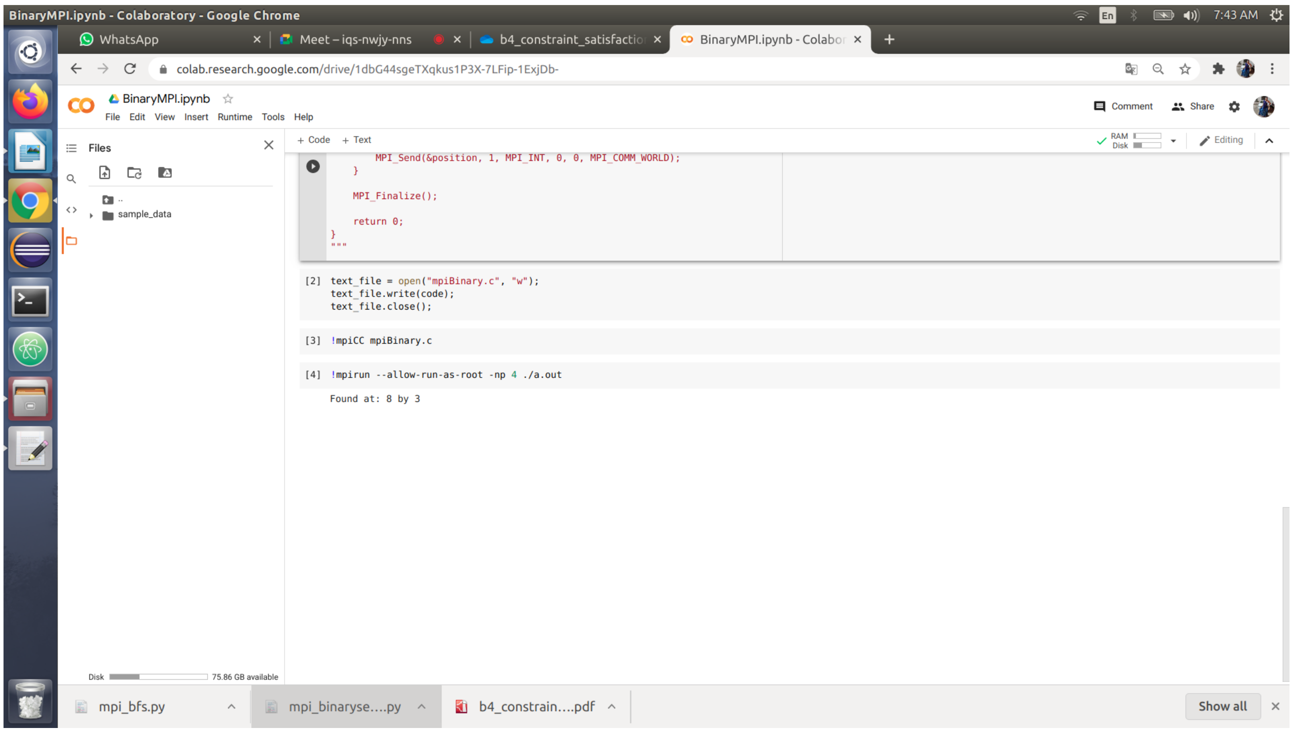
CODE:::

```
# -*- coding: utf-8 -*-
"""mpi_binarysearch.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1XEY7wmW1GaCtZ6Nmsx9OPFZRb5r9ideu
"""

code = """
#include<mpi.h>
#include<stdio.h>

#define n 12

#define key 55

int a[] = {1,2,3,4,7,9,13,24,55,56,67,88};

int a2[20];

int binarySearch(int *array, int start, int end, int value) {
    int mid;

    while(start <= end) {
        mid = (start + end) / 2;
        if(array[mid] == value)
            return mid;
        else if(array[mid] > value)
            end = mid - 1;
        else
            start = mid + 1;
    }
    return -1;
}

int main(int argc, char* argv[]) {
    int pid, np, elements_per_process, n_elements_received;

    MPI_Status status;

    MPI_Init(&argc, &argv);

    MPI_Comm_rank(MPI_COMM_WORLD, &pid);
    MPI_Comm_size(MPI_COMM_WORLD, &np);

    if(pid == 0) {
        int index, i;

        if(np > 1) {
            for(i=1; i<np-1; i++) {

                index = i * elements_per_process;
                //element count
                MPI_Send(&elements_per_process, 1, MPI_INT, i, 0, MPI_COMM_WORLD);

                MPI_Send(&a[index], elements_per_process, MPI_INT, i, 0, MPI_COMM_WORLD);

            }

            index = i* elements_per_process;

            int elements_left = n - index;

            MPI_Send(&elements_left, 1, MPI_INT, i, 0, MPI_COMM_WORLD);

            MPI_Send(&a[index], elements_left, MPI_INT, i, 0, MPI_COMM_WORLD);
        }

        int position = binarySearch(a, 0, elements_per_process-1, key);

        if(position != -1)
          printf("Found at: %d", position);

        int temp;

        for(i=1; i<np; i++) {
            MPI_Recv(&temp, 1, MPI_INT, MPI_ANY_SOURCE, 0, MPI_COMM_WORLD,
&status);
            int sender = status.MPI_SOURCE;

            if(temp != -1)
                printf("Found at: %d by %d", (sender*elements_per_process)+temp, sender);
        }
    }

    else {
        MPI_Recv(&n_elements_received, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &status);

        MPI_Recv(&a2, n_elements_received, MPI_INT, 0, 0, MPI_COMM_WORLD, &status);

        int position = binarySearch(a2, 0, n_elements_received-1, key);

        MPI_Send(&position, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
    }

    MPI_Finalize();

    return 0;
}
"""
```

```
text_file = open("mpiBinary.c", "w");
text_file.write(code);
text_file.close();

!mpiCC mpiBinary.c

!mpirun --allow-run-as-root -np 4 ./a.out
```

OUTPUT::



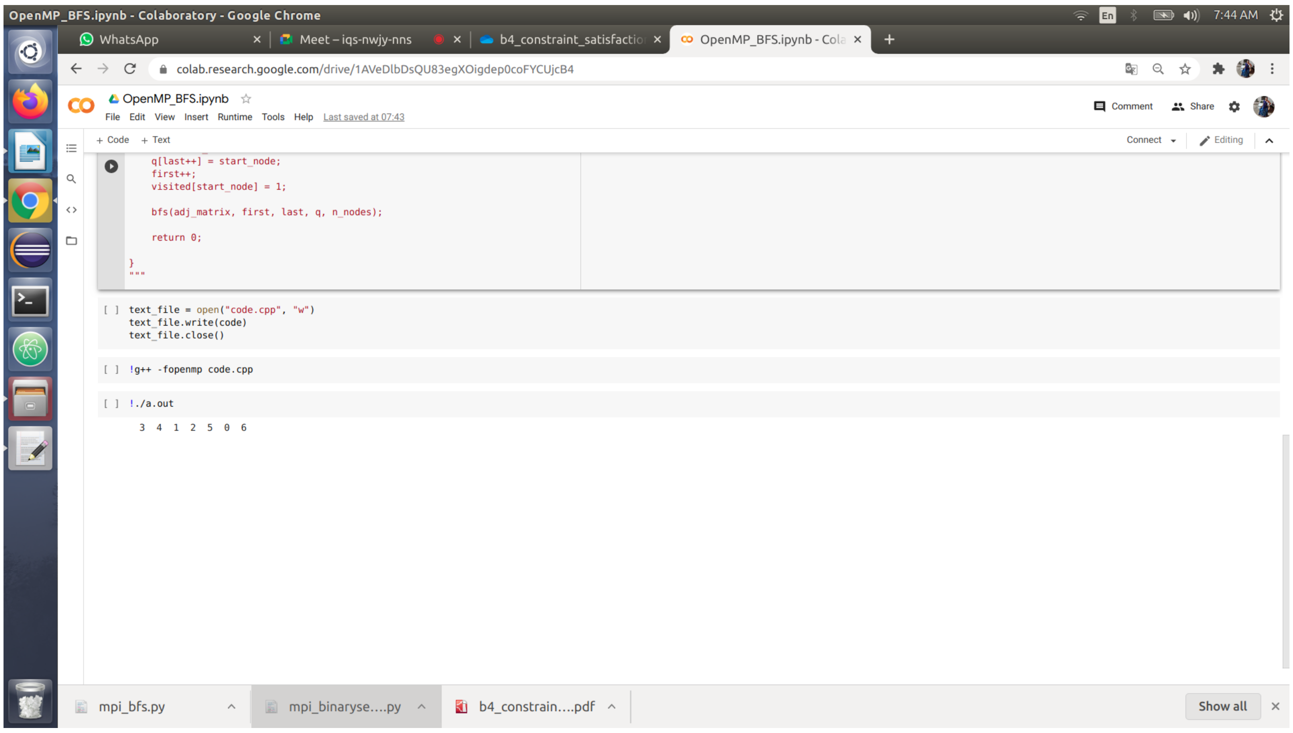
```
            MPI_Send(&position, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
        }

        MPI_Finalize();

        return 0;
}
"""
```

```
[2] text_file = open("mpiBinary.c", "w");
    text_file.write(code);
    text_file.close();

[3] !mpiCC mpiBinary.c

[4] !mpirun --allow-run-as-root -np 4 ./a.out

    Found at: 8 by 3
```

CODE:::

```
# -*- coding: utf-8 -*-
"""mpi_bfs.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1TDGr4zoYX8bOYXis2VNjnGYQSF9Ug1i3
"""

code = """
#include<iostream>
#include<omp.h>

using namespace std;
int q[100];
int visited[7];
int local_q;

void bfs(int adj_matrix[7][7], int first, int last, int q[], int n_nodes) {
    if(first==last)
      return;

    int cur_node = q[first++];
    cout<<"  "<<cur_node;

    omp_set_num_threads(3);

    #pragma omp parallel for shared(visited)
    for(int i=0; i<n_nodes; i++) {
        if(adj_matrix[cur_node][i] == 1 && visited[i] == 0){
            q[last++] = i;
            visited[i] = 1;
        }
    }

    bfs(adj_matrix, first, last, q, n_nodes);
}

int main() {
    int first = -1;
    int last = 0;
    int n_nodes = 7;

    for(int i=0; i<n_nodes; i++) {
        visited[i] = 0;
    }

    int adj_matrix[7][7] = {
      {0, 1, 1, 0, 0, 0, 0},
      {1, 0, 1, 1, 0, 0, 0},
      {1, 1, 0, 0, 1, 0, 0},
      {0, 1, 0, 0, 1, 0, 0},
      {0, 0, 1, 1, 0, 1, 0},
      {0, 0, 0, 0, 1, 0, 1},
      {0, 0, 0, 0, 0, 1, 0}
    };

    int start_node = 3;
    q[last++] = start_node;
    first++;
    visited[start_node] = 1;

    bfs(adj_matrix, first, last, q, n_nodes);

    return 0;

}
"""

text_file = open("code.cpp", "w")
text_file.write(code)
text_file.close()

!g++ -fopenmp code.cpp

!./a.out
```

OUTPUT:::


OpenMP_BFS.ipynb - Colaboratory - Google Chrome
WhatsApp
Meet – iqs-nwjy-nns
b4_constraint_satisfactio
OpenMP_BFS.ipynb - Cola
colab.research.google.com/drive/1AVeDlbDsQU83egXOigdep0coFYCUjcB4
OpenMP_BFS.ipynb
File Edit View Insert Runtime Tools Help Last saved at 07:43
Comment Share
+ Code + Text
Connect Editing
mpi_bfs.py
mpi_binaryse....py
b4_constrain....pdf
Show all

```
    q[last++] = start_node;
    first++;
    visited[start_node] = 1;

    bfs(adj_matrix, first, last, q, n_nodes);

    return 0;

}
"""
```

```
[ ] text_file = open("code.cpp", "w")
    text_file.write(code)
    text_file.close()
```

```
[ ] !g++ -fopenmp code.cpp
```

```
[ ] !./a.out
```

```
   3  4  1  2  5  0  6
```