```
%%writefile binary.cpp
#include<iostream>
#include<omp.h>
using namespace std;
int par_result = -1;

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 *arr,i,j;
  int n = 100000000;
  srand(n);
  arr=new int[n];
  cout<<"\nArray to be searched: 0 - "<<n;
  for(int i=0; i<n; i++) {
      arr[i] = i;
  }
  int val = rand()%n;
  cout<<"\nValue to be searched: "<<val<<endl;
  //BINARY SEARCH
    int threads = 4;
    omp_set_num_threads(threads);
    int blocksize = n/threads;
  //Calling parallel binary search
    #pragma omp parallel for shared(par_result)
    for(int i =0;i<threads;i++){
        cout<<binarysearch(arr,i*blocksize,i*blocksize+blocksize-1,val)<<" ";
    }
  return 0;
}
```

```
Writing binary.cpp
```

```
!g++ -fopenmp binary.cpp
```

```
!./a.out
```

```
Array to be searched: 0 - 100000000
Value to be searched: 67557522
67557522 -1 -1 -1
```

```
%%writefile bfs.cpp
#include<iostream>
#include<omp.h>
int visited[11];
using namespace std;

void bfs(int adj_matrix[11][11], 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 = 0;
    int last = 0;
    int n_nodes = rand()%10+2;
    int q[100];
    int local_q;
    for(int i=0; i<n_nodes; i++) {
        visited[i] = 0;
    }
    int adj_matrix[11][11];
    cout<<"\nMatrix: "<<endl;
    for(int i=0; i<n_nodes;i++){
        for(int j=0;j<n_nodes;j++){
            adj_matrix[i][j] = rand()%2;
            cout<<adj_matrix[i][j]<<" ";
        }
        cout<<endl;
    }
    int start_node = rand()%n_nodes;
    cout<<"\nStart node: "<<start_node<<endl;
    q[last++] = start_node;
    visited[start_node] = 1;
    bfs(adj_matrix, first, last, q, n_nodes);
    return 0;

```

```
52 }
```

```
Writing bfs.cpp
```

```
1 !g++ -fopenmp bfs.cpp
```

```
1 !./a.out
```

```
Matrix:
0 1 1 1 1
0 0 1 1 0
1 0 1 1 0
0 0 0 0 1
0 1 1 0 0

Start node: 2
2 3 0 4 1
```