REPORT on Minor I

Submitted by

Ojasvi Singh Chauhan (Enroll: R134218111)

Shrishty Dayal (Enroll: R134218158)

Under the guidance of

Ms. Deepa Joshi

Assistant Professor
Department of Systemics

SCHOOL OF COMPUTER SCIENCE
UNIVERSITY OF PETROLEUM & ENERGY STUDIES
Bidholi Campus, Energy Acres, Dehradun – 248007.
August – December(2020)

School of Computer Science

University of Petroleum & Energy Studies, Dehradun

PROJECT TITLE: Percolation

ABSTRACT

The theory of percolation was first introduced some time ago to describe polymerization and penetration of fluids in porous media, and the subject has since been intensively studied, primarily in the field of physics. However, little direct use of percolation theory results has been made to date in the field of hydrology. The theory has been extensively developed as a branch of statistical physics and has found successful applications in a diverse range of problems including the design of electronic and magnetic materials, conceptualization of geometrical and topological characteristics of porous media, and understanding of miscible and immiscible displacements in discarded media. The principal advantage of percolation theory is that it provides universal laws which determine the geometrical and physical properties of a system. These laws are independent of the local geometry or configuration of the system. In particular, many transport processes can be successfully understood by considering an idealized transport of an abstract fluid through an abstract medium which is itself in some sense disordered(or random), the flow through the system may be described by a so-called percolation process.

KEYWORDS: porous media; percolation; hydrology; statistical physics; universal laws

INTRODUCTION

Percolation is the process of a liquid moving slowly through a substance that has very small holes in it, or in other words, the process of something spreading slowly [1]. The theory of Percolation was originally proposed over 30 years ago to describe flow phenomena in porous media. In this project, prime goal would be on the percolation of water. Because when there will be a procedure to know whether water will percolate or not, then all the geological activities can be done accordingly. Like Drainage Systems can be built according to the results obtained from percolation theory, sewer systems can be maintained according to Percolation results. Similarly in other fields also, Conductors and Insulators can be made according to the answers coming from our percolation model.

As all these problems are dependent on this Percolation model, so main focus has been on building this model and how to calculate the probability of percolation of water from top to bottom, like what are the chances of water started pouring from the top and end up coming out from the bottom of the surface. That model is being created by creating an N-by-N grid of squares that is known as sites. Classical percolation theory studies site and bond percolations. In a graph model, the site is represented as the vertex while the bond is represented as the edge. In a grid structure, each cell is a site and the bond is an edge between cells. By assigning each site/bond with a probability p of being occupied/open, the major interest is in whether it is possible to find a path from the leftmost to the rightmost or from the top to the bottom [2].

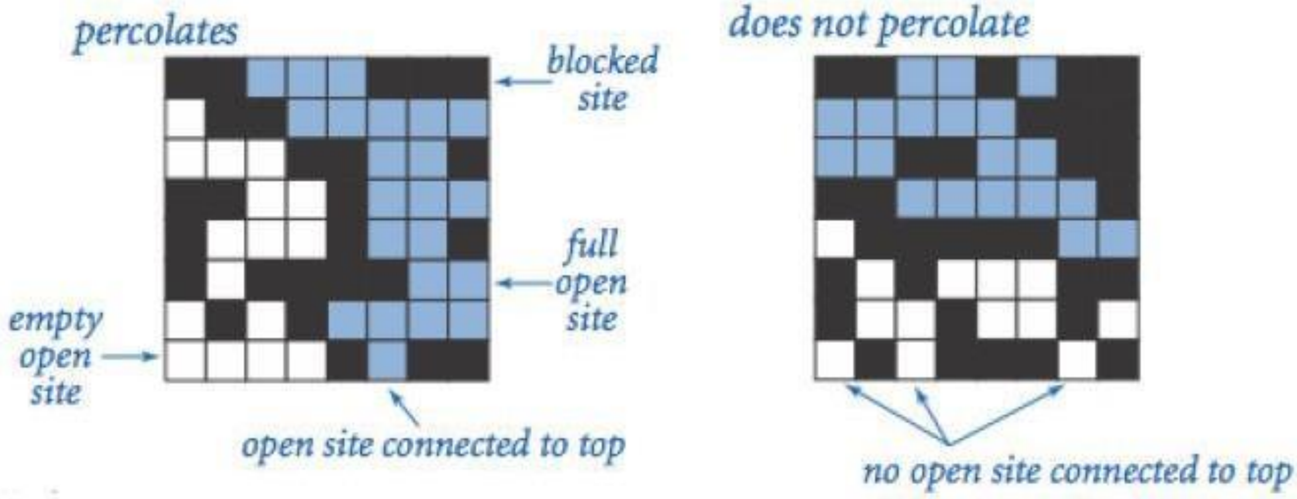


Fig 1 Percolation in 8-by-8 grid system [3]

If the sites are independently set to be open with probability p (and therefore blocked with probability 1 - p), what is the probability that the system percolates? When p equals 0, the system does not percolate; when p equals 1, the system percolates. The plots below show the site vacancy probability p versus the percolation probability for a 20-by-20 random grid (left) and 100-by-100 random grid (right).

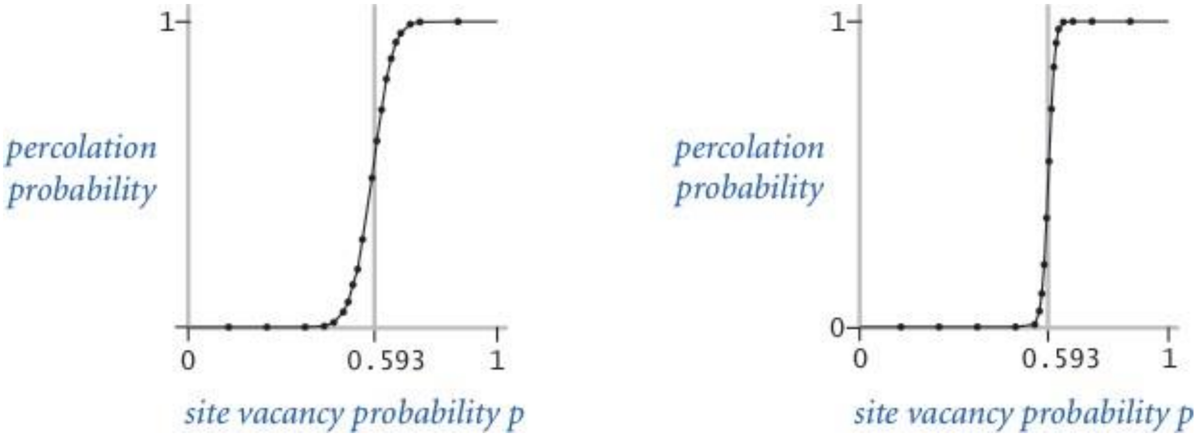


Fig 2 Probability distribution graph [4]

When N is sufficiently large, there is a threshold value p* such that when p < p* a random N-by-N grid almost never percolates, and when p > p*, a random N-by-N grid almost always percolates. No mathematical solution for determining the percolation threshold p* has yet been derived. So, our task is to write a computer program to estimate p* [5].

PROBLEM STATEMENT

What fraction of the material need to be metallic so that the composite system is an electrical conductor. What fraction of the ground should be used to lay the pipes to ensure the water supply with the minimum effort? In the social network where it's people connected and either there's a connection between two people or not.

REAL LIFE EXAMPLES

1. In the field of hydrology, Percolation can be used in making drainage systems, sewer systems, in irrigation of farms, etc.

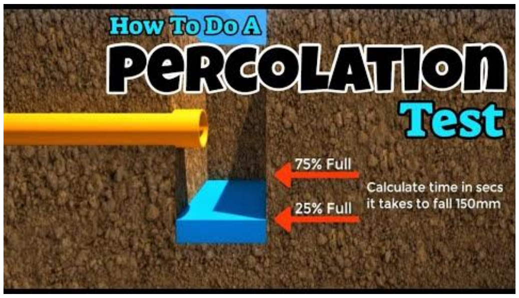


Fig 3: Percolation in soil

2. In the field of Electricity, percolation can be used to make conductors and semi conductors.

   Metal cost can be reduced by ensuring the right amount of usage of metals in a substance.

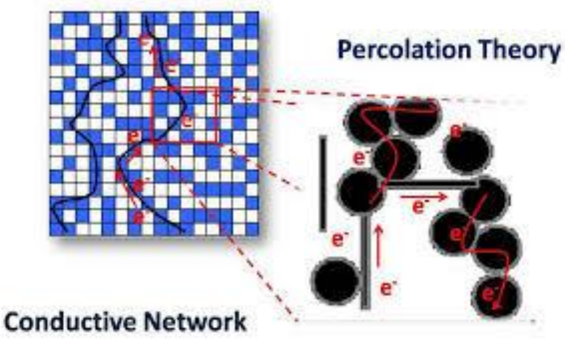


Fig 4: Percolation in Conductors

3. In Social networking sites like Facebook, Linkedin, etc. we can apply this algorithm to check whether person A is connected to person B or not.

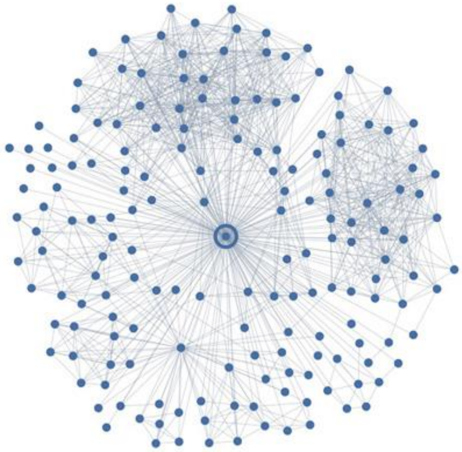


Fig 5: Connections in Social Media

LITERATURE REVIEW

Title	Link	Author	Remarks
Percolation Theory and Its Application to Groundwater Hydrology	[6]	Brian Berkovitz	The methods of percolation theory are discussed, previous applications of the theory to hydrological problems are reviewed, and future directions of study are suggested.
The Union-Find problem is Linear	[7]	Hantao Zhang	This paper work has presented a new proof which shows that the complexity of the union-find problem is linear.
Data Structures and Algorithms for Disjoint Set Union Problems	[8]	Galil, Zvi & Italiano, Giuseppe	Explained the Data Structure for disjoint set union find problem. They showed that union-find visualization is done on a tree but its actual implementation is done on an array.
Theory of Percolation	[9]	Swami Iyer	In his theory he basically talked about how to calculate probability threshold using Monte Carlo simulation and various mathematical calculations

OBJECTIVES

To create an application that will recognize whether the system will percolate or not.

Sub Objectives:

- To detect the connected components of water inflow.
- To calculate the threshold and probability of percolation.
- To create a visualizer for communicating with grid.

METHODOLOGY

For this project, we will use the applications of **Union & Find** data structures to achieve the goal. In this model, we would have to create a grid, which we'll build by using graphs. We'll use vertices of graphs to represent sites and edges to represent the connection between sites. So, Union & Find data structures also known as Disjoint-set-data-structure is a data structure that keeps track of a set of elements partitioned into a number of disjoint (non-overlapping) subsets. A union-find algorithm is an algorithm that performs two useful operations on such a data structure:

Find: Determine which subset a particular element is in. This can be used for determining if two elements are in the same subset.

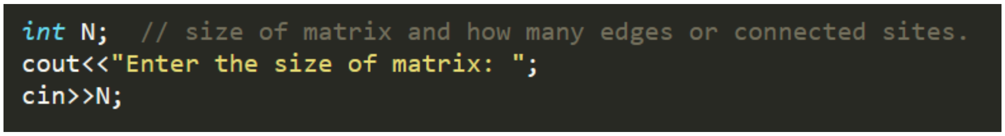
Union: Join two subsets into a single subset.

There are many applications of a union-find algorithm like whether a graph forms a cycle or not, whether a system will percolate or not, etc. So here we'll use union-find API to obtain the connected components of a graph and hence the connected open sites.
The entire implementation of this project can be summarized in the following steps:

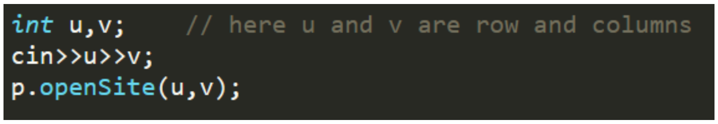
1. **Create an API for the union-find algorithm:** Firstly we have to write the unionSet() and findSet() functions in a different .cpp file to ensure their usage afterward.

2. **Optimize the Algorithm:** Then we will optimize our Union-Find Algorithm by using mainly two optimizations i.e. Path Compression and Optimization by Rank. Without any optimization union-find works in O(n) time, but after path compression, its complexity will get reduced to O(logn) and after the second optimization, its complexity will further get reduced to nearly a constant time or we can say nearly O(1).

3. **Analyze the Grid:** Then we have to create an N-by-N matrix by the use of undirected graphs. We will connect its vertices as per the requirement of input.

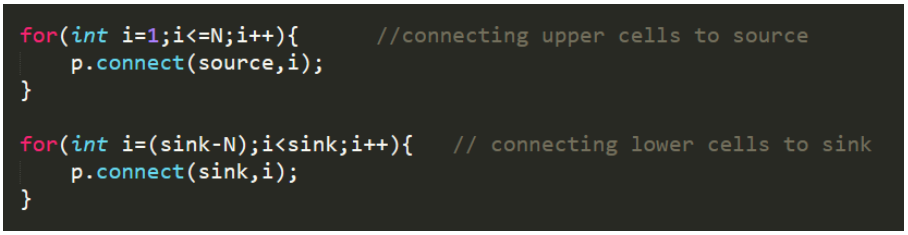


```
int N;  // size of matrix and how many edges or connected sites.
cout<<"Enter the size of matrix: ";
cin>>N;
```



```
int u,v;    // here u and v are row and columns
cin>>u>>v;
p.openSite(u,v);
```

4. **Virtual Top Site:** We have to create two extra sites (vertices), one above the matrix, which will act as the start, and one below the matrix, which will act as a sink. Then we'll connect all the cells of the top row with the start site and all the cells of the most bottom row with the sink site.



```
for(int i=1;i<=N;i++){      //connecting upper cells to source
    p.connect(source,i);
}

for(int i=(sink-N);i<sink;i++){   // connecting lower cells to sink
    p.connect(sink,i);
}
```

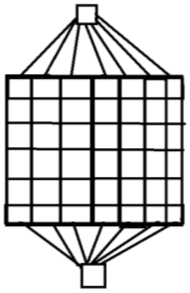
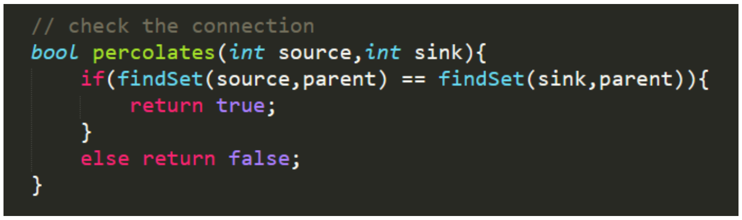


Fig 6: Virtual top site

5. **Avoid backwash:** After all these preparations, we have to avoid the backflow problem. Because it might also happen that when water will reach the sink, it might also flow back inside the grid again through other open sites at the bottom row. Therefore avoidance of that problem is also necessary to maintain the correctness in the probability and threshold of the system.

6. **Check the connection:** For checking the connection, we have to check whether findSet(start) is equal to findSet(sink). If return value of both these function calls is same, then it means that the start is connected with sink through open sites and hence the system will percolate.



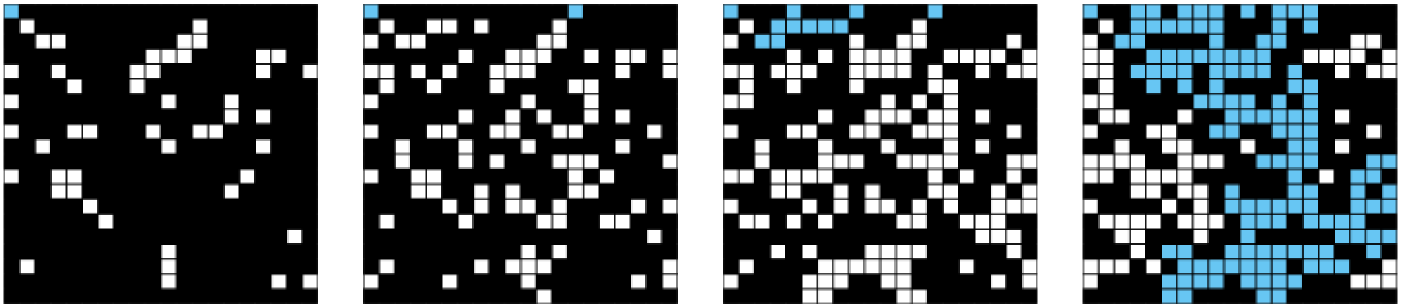
```
// check the connection
bool percolates(int source,int sink){
    if(findSet(source,parent) == findSet(sink,parent)){
        return true;
    }
    else return false;
}
```

7. **Monte Carlo Simulation:** Monte Carlo methods, or Monte Carlo experiments, are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle [10].

- Initialize all sites to be blocked.
- Repeat the following until the system percolates:
  -- Choose a site (row i, column j) uniformly at random among all blocked sites.
  -- Open the site (row i, column j).

- The fraction of sites that are opened when the system percolates provides an estimate of the percolation threshold.

For example, if sites are opened in a 20-by-20 grid according to the snapshots below, then our estimate of the percolation threshold is 204/400 = 0.51 because the system percolates when the 204th site is opened.



50 open sites　*100 open sites*　*150 open sites*　*204 open sites*

Fig 7 Site distribution at various time[11]

By repeating this computational experiment T times and averaging the results, we obtain a more accurate estimate of the percolation threshold. Let x_t be the fraction of open sites in computational experiment t. The sample mean μ provides an estimate of the percolation threshold, and the sample standard deviation σ measures the sharpness of the threshold:

$$\mu = \frac{x_1 + x_2 + \cdots + x_T}{T}, \qquad \sigma^2 = \frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + \cdots + (x_T - \mu)^2}{T - 1}.$$

Assuming T is sufficiently large (say, at least 30), the following provides a 95% confidence interval for the percolation threshold:

$$\left[\mu - \frac{1.96\sigma}{\sqrt{T}}, \mu + \frac{1.96\sigma}{\sqrt{T}}\right].$$

SYSTEM REQUIREMENTS

Hardware:

- RAM: 4GB
- Disk Space: 500 MB

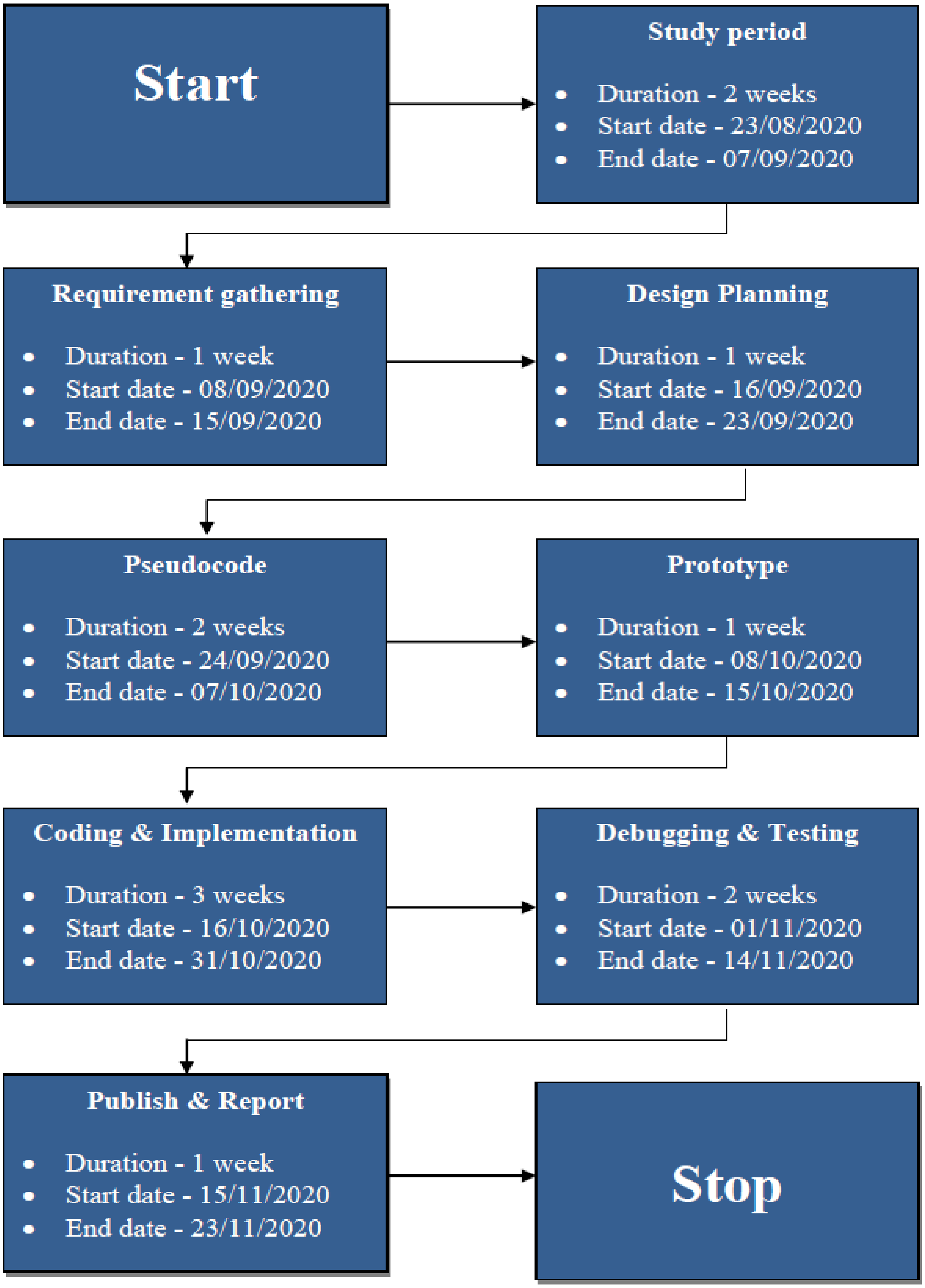
Software:

- Sublime Text(or any other C++ IDE)
- Gcc Compiler should be installed.

Operating System:

- Windows or Linux.

SCHEDULE (pert chart)



Start

Study period

- Duration - 2 weeks
- Start date - 23/08/2020
- End date - 07/09/2020

Requirement gathering

- Duration - 1 week
- Start date - 08/09/2020
- End date - 15/09/2020

Design Planning

- Duration - 1 week
- Start date - 16/09/2020
- End date - 23/09/2020

Pseudocode

- Duration - 2 weeks
- Start date - 24/09/2020
- End date - 07/10/2020

Prototype

- Duration - 1 week
- Start date - 08/10/2020
- End date - 15/10/2020

Coding & Implementation

- Duration - 3 weeks
- Start date - 16/10/2020
- End date - 31/10/2020

Debugging & Testing

- Duration - 2 weeks
- Start date - 01/11/2020
- End date - 14/11/2020

Publish & Report

- Duration - 1 week
- Start date - 15/11/2020
- End date - 23/11/2020

Stop

Fig 8 Pert Chart

PROJECT PROGRESS

Task Achieved:

1. Requirement Gathering.
2. Design Planning
3. Pseudo Code

We have written Unionset and Findset functions till now:

```
#include <bits/stdc++.h>
using namespace std;

class Graph{
    int V;
    list<pair<int,int>> l;

    public:
        Graph(int V){
            this->V = V;
        }

        void addEdge(int u,int v){
            l.push_back(make_pair(u,v));
        }

        int findSet(int i,int parent[]){
            if(parent[i]==-1){
                return i;
            }
            return parent[i] = findSet(parent[i],parent);
        }

        void unionSet(int x,int y,int parent[],int rank[]){
            int s1 = findSet(x,parent);
            int s2 = findSet(y,parent);

            if(s1!=s2){
                if(rank[s1]<rank[s2]){
                    parent[s1] = s2;
                    rank[s2] += rank[s1];
                }
                else{
                    parent[s2] = s1;
                    rank[s1] += rank[s2];
                }
            }
        }
};
```

```
int main(){
    int V;
    cin>>V;

    int *parent = new int[V];
    int *rank = new int[V];

    for(int i=0;i<V;i++){
        parent[i] = -1;
        rank[i] = 1;
    }

    Graph g(V);
    for(int i=0;i<V;i++){
        int u,v;
        cin>>u>>v;
        g.addEdge(u,v);
    }
    cout<<g.findSet(1,parent)<<endl;

return 0;
}
```

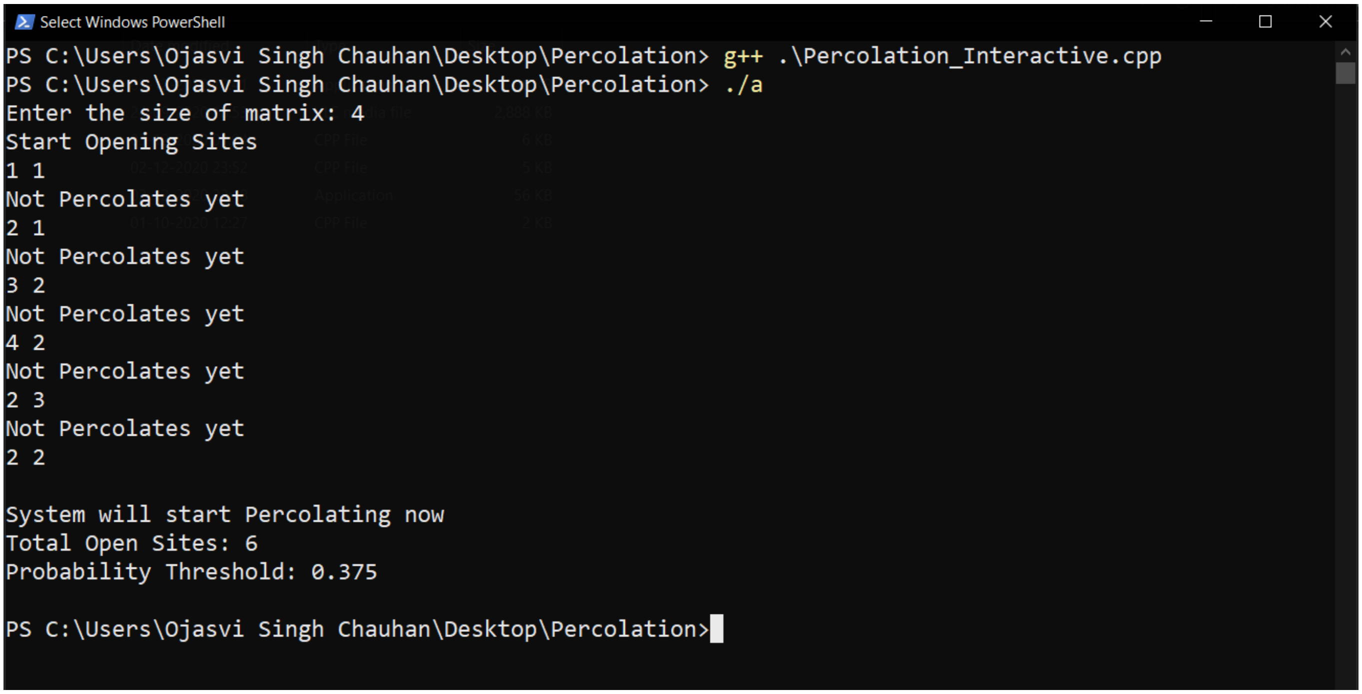
Progress after Mid Sem Presentation:

4. Code and Implementation.
5. Debugging and Testing.

Here is the main interactive code for our Percolation Model:

```
#include <bits/stdc++.h>
using namespace std;

class Percolation{
    int n;
    bool *open;
    int *parent;
    int *rank;
    int N;

    public:
        int openCount;
        Percolation(int N){
            if(N<=0){
                cout<<"Grid size should be 1 or more";
                exit(1);
            }
            n = N*N;        // Total Cells
            this->N = N;    // Number of rows and columns


            // this array will store whether any cell is open or close
            open = new bool[n+2];
            for(int i=0;i<n+2;i++){
                // initially all cells are closed
                open[i]=false;
            }

            // but source and sink will remain open
            open[0] = true;
            open[n+1] = true;

            // initialization for union & find
            parent = new int[n+2];
            rank = new int[n+2];
            for(int i=0;i<n+2;i++){
                parent[i] = -1;
                rank[i] = 1;
            }

            openCount=0;
        }


        int findSet(int i,int parent[]){
            if(parent[i]==-1){
                return i;
            }

            return parent[i] = findSet(parent[i],parent);
        }

        void unionSet(int x,int y,int parent[],int rank[]){
            int s1 = findSet(x,parent);
            int s2 = findSet(y,parent);

            if(s1!=s2){
                if(rank[s1]<rank[s2]){
                    parent[s1] = s2;
                    rank[s2] += rank[s1];
                }
                else{
                    parent[s2] = s1;
                    rank[s1] += rank[s2];
                }
            }
        }

        void connect(int a,int b){
            int s1 = findSet(a,parent);         //O(1)
            int s2 = findSet(b,parent);         //O(1)

            if(s1 != s2){
                unionSet(s1,s2,parent,rank);    //O(1)
            }

            // for(int i=0;i<n+2;i++){
            //     cout<<parent[i]<<" ";
            // }
            // cout<<endl;
        }

        bool validate(int m) {
            if (m < 0 || m > n){
                cout<< "Incorrect location supplied!\nPlease Re-enter the
cell\n";
                return false;
            }
            return true;
        }

        int getIndex(int i,int j){
            return (i-1)*N + j;
        }

        void openSite(int row,int col){
            int index = getIndex(row,col);
            if(!validate(index))
                return;

            //cout<<"Index ID: "<<index<<endl;
            if(open[index])
                return;
            open[index] = true;
            openCount++;

            // for(int i=0;i<n+2;i++){
            //     cout<<open[i]<<" ";
            // }
            // cout<<endl;


            // connect to top component
            if (row > 1 && open[index - N]){
                //cout<<1<<endl;
                connect(index, index - N);
            }
            // connect to bottom component
            if (row < N && open[index + N]){
                //cout<<2<<endl;
                connect(index, index + N);
            }
            // connect to left component
            if (col > 1 && open[index - 1]){
                //cout<<3<<endl;
                connect(index, index - 1);
            }
            // connect to right component
            if (col < N && open[index + 1]){
                //cout<<4<<endl;
                connect(index, index + 1);
            }
        }

        // check the connection
        bool percolates(int source,int sink){
            if(findSet(source,parent) == findSet(sink,parent)){
                return true;
            }
            else return false;
        }
};

int main(){

    int N;               // size of matrix and how many edges or connected sites.
    cout<<"Enter the size of matrix: ";
    cin>>N;

    int source = 0;
    int sink = N*N+1;

    Percolation p(N);

    for(int i=1;i<=N;i++){      //connecting upper cells to source
        p.connect(source,i);
    }

    for(int i=(sink-N);i<sink;i++){   // connecting lower cells to sink
        p.connect(sink,i);
    }

    cout<<"Start Opening Sites\n";

    int count=0;
    while(true){       // opening sites in between

        if(count > N*N) break;  // Checking overflow condition for while loop

        int u,v;    // here u and v are row and columns
        cin>>u>>v;
        p.openSite(u,v);

        if(p.percolates(source,sink)){
            cout<<"\nSystem will start Percolating now"<<endl;
            cout<<"Total Open Sites: "<<p.openCount<<endl;
            cout<<"Probability Threshold: "<<(double)p.openCount/(N*N)<<endl;
            cout<<endl;
            break;
        }
        else{
            cout<<"Not Percolates yet\n";
        }
        count++;
    }

return 0;
}

// total time complexity O(row*col)
```

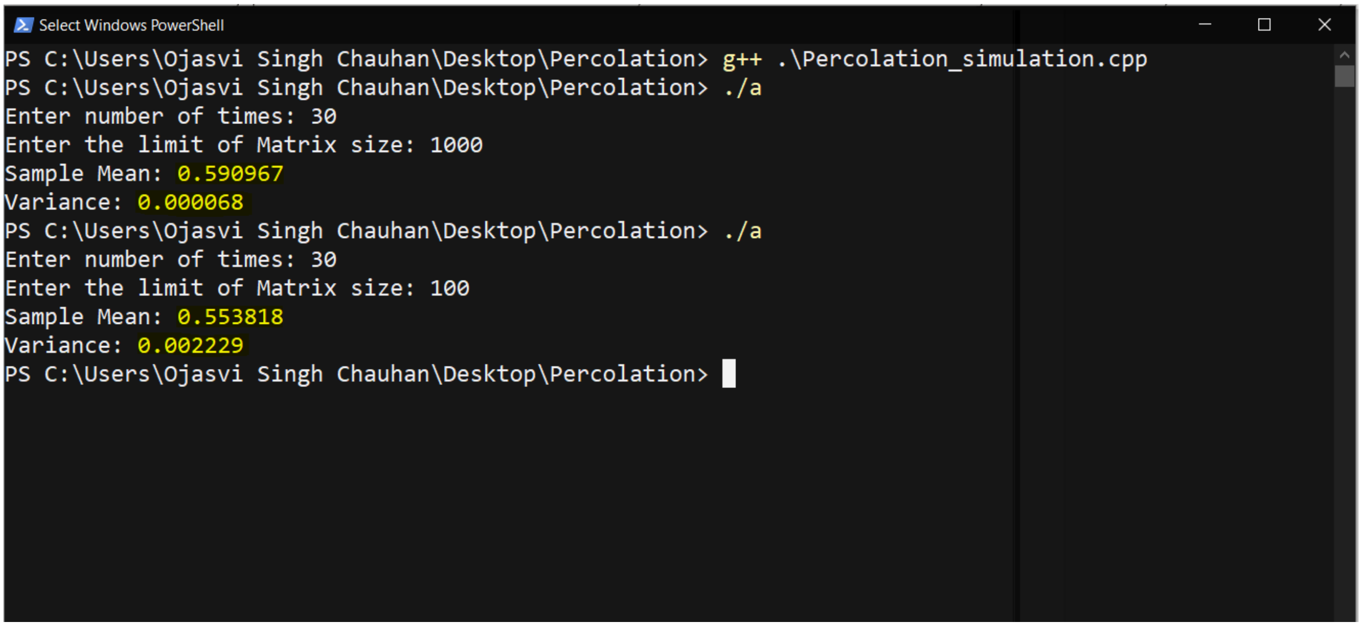
Output:



```
PS C:\Users\Ojasvi Singh Chauhan\Desktop\Percolation> g++ .\Percolation_Interactive.cpp
PS C:\Users\Ojasvi Singh Chauhan\Desktop\Percolation> ./a
Enter the size of matrix: 4
Start Opening Sites
1 1
Not Percolates yet
2 1
Not Percolates yet
3 2
Not Percolates yet
4 2
Not Percolates yet
2 3
Not Percolates yet
2 2

System will start Percolating now
Total Open Sites: 6
Probability Threshold: 0.375

PS C:\Users\Ojasvi Singh Chauhan\Desktop\Percolation>
```

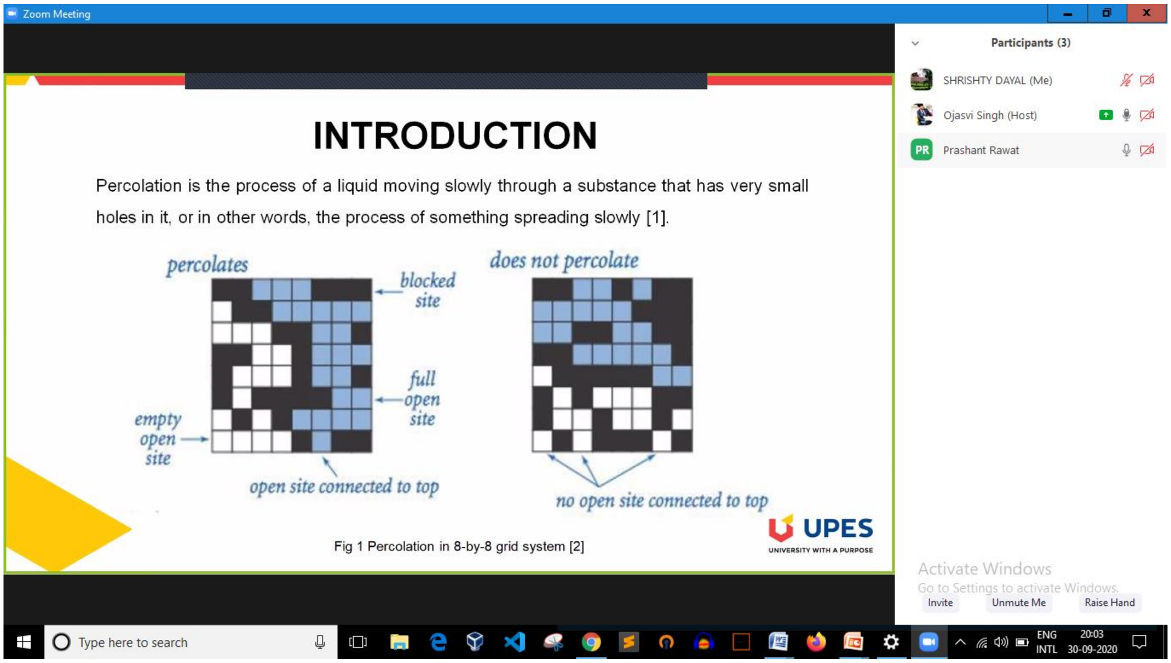
Here is the Monte Carlo Simulation Code for our Model:

```
#include <bits/stdc++.h>
using namespace std;

class Percolation{
      **same as previous code**
};

int main(){
    int times;
    cout<<"Enter number of times: ";
    cin>>times;
    int limit;
    cout<<"Enter the limit of Matrix size: ";
    cin>>limit;

    double sum=0,sum2=0;
    double mean;
    double deviation;
    vector<double> Xi;

    srand((unsigned) time(0));
    for(int i=0;i<times;i++){
        int N = 10 + (rand() % limit);
        int source = 0;
        int sink = N*N+1;
        Percolation p(N);
        for(int i=1;i<=N;i++){
            p.connect(source,i);
        }
        for(int i=(sink-N);i<sink;i++){
            p.connect(sink,i);
        }
        int count=0;
        while(true){
            if(count > N*N) break;
            int u = 1 + (rand() % N);
            int v = 1 + (rand() % N);
            p.openSite(u,v);
            if(p.percolates(source,sink)){
                double x = (double)p.openCount/(N*N);
                //cout<<x<<" ";
                Xi.push_back(x);
                sum += x;
                break;
            }
            count++;
        }
    }

    mean = sum/times;
    cout<<"Sample Mean: ";
    cout<<mean<<endl;
    //cout<<Xi.size()<<endl;
    for(auto ele: Xi){
      sum2 += (ele-mean)*(ele-mean);
    }
    deviation = sum2/(times-1);

      cout.precision(6);
      cout<<std::fixed;
      cout<<"Variance: "<<deviation<<endl;

return 0;
}
```



```
PS C:\Users\Ojasvi Singh Chauhan\Desktop\Percolation> g++ .\Percolation_simulation.cpp
PS C:\Users\Ojasvi Singh Chauhan\Desktop\Percolation> ./a
Enter number of times: 30
Enter the limit of Matrix size: 1000
Sample Mean: 0.590967
Variance: 0.000068
PS C:\Users\Ojasvi Singh Chauhan\Desktop\Percolation> ./a
Enter number of times: 30
Enter the limit of Matrix size: 100
Sample Mean: 0.553818
Variance: 0.002229
PS C:\Users\Ojasvi Singh Chauhan\Desktop\Percolation>
```

PRESENTATION SNAPSHOTS

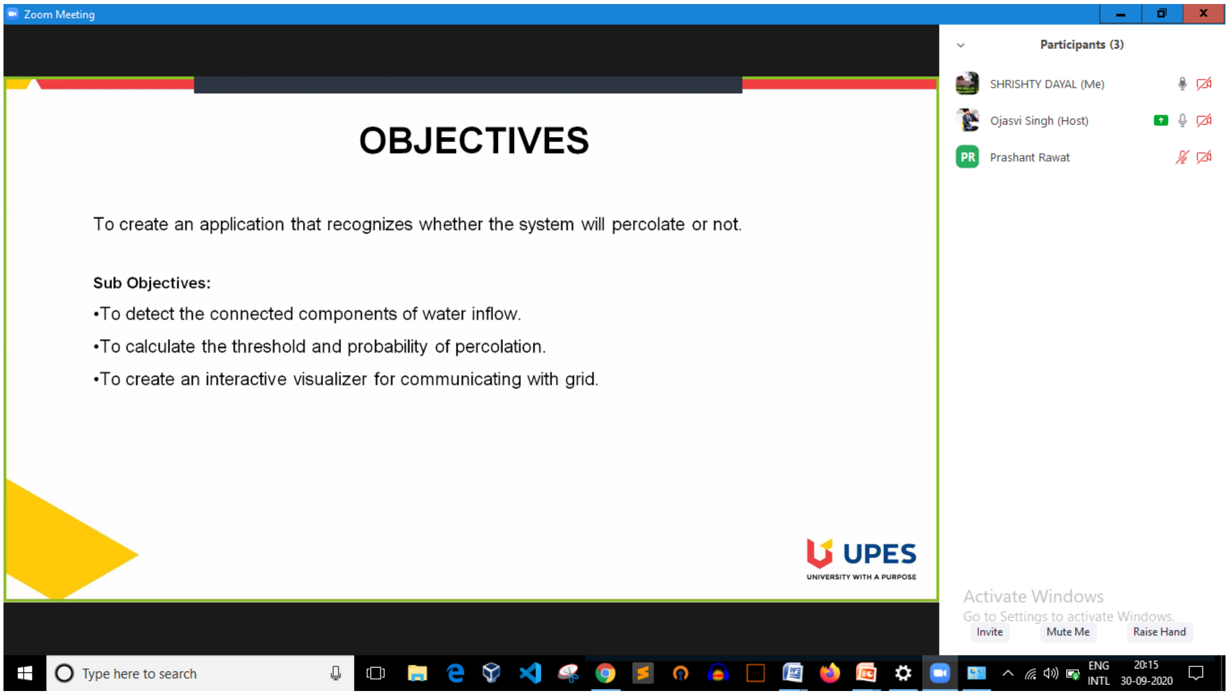


INTRODUCTION

Percolation is the process of a liquid moving slowly through a substance that has very small holes in it, or in other words, the process of something spreading slowly [1].

percolates
blocked site
full open site
empty open site
open site connected to top
does not percolate
no open site connected to top
UPES
UNIVERSITY WITH A PURPOSE

Fig 1 Percolation in 8-by-8 grid system [2]



OBJECTIVES

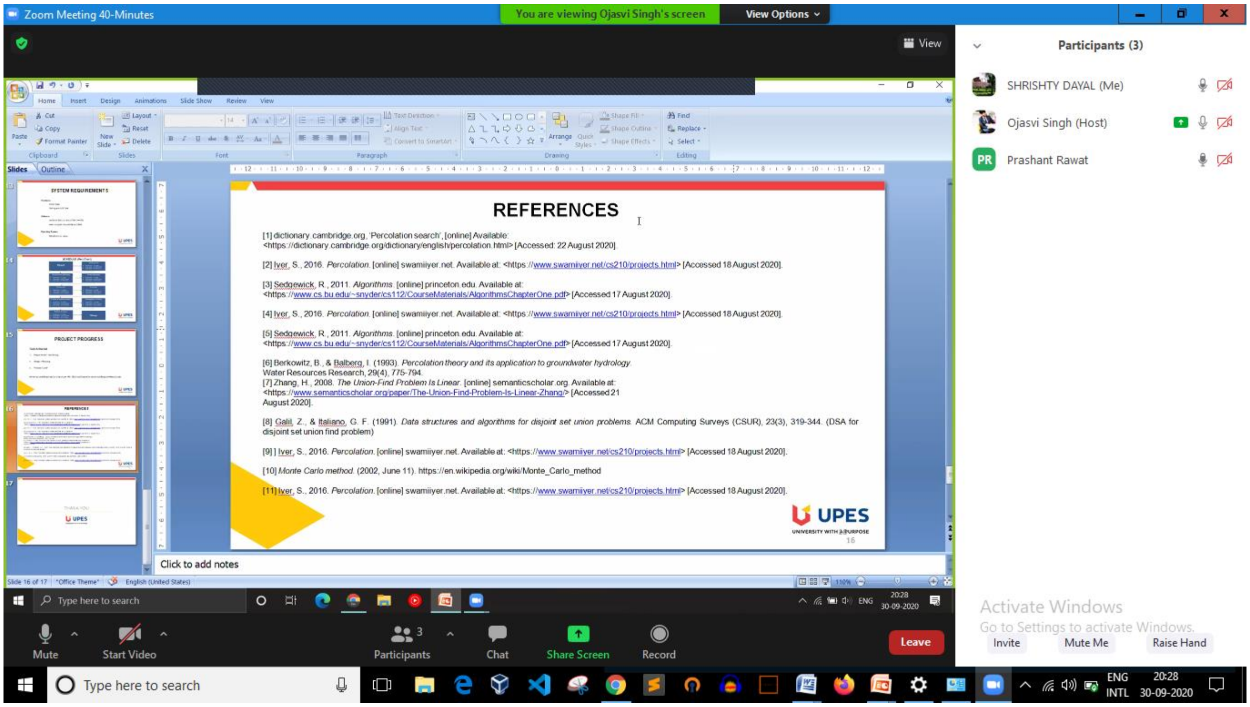
To create an application that recognizes whether the system will percolate or not.

Sub Objectives:

•To detect the connected components of water inflow.

•To calculate the threshold and probability of percolation.

•To create an interactive visualizer for communicating with grid.



REFERENCES

[1] dictionary.cambridge.org, 'Percolation search', [online] Available: <https://dictionary.cambridge.org/dictionary/english/percolation.html> [Accessed: 22 August 2020].

[2] Iyer, S., 2016. *Percolation*. [online] swamiiyer.net. Available at: <https://www.swamiiyer.net/cs210/projects.html> [Accessed 18 August 2020].

[3] Sedgewick, R., 2011. *Algorithms*. [online] princeton.edu. Available at: <https://www.cs.bu.edu/~snyder/cs112/CourseMaterials/AlgorithmsChapterOne.pdf> [Accessed 17 August 2020].

[4] Iyer, S., 2016. *Percolation*. [online] swamiiyer.net. Available at: <https://www.swamiiyer.net/cs210/projects.html> [Accessed 18 August 2020].

[5] Sedgewick, R., 2011. *Algorithms*. [online] princeton.edu. Available at: <https://www.cs.bu.edu/~snyder/cs112/CourseMaterials/AlgorithmsChapterOne.pdf> [Accessed 17 August 2020].

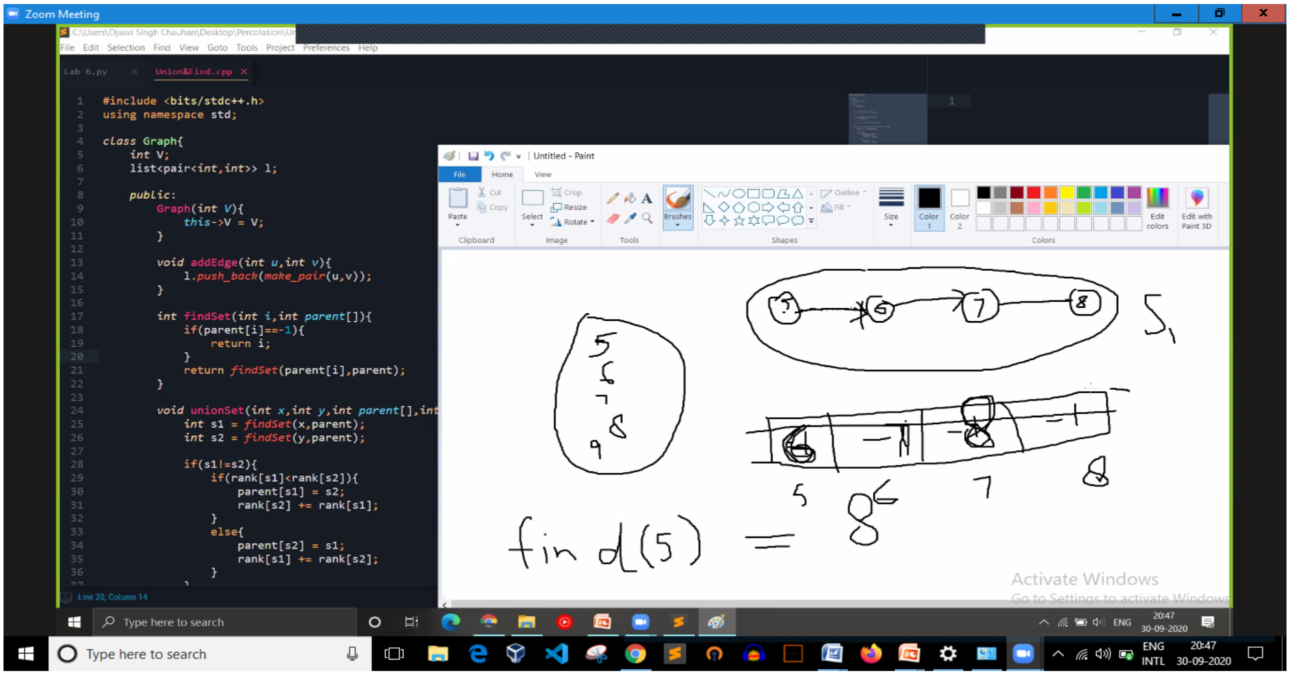
[6] Berkowitz, B., & Balberg, I. (1993). *Percolation theory and its application to groundwater hydrology*. Water Resources Research, 29(4), *775-794*.
[7] Zhang, H., 2008. *The Union-Find Problem Is Linear*. [online] semanticscholar.org. Available at: <https://www.semanticscholar.org/paper/The-Union-Find-Problem-Is-Linear-Zhang/> [Accessed 21 August 2020].

[8] Galil, Z., & Italiano, G. F. (1991). *Data structures and algorithms for disjoint set union problems*. ACM Computing Surveys (CSUR), 23(3), 319-344. (DSA for disjoint set union find problem)

[9]] Iver, S., 2016. *Percolation*. [online] swamiiyer.net. Available at: <https://www.swamiiyer.net/cs210/projects.html> [Accessed 18 August 2020].

[10] *Monte Carlo method*. (2002, June 11). https://en.wikipedia.org/wiki/Monte_Carlo_method

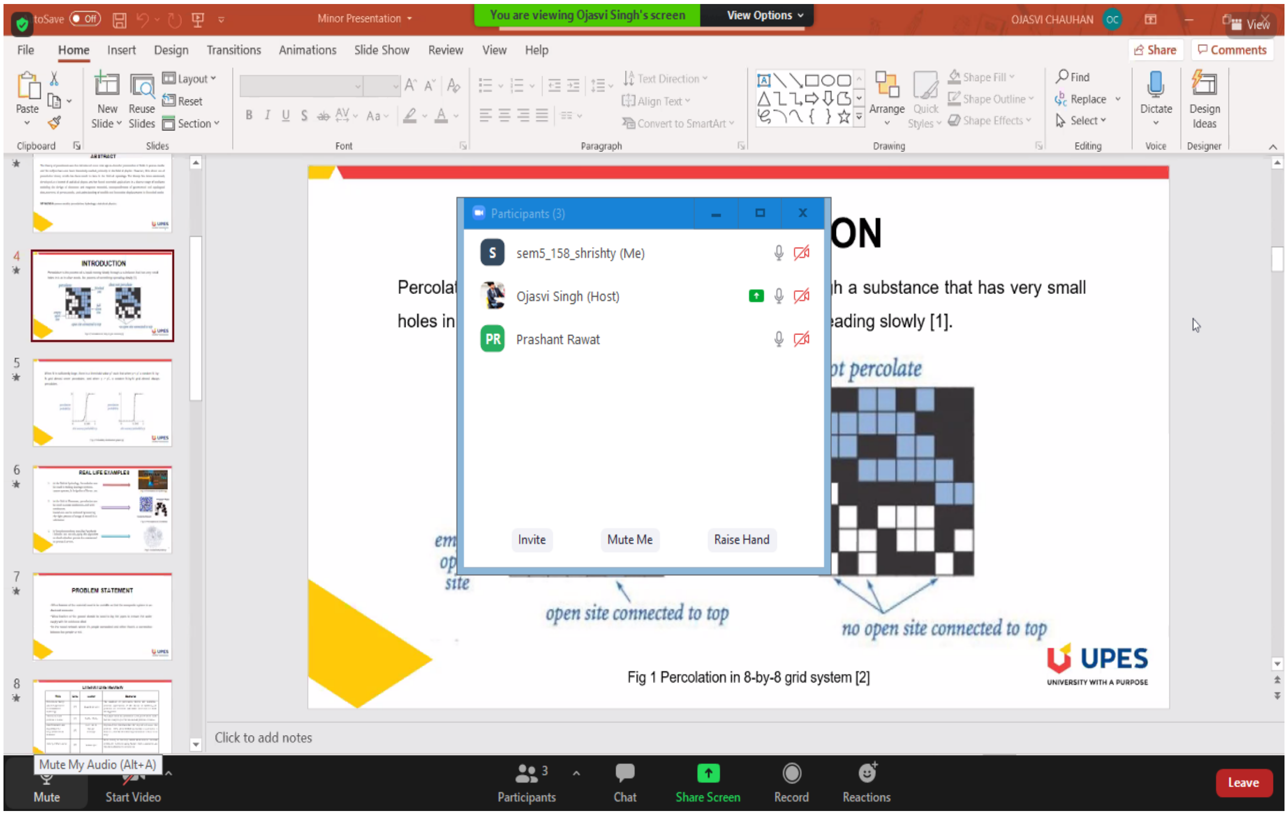
[11] Iver, S., 2016. *Percolation*. [online] swamiiyer.net. Available at: <https://www.swamiiyer.net/cs210/projects.html> [Accessed 18 August 2020].



Zoom Meeting
C:\Users\Ojasvi Singh Chauhan\Desktop\Percolation\Ur
File Edit Selection Find View Goto Tools Project Preferences Help
Lab 6.py
Union&Find.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

class Graph{
    int V;
    list<pair<int,int>> l;

    public:
        Graph(int V){
            this->V = V;
        }

        void addEdge(int u,int v){
            l.push_back(make_pair(u,v));
        }

        int findSet(int i,int parent[]){
            if(parent[i]==-1){
                return i;
            }
            return findSet(parent[i],parent);
        }

        void unionSet(int x,int y,int parent[],int
            int s1 = findSet(x,parent);
            int s2 = findSet(y,parent);

            if(s1!=s2){
                if(rank[s1]<rank[s2]){
                    parent[s1] = s2;
                    rank[s2] += rank[s1];
                }
                else{
                    parent[s2] = s1;
                    rank[s1] += rank[s2];
                }
```

Line 20, Column 14
Untitled - Paint
File Home View
Paste Cut Copy Select Crop Resize Rotate Brushes Outline Fill Size Color 1 Color 2 Edit colors Edit with Paint 3D
Clipboard Image Tools Shapes Colors
5
6
7
8
9
5 6 7 8
5,
6 -1 -8 -1
5 8 6 7 8
find(5) = 8
Activate Windows
Go to Settings to activate Windows
Type here to search
20:47
30-09-2020
ENG INTL



AutoSave Off
Minor Presentation
You are viewing Ojasvi Singh's screen
View Options
OJASVI CHAUHAN
File Home Insert Design Transitions Animations Slide Show Review View Help
Share Comments
Paste New Slide Reuse Slides Layout Reset Section
Clipboard Slides Font Paragraph Drawing Editing Voice Designer
Text Direction Align Text Convert to SmartArt Arrange Quick Styles Shape Fill Shape Outline Shape Effects Find Replace Select Dictate Design Ideas
ABSTRACT
INTRODUCTION
REAL LIFE EXAMPLES
PROBLEM STATEMENT
Participants (3)

sem5_158_shrishty (Me)

Ojasvi Singh (Host)

Prashant Rawat

Invite
Mute Me
Raise Hand

ON

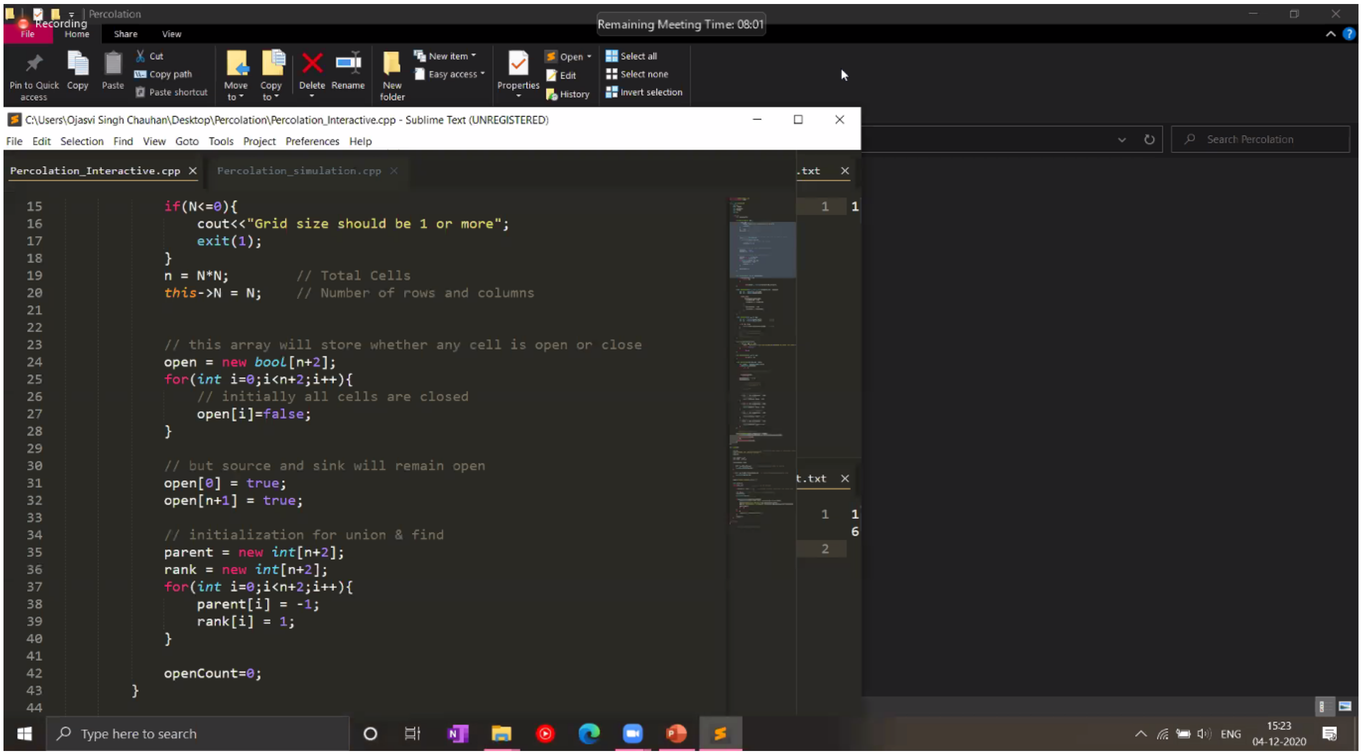
Percolat h a substance that has very small

holes in eading slowly [1].

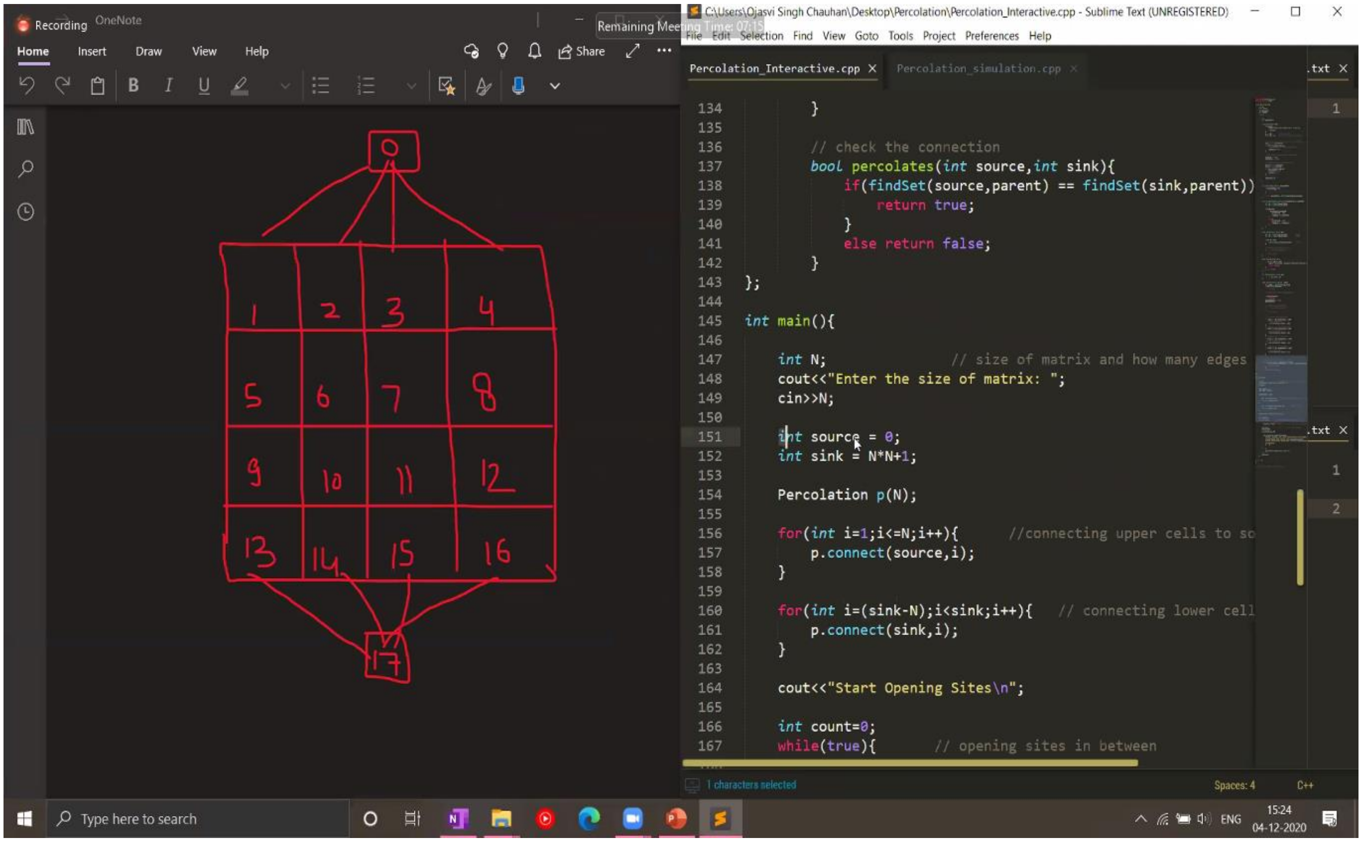
ot percolate
em
op
site
open site connected to top
no open site connected to top
UPES
UNIVERSITY WITH A PURPOSE

Fig 1 Percolation in 8-by-8 grid system [2]

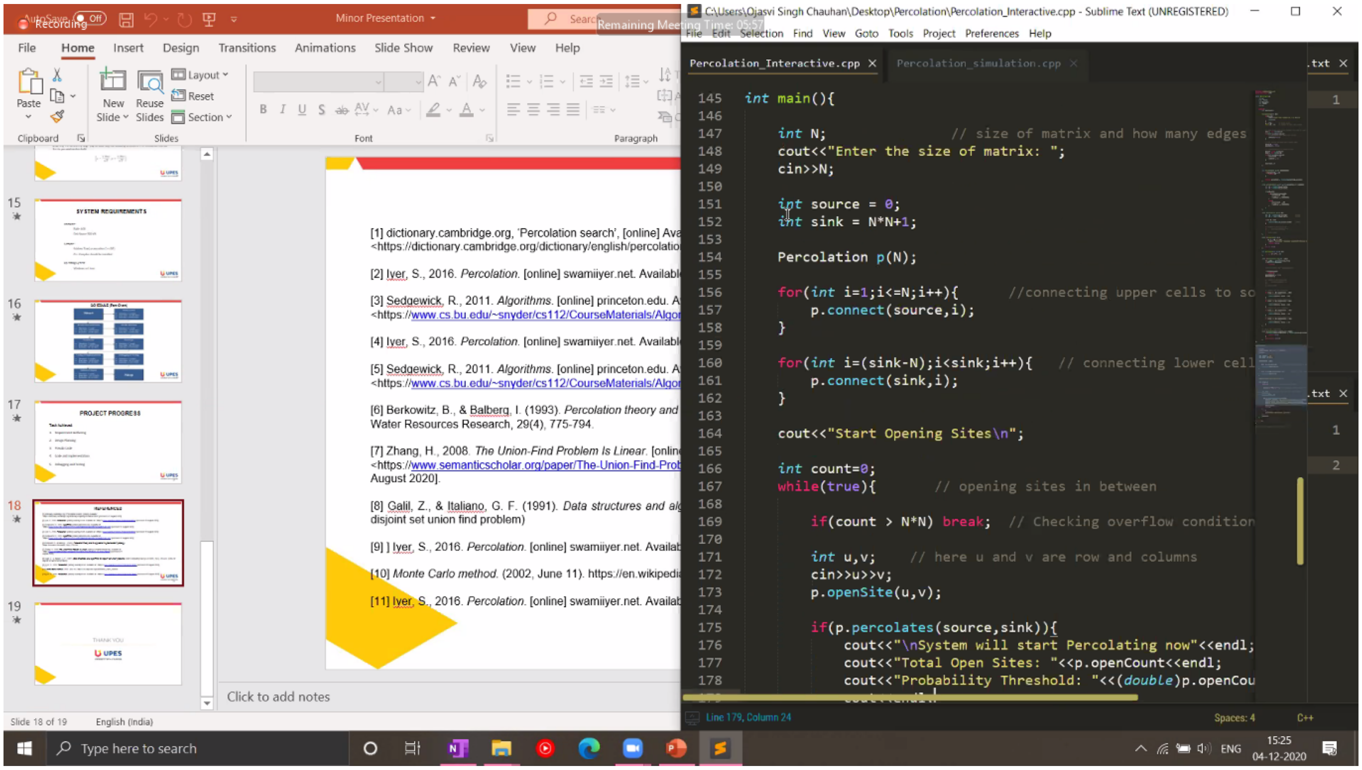
Click to add notes
Mute My Audio (Alt+A)
Mute
Start Video
Participants 3
Chat
Share Screen
Record
Reactions
Leave


Remaining Meeting Time: 08:01
C:\Users\Ojasvi Singh Chauhan\Desktop\Percolation\Percolation_Interactive.cpp - Sublime Text (UNREGISTERED)
Percolation_Interactive.cpp
Percolation_simulation.cpp

```
            if(N<=0){
                cout<<"Grid size should be 1 or more";
                exit(1);
            }
            n = N*N;        // Total Cells
            this->N = N;    // Number of rows and columns


            // this array will store whether any cell is open or close
            open = new bool[n+2];
            for(int i=0;i<n+2;i++){
                // initially all cells are closed
                open[i]=false;
            }

            // but source and sink will remain open
            open[0] = true;
            open[n+1] = true;

            // initialization for union & find
            parent = new int[n+2];
            rank = new int[n+2];
            for(int i=0;i<n+2;i++){
                parent[i] = -1;
                rank[i] = 1;
            }

            openCount=0;
        }
```


Recording
OneNote
Remaining Meeting Time: 07:15
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
C:\Users\Ojasvi Singh Chauhan\Desktop\Percolation\Percolation_Interactive.cpp - Sublime Text (UNREGISTERED)
Percolation_Interactive.cpp
Percolation_simulation.cpp
1 characters selected

```
        }

        // check the connection
        bool percolates(int source,int sink){
            if(findSet(source,parent) == findSet(sink,parent))
                return true;
            }
            else return false;
        }
};

int main(){

    int N;              // size of matrix and how many edges
    cout<<"Enter the size of matrix: ";
    cin>>N;

    int source = 0;
    int sink = N*N+1;

    Percolation p(N);

    for(int i=1;i<=N;i++){      //connecting upper cells to so
        p.connect(source,i);
    }

    for(int i=(sink-N);i<sink;i++){   // connecting lower cell
        p.connect(sink,i);
    }

    cout<<"Start Opening Sites\n";

    int count=0;
    while(true){        // opening sites in between
```


Minor Presentation
Remaining Meeting Time: 05:57
C:\Users\Ojasvi Singh Chauhan\Desktop\Percolation\Percolation_Interactive.cpp - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Percolation_Interactive.cpp
Percolation_simulation.cpp
Slide 18 of 19
English (India)
Click to add notes
Line 179, Column 24
Spaces: 4
C++
Type here to search
15:25
04-12-2020

[1] dictionary.cambridge.org, 'Percolation search', [online] Ava
<https://dictionary.cambridge.org/dictionary/english/percolatio

[2] Iyer, S., 2016. *Percolation*. [online] swamiiyer.net. Availabl

[3] Sedgewick, R., 2011. *Algorithms*. [online] princeton.edu. A
<https://www.cs.bu.edu/~snyder/cs112/CourseMaterials/Algor

[4] Iyer, S., 2016. *Percolation*. [online] swamiiyer.net. Availabl

[5] Sedgewick, R., 2011. *Algorithms*. [online] princeton.edu. A
<https://www.cs.bu.edu/~snyder/cs112/CourseMaterials/Algor

[6] Berkowitz, B., & Balberg, I. (1993). *Percolation theory and*
Water Resources Research, 29(4), 775-794.

[7] Zhang, H., 2008. *The Union-Find Problem Is Linear*. [onlin
<https://www.semanticscholar.org/paper/The-Union-Find-Prob
August 2020].

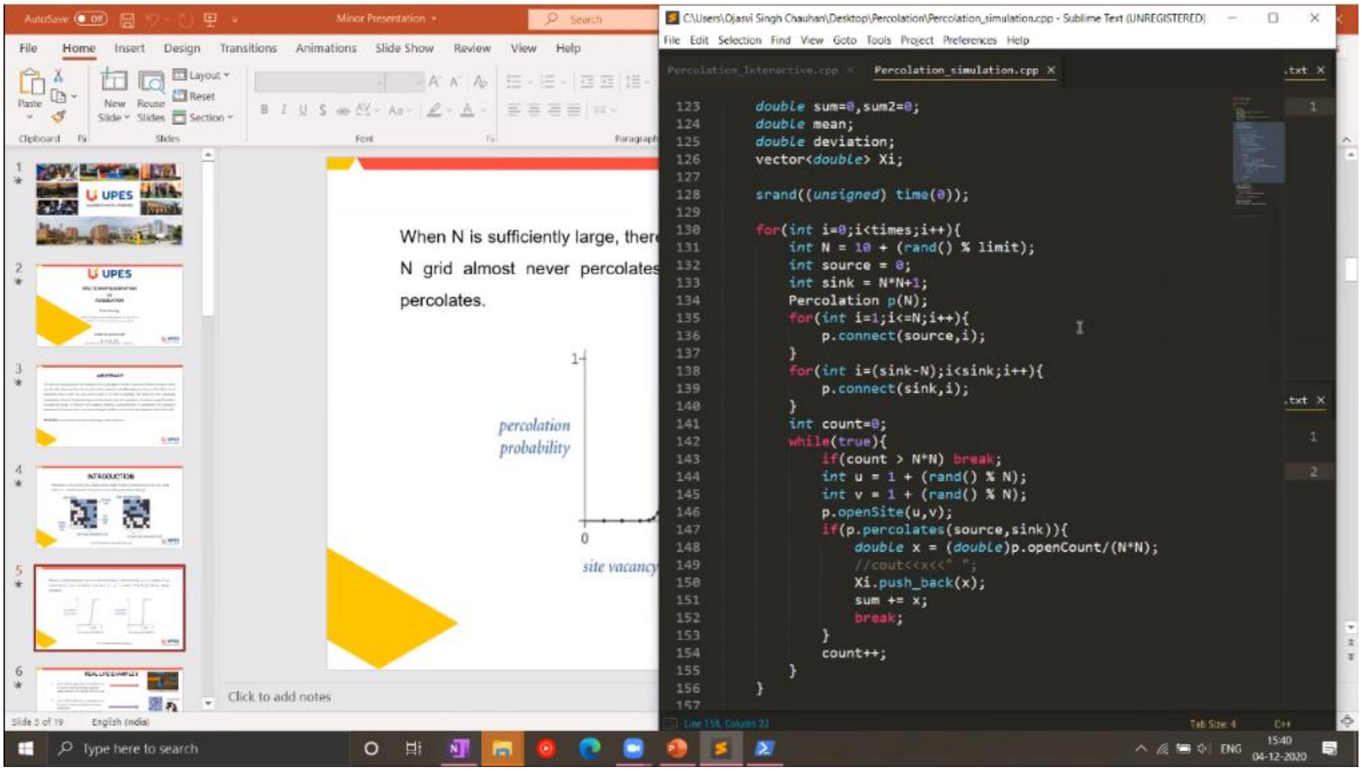
[8] Galil, Z., & Italiano, G. F. (1991). *Data structures and alg*
disjoint set union find problem)

[9]] Iyer, S., 2016. *Percolation*. [online] swamiiyer.net. Availab

[10] *Monte Carlo method*. (2002, June 11). https://en.wikipedia

[11] Iyer, S., 2016. *Percolation*. [online] swamiiyer.net. Availab

```cpp
int main(){

    int N;                  // size of matrix and how many edges
    cout<<"Enter the size of matrix: ";
    cin>>N;

    int source = 0;
    int sink = N*N+1;

    Percolation p(N);

    for(int i=1;i<=N;i++){       //connecting upper cells to so
        p.connect(source,i);
    }

    for(int i=(sink-N);i<sink;i++){    // connecting lower cell
        p.connect(sink,i);
    }

    cout<<"Start Opening Sites\n";

    int count=0;
    while(true){        // opening sites in between

        if(count > N*N) break;  // Checking overflow condition

        int u,v;    // here u and v are row and columns
        cin>>u>>v;
        p.openSite(u,v);

        if(p.percolates(source,sink)){
            cout<<"\nSystem will start Percolating now"<<endl;
            cout<<"Total Open Sites: "<<p.openCount<<endl;
            cout<<"Probability Threshold: "<<(double)p.openCou
```


Minor Presentation
C:\Users\Ojasvi Singh Chauhan\Desktop\Percolation\Percolation_simulation.cpp - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Percolation_Interactive.cpp
Percolation_simulation.cpp
Slide 5 of 19
English (India)
Click to add notes
1
percolation probability
0
site vacancy
Line 158, Column 22
Tab Size: 4
C++
Type here to search
15:40
04-12-2020

When N is sufficiently large, ther
N grid almost never percolates
percolates.

```cpp
    double sum=0,sum2=0;
    double mean;
    double deviation;
    vector<double> Xi;

    srand((unsigned) time(0));

    for(int i=0;i<times;i++){
        int N = 10 + (rand() % limit);
        int source = 0;
        int sink = N*N+1;
        Percolation p(N);
        for(int i=1;i<=N;i++){
            p.connect(source,i);
        }
        for(int i=(sink-N);i<sink;i++){
            p.connect(sink,i);
        }
        int count=0;
        while(true){
            if(count > N*N) break;
            int u = 1 + (rand() % N);
            int v = 1 + (rand() % N);
            p.openSite(u,v);
            if(p.percolates(source,sink)){
                double x = (double)p.openCount/(N*N);
                //cout<<x<<" ";
                Xi.push_back(x);
                sum += x;
                break;
            }
            count++;
        }
    }
```

REFERENCES

[1] dictionary.cambridge.org, 'Percolation search', [online] Available: <https://dictionary.cambridge.org/dictionary/english/percolation> [Accessed: 22 August 2020].

[2] Iyer, S., 2016. *Percolation*. [online] swamiiyer.net. Available at: <https://www.swamiiyer.net/cs210/projects.html> [Accessed 18 August 2020].

[3] Sedgewick, R., 2011. *Algorithms*. [online] princeton.edu. Available at: <https://www.cs.bu.edu/~snyder/cs112/CourseMaterials/AlgorithmsChapterOne.pdf> [Accessed 17 August 2020].

[4] Sedgewick, R., 2011. *Algorithms*. [online] princeton.edu. Available at: <https://www.cs.bu.edu/~snyder/cs112/CourseMaterials/AlgorithmsChapterOne.pdf> [Accessed 17 August 2020].

[5] Iyer, S., 2016. *Percolation*. [online] swamiiyer.net. Available at: <https://www.swamiiyer.net/cs210/projects.html> [Accessed 18 August 2020].

[6] Berkowitz, B., & Balberg, I. (1993). *Percolation theory and its application to groundwater hydrology.*Water Resources Research, 29(4), 775-794.

[7] Zhang, H., 2008. *The Union-Find Problem Is Linear*. [online] semanticscholar.org. Available at: <https://www.semanticscholar.org/paper/The-Union-Find-Problem-Is-Linear-Zhang/> [Accessed 21 August 2020].

[8] Galil, Z., & Italiano, G. F. (1991). *Data structures and algorithms for disjoint set union problems.* ACM Computing Surveys (CSUR)

[9] Iyer, S., 2016. *Percolation*. [online] swamiiyer.net. Available at: <https://www.swamiiyer.net/cs210/projects.html> [Accessed 18 August 2020].

[10] *Monte Carlo method*. (2002, June 11). <https://en.wikipedia.org/wiki/Monte_Carlo_method>

[11] Sedgewick, R., 2011. *Algorithms*. [online] princeton.edu. Available at: <https://www.cs.bu.edu/~snyder/cs112/CourseMaterials/AlgorithmsChapterOne.pdf> [Accessed 17 August 2020].