. MostinfluentalSubModular(int k)



mostinfluen SubModular(int k):

$\sum_{i=0}^{n} c$ // construct boolean arr

$\sum_{j=0}^{k} \left(\sum_{z=0}^{n} + \sum_{y=0}^{n} (S \cdot n(n+m)) \right)$

$\sum_{i=0}^{n} c + \sum_{j=0}^{k} n + (Sn^2(n+m)) + \log n$ // $S \leq n$

$n + kn + kSn^2(n+m) + \log n$

$O(kSn^3 + kSnm)$