What are the algorithms used in this project?
->Depth First Search & Breadth First Search

Describe the algorithms briefly based on your understanding.
->Depth First Searchはオープンリストにスタックを使って、隣接する未訪問の頂点を訪問し、それを訪問済みとしてマークする。次に、それをスタックにプッシュして、隣接する頂点が見つからない場合は、スタックから頂点をポップアップします。
Breadth First Searchはオープンリストにキューを使って、未訪問の隣接頂点を訪問しながら、キューから頂点を繰り返し削除します。

Explain the meaning of the results obtained by running the completed program.
->現在の頂点につながっている頂点とつながっていない頂点が1と0で隣接行列にあらわされているのでそれを読み込み、DFSとBFSで頂点のつながりをみつける。
今回は頂点を訪問したあとその頂点までのコストも計算し、最後にその合計を出力する。

Explanation of modified points (required)
TODO:1
  float cost = dfs_visit(g, vertex_states, g->vertex_starting_search, 0);
  dfs_visit関数の引数に一番最初の頂点（ルート）を入れて、最初のコストを0にしてdfs_visit関数を動かした。
TODO:2
  if (current_vertex == g->vertex_end_search)
  現在の頂点が最後の頂点だった場合、再起を終了させる。
TODO:3
  accumulate_cost = dfs_visit(g, vertex_states, i, accumulate_cost + g->edge_costs[current_vertex][i]);
  今までのコストと現在の頂点と繋がっている頂点とのコストを足して再起処理をする。
TODO:4
  g->vertex_end_search == current_vertex
  TODO:2と同じ。
TODO:5
  vertex_states[g->vertex_starting_search] = VISITED;
  一番最初の頂点（ルート）をVISITEDの状態にした。
TODO:6
  accumulate_costs[i] = accumulate_costs[current_vertex] + g->edge_costs[current_vertex][i];
 今までのコストと現在の頂点と繋がっている頂点とのコストをたす。

Discussion (if needed)

Comments (if needed)