```
#include<iostream>
#include<omp.h>
#include<bits/stdc++.h>

using namespace std;
int visited[7];
typedef pair<int, int> pi;

int main() {
    int n_nodes = 7;
    int target = 6;
    priority_queue<pi, vector<pi>, greater<pi> > q;

    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.push({0,start_node});
    visited[start_node] = 1;

    while(q.size()){
        pi cur_node = q.top();
        q.pop();

        int flag = 0;

        cout<<"Current Node: " <<cur_node.second << endl;

        #pragma omp parallel for shared(visited)
        for(int i=0; i<n_nodes; i++) {
            int tid = omp_get_thread_num();
            cout << "Thread name : " << tid << " " << i << endl;
            if(adj_matrix[cur_node.second][i] == 1 && visited[i] == 0){
                q.push({cur_node.first + 1, i});
                if(i == target)
                    flag = 1;
                visited[i] = 1;
            }
        }
        if(flag == 1) {
            cout << "Target Reached, distance travelled is : " << cur_node.first +
1 << endl;
            break;
        }
        cout << "DONE WITH Node : " << cur_node.second << endl;

    }
```

```
    return 0;
}
```