A* Searching Algorithm

Name: Bhavin Patil
Roll No.: 66

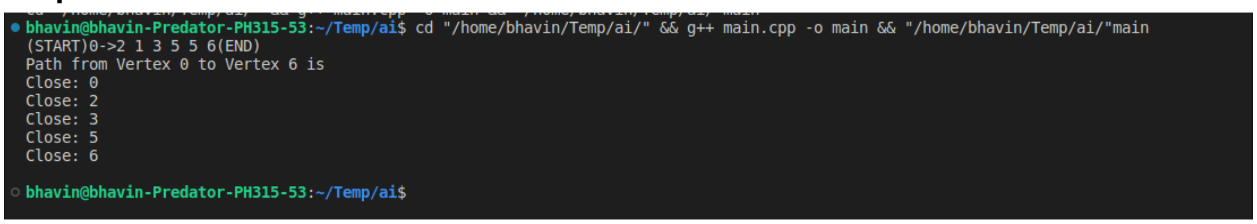
Code:

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

typedef pair<int, int> iPair;

struct node
{
   node(int a, int b)
   {
       dest = a;
       weight = b;
   }
   int dest;
   int weight;
};

void astar(int start, int goal, vector<node> v[], vector<bool> visited,
int parent[], int heu[], int dist[], int n)
{
   priority_queue<iPair, vector<iPair>, greater<iPair>> pq;
   pq.push(make_pair(heu[start], start));
   dist[start] = 0;
   int currStart = start;
   while (!pq.empty())
   {
       int x = pq.top().second;
       // if (x == goal)
       // {
       //     cout << goal << "(END) ";
       //     return;
       // }
       if (x == start)
```

```
           cout << "(START)" << x << "->";
       else if (x == goal)
       {
           cout << x << "(END)";
           break;
       }
       else
       {
           cout << x << " ";
       }
       pq.pop();
       if (!visited[x])
       {
           for (int i = 0; i < v[x].size(); i++)
           {
               if (!visited[v[x][i].dest])
               {
                   int f = dist[x] + v[x][i].weight + heu[v[x][i].dest];
                   pq.push(make_pair(f, v[x][i].dest));

                   dist[v[x][i].dest] = dist[x] + v[x][i].weight;
                   parent[v[x][i].dest] = x;
               }
           }
           // cout << "Open: " << p_s << "\n";
           visited[x] = 1;
       }
   }
}
int main()
{
   int numOfNodes = 7;

   vector<node> v[numOfNodes];
   vector<bool> visited(numOfNodes, false);
   int parent[numOfNodes];
   int heu[numOfNodes];
   int dist[numOfNodes];

   v[0].push_back(node(1, 4));
   v[0].push_back(node(2, 3));
   v[1].push_back(node(4, 5));
   v[1].push_back(node(5, 12));
   v[2].push_back(node(5, 10));
   v[2].push_back(node(3, 7));
   v[3].push_back(node(5, 2));
   v[4].push_back(node(6, 16));
   v[5].push_back(node(6, 5));

   heu[0] = 14;
   heu[1] = 12;
   heu[2] = 11;
   heu[3] = 6;
   heu[4] = 11;
   heu[5] = 4;
   heu[6] = 0;
   for (int i = 0; i < 7; i++)
   {
       visited[i] = 0;
       parent[i] = i;
       dist[i] = INT_MAX;
   }
   astar(0, 6, v, visited, parent, heu, dist, numOfNodes);

   int cur = 6;
   cout << endl;
   cout << "Path from Vertex " << 0 << " to Vertex " << 6 << " is " <<
endl;
   stack<int> path;
   do
   {
       path.push(cur);
       cur = parent[cur];
   } while (cur != 0);

   path.push(0);

   while (!path.empty())
   {
       cout << "Close: " << path.top() << "\n";
```

```
        path.pop();
    }
    cout << endl;
    return 0;
}
```

Output:



```
● bhavin@bhavin-Predator-PH315-53:~/Temp/ai$ cd "/home/bhavin/Temp/ai/" && g++ main.cpp -o main && "/home/bhavin/Temp/ai/"main
  (START)0->2 1 3 5 5 6(END)
  Path from Vertex 0 to Vertex 6 is
  Close: 0
  Close: 2
  Close: 3
  Close: 5
  Close: 6

○ bhavin@bhavin-Predator-PH315-53:~/Temp/ai$
```