1) What are the algorithms used in this project?
幅優先探索(BFS)と深さ優先探索(DFS)

2) Describe the algorithms briefly based on your understanding.
深さ優先探索(DFS)
スタックを使って実装する。
1．現在の頂点に探索を開始する頂点(vertex_starting_search)をセットする。
2．現在の頂点を探索済みにする。
3．現在の頂点が探索を終了する頂点だったら累積コストを返して探索終了する。
4．グラフの頂点の数だけ5〜8を繰り返す。
5．その頂点が未探索かつ取り出した頂点の子である場合、6〜7の処理を行う。
6．現在の頂点からその頂点までのコストを累積コストに加算する。
7．その頂点を現在の頂点として2〜処理を再帰的に呼び出す。
8．探索を終了する頂点が探索済みだったら繰り返しを終了する。
9．累積コストを返す。

幅優先探索(BFS)
キューを使って実装する。
1．キューを作成して探索を開始する頂点(vertex_starting_search)を入れる。
2．取り出した頂点を探索済みにする。
3．キューが空になるまで3〜10を繰り返す。
4．キューから頂点を取り出す。
5．取り出した頂点が探索を終了する頂点だったら探索終了する。
6．グラフの頂点の数だけ7〜10を繰り返す。
7．その頂点が未探索かつ取り出した頂点の子である場合、8〜10の処理を行う。
8．その頂点を探索済みにする。
9．その頂点をキューに入れる。
10．取り出した頂点からその頂点までのコストを累積コストに加算する。

3) Explain the meaning of the results obtained by running the completed program.
DFS、BFSで得られた結果が表示される。
どちらも最小の累積コストが得られる。

4) Add the algorithm assignment here:
```
// TODO: 1. Set the arguments in order to start DFS.
float cost = dfs_visit(g, vertex_states, g->vertex_starting_search, 0);

// TODO: 2. Write the terminating condition for the recursive search.
if (current_vertex == g->vertex_end_search) {
    return accumulate_cost;
}

// TODO: 3. Call dfs_visit recursively.
// Note: Cost of the child = cost of the edge + accumulate cost.
//       Also, pay attention to the return value.
accumulate_cost = dfs_visit(g, vertex_states, i, g->edge_costs[current_vertex][i] +
```

```
accumulate_cost);

// TODO: 4. Write the terminating condition for the search.
if (current_vertex == g->vertex_end_search) {
    break;
}

// TODO: 5. Calculate the accumulate cost.
// Set accumulate cost from the starting vertex to the i-th child vertex.
accumulate_costs[i] = accumulate_costs[current_vertex] + g-
>edge_costs[current_vertex][i];
```