ユーザが入力する地図データに対する経路計算プログラムの開発

岩手県立一関第一高等学校理数科３年　鈴木明星　菅原和真　鈴木毅　長尾瞬
SUZUKI Meisei, SUGAWARA Kazuma, SUZUKI Takeshi and NAGAO Shun

要約

今日，様々な経路計算アプリケーションが幅広く利用されている一方で，データベース化されていない地図上での最短経路情報を提示できるプログラムは普及していない。私たちはユーザが自ら簡単な操作で地図データを作成でき，そのデータ上での最短経路計算の結果を得られるプログラムのモデルを実際に開発することによって，その実現可能性を示した。

〈キーワード〉　グラフ理論　ダイクストラ法　最短経路問題　災害避難

ABSTRACT

While various kinds of route calculating applications have already been developed and commonly used today, programs that calculates and shows the shortest routes based on a map data which does not exist as a compiled database are not available. We demonstrated the feasibility of such programs by actually developing the model of a program with which users can easily create a map data and acquire the result of the calculation of shortest routes on the created map data.

Keywords:　graph theory, Dijkstra's algorithm, shortest path problem, disaster evacuation

１　はじめに

今日，あらかじめ存在する大規模な地図データベースにおいて経路情報を提示するアプリケーションが幅広く普及している一方で，特定の施設内や，催事会場の地図のような，データベース化されていない地図について容易に経路情報が得られるプログラムは普及していない。しかしながらこのような場所においても，最短経路の情報の取得は，利便性の確保，混雑の軽減，より迅速で安全な災害避難方法の構築等の目的の達成のために重要であるといえる。

本研究では，ユーザが地図データを作成・編集できるインターフェースと，そこで入力された地図データにおいての経路計算の実行と計算結果の提示を行うプログラムを開発することによって，地図がデータベース上にない場合にも容易に経路情報を得られるようにすることが可能であることを示した。

２　開発方法

本研究ではすべてのプログラムの開発に JavaScript 及び HTML を使用した。これにより，ウェブページ上でプログラムを動作させることができる。

３　地図データ

3-1 地図データの定義

本研究で開発するプログラムは，エッジとノードからなる重み付き有向グラフ（Fig. 1）においてのノード間の経路を計算し，提示するものであり，本研究ではこの計算対象のグラフを地図データと呼ぶ。

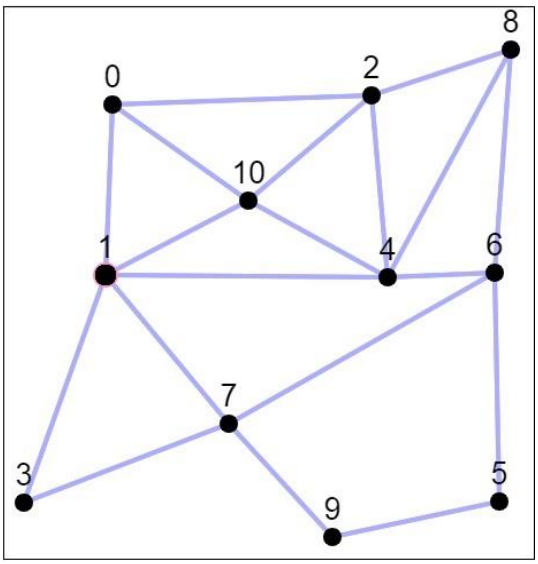

Fig.1 Graph consisting of nodes and edges

3-2 地図データの記録

プログラムの内部では，ノードの座標を格

納するための配列，各ノードの接続関係とエッジのコストを格納するための配列を用意することにより，地図データを記録する。

以降の説明において，前者をノード配列，後者をエッジ配列と呼ぶ。

3-3 地図データの書出・読込

プログラム内部で管理されているこれらの配列の数値を文字列に変換して出力することによって，ユーザが作成した地図データを保存・共有できるようにする（Fig.2）。

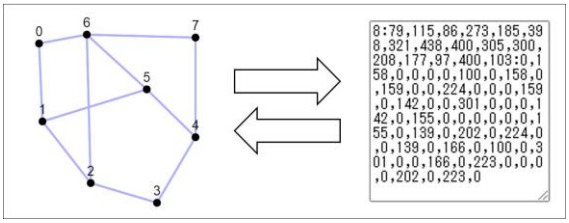

Fig.2 Conversion between the graph and the strings

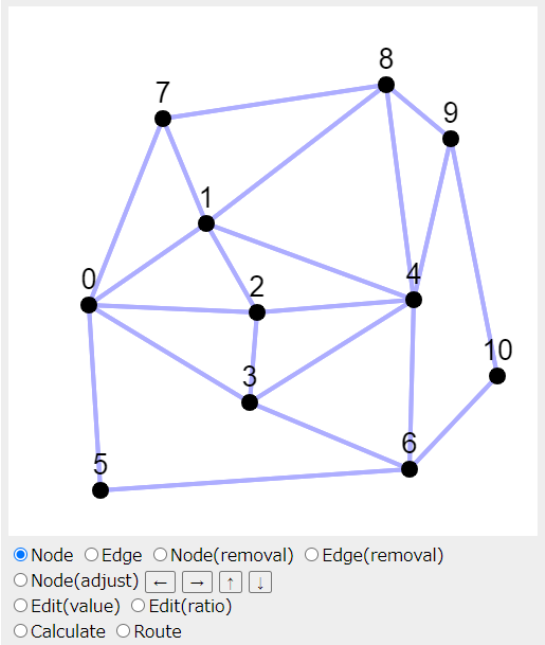

Fig.3 Graphical interface of the program

4 モデルのユーザインターフェース

4-1 キャンバス

開発したプログラムでは，ページ上に長方形状のキャンバスを配置し，そのキャンバス上に地図データを描画している。これによって，表示されている地図データの上からのデータの編集，経路計算の対象のノード指定を可能にする（Fig.3）。

また，ページの HTML ファイルに対して相対パスで指定された特定のディレクトリに画像ファイルを入れておくことで，キャンバス上に背景としてその画像を表示することができるようにする。

4-2 キャンバス外

５節及び６節で述べるような操作は主にキャンバス上でのクリックによって実行可能なものだが，それらの操作を区別するために，キャンバス外に操作の種類を切り替えるためのラジオボタンなどを配置している。

4-3 スマートフォン上での操作

プログラムをスマートフォンやタブレット端末上で使用する場合には，クリックによる操作をすべて画面タッチによっても行うことができるようにする。ただし，以降の説明では，クリックによる方法のみを述べる。

5 地図データの入力

5-1 ノードの指定

ノードを指定する際には，キャンバス上をクリックした時にクリックした位置から最も近いノードが指定されるようにする。

また，マウスを使用してノードを指定する場合には，移動中のマウスカーソルに最も近い位置にあるノードをハイライト表示する(Fig.4)。

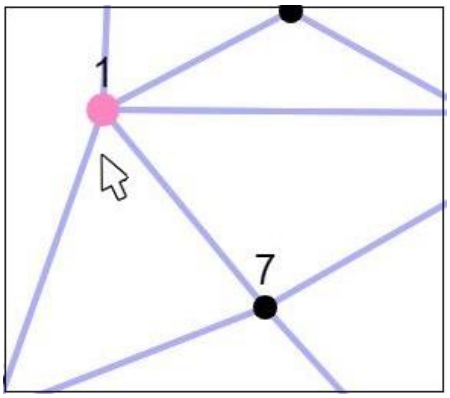

Fig.4 Highlighted node

5-2 ノードの追加

プログラム画面上のキャンバス内部をクリックすることによりノードを入力することができるようにする（Fig.5）。

この時，クリックした位置のキャンバス上

での座標を取得し，その値をノード配列に記録する。また，エッジ配列の長さを拡張すると同時に，新規ノードは他のノードのいずれとも接続されていない状態で入力されることから，新規ノードと既存ノードの間のコストとして，ノードがエッジで接続されていないことを内部的に表す数値の０を記録する。

5-3 ノードの削除

5-2 節の操作によってノードを指定することで，特定のノードを地図データから取り除くことができるようにする（Fig.5）。

このとき，5-1 節とは逆に，ノード配列から削除されたノードの座標を取り除き，エッジ配列内での，削除されるノードと他のノード間のコストを削除し，エッジ配列の長さを小さくする。

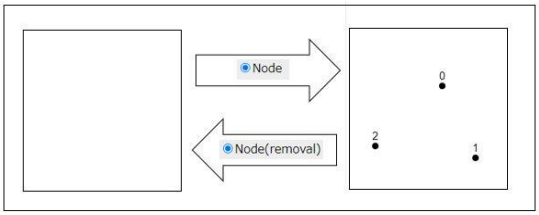

Fig.5 Addition and removal of nodes

5-4 エッジの追加

キャンバス上の２ノードを連続して指定することで，エッジを追加することができるようにする（Fig.6）。

このとき，エッジ配列内の指定された２ノード間のコストを，未接続を表す０から，ノード配列の座標に基づいて計算された，平面上での２ノード間の距離に置き換える。

5-5 エッジの削除

5-2 節の操作によってエッジを指定することで，特定のエッジを地図データから取り除くことができるようにする(Fig.6)。

このとき，5-4 節でのコストの計算処理とは逆に，エッジ配列内の該当エッジのコストを０に置き換える。

5-6 ノードの座標調整

5-2 節の操作によってノードを指定したのち，指定したノードの座標をピクセル単位で調整することができるようにする。

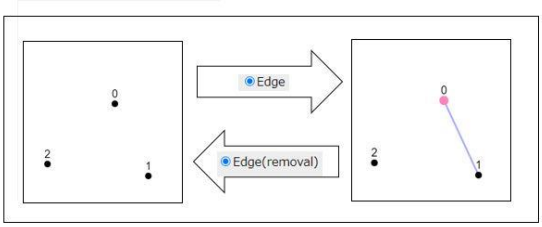

Fig.6 Addition and removal of edges

このとき，ノード配列内の該当ノードの数値を変更すると同時に，座標を調整したノードを端点とするエッジのコストを，調整後の座標を使用して距離を計算することにより修正する。

5-7 エッジの編集

5-2 節の操作によってエッジを指定することで，特定のエッジのコストの値を修正することができるようにする。

変更後のコストは実際の値を入力するか，変更前の値に対する変更後の値の割合を入力することによって指定できるようにする。

この操作においては，選択した２ノードの選択順の方向のみのエッジを対象としてコストを修正する。したがって，この操作によって通行方向により通行コストが異なるエッジや，単一の方向にのみ通行可能なエッジを含む地図データを入力することも可能にする。

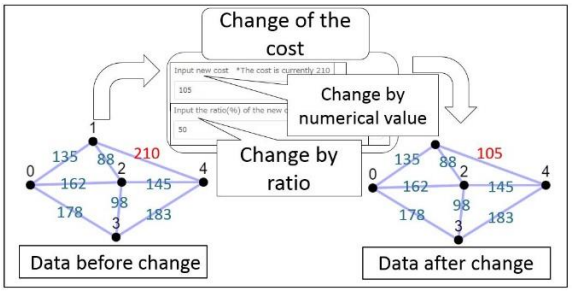

Fig.7 Change of the cost of nodes

６　経路の計算

6-1 特定１ノードから全ノードへの最短経路長の計算

本研究で扱うようなコストが非負実数値をとるグラフにおいては，ダイクストラのアルゴリズムによって特定の１ノードを始点としたときの他の全てのノードへの最短経路の長さが計算できることが知られている。

このアルゴリズムを利用し，ユーザがク

リックで指定したノードを始点として経路長計算を行う。

6-2 特定２ノード間の最短経路計算

6-1 節と同様にして，ダイクストラのアルゴリズムでは特定２ノード間の最短経路を計算することができる。

このアルゴリズムを利用して，ユーザがクリックで指定した２ノード間の最短経路を計算する。

6-3 全ノード間の最短経路長計算

6-1 節で行った計算を，始点となるノードを変更しながら繰り返し実行することにより，最終的にグラフ上のあらゆる２ノード間の最短経路長を計算することができる。

ユーザはキャンバス外の特定のボタンをクリックすることによりこの処理を実行できるようにする。

6-4 ワーシャル・フロイド法

6-1 節に示したダイクストラのアルゴリズムによる最短経路長計算では，グラフ上のノード数を n としたとき，計算量は$O(n^2)$であるから，最短経路長計算を，始点を変えながら n 回繰り返すことにより全ノード間の最短経路長を求める 6-3 節の計算では計算量が$O(n^3)$となる。一方，6-3 節の計算を行うためのアルゴリズムとして知られているワーシャル・フロイド法を用いる場合も，計算量を$O(n^3)$として処理できる。

6-5 アルゴリズムの実行速度の比較

6-3 節の計算をダイクストラ法，ワーシャル・フロイド法のそれぞれで行った場合の実行速度を比較する。以下にその方法を説明する。

まず，半径 1000 の円上のランダムな座標に一定数のノードを追加し，それらをランダムに結ぶエッジをノードと同じ数だけ追加することにより地図データを生成する。その後，その地図データにおいて 6-3 節の計算を２つのアルゴリズムのそれぞれで実行し，その実行時間を計測する。そしてこの地図データの生成と実行時間計測を 50 回繰り返し実行し，実行時間の平均値を求める。

Table.1 Average of run time in two algorithm with different number of nodes and edges

Number of nodes and edges	Run time in Dijkstra's algorithm[ms]	Run time in Warshall-Floyd algorithm[ms]
20	0.553	0.037
40	5.481	0.259
80	48.981	1.973
160	546.945	15.587

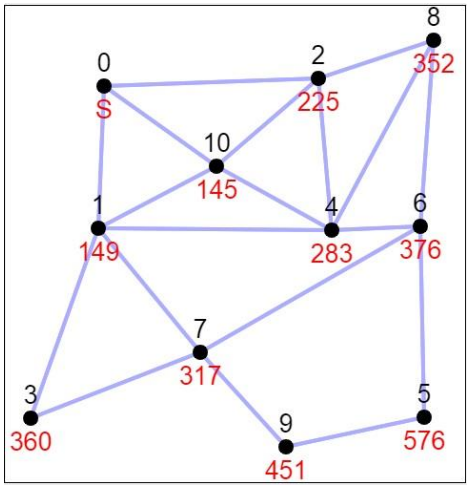

Fig.8 Visual display of the calculated length of shortest routes

結果として，ワーシャル・フロイド法での計算がより高速であった（Table. 1）。

7 経路計算結果の出力

7-1 特定１ノードから全ノードへの最短経路長

6-1 節に示した計算の結果を，キャンバス上の始点ノードの下に“S”を，他の各ノードの下に始点ノードからの最短経路長の値を表示することにより提示する（Fig. 8）。

7-2 特定２ノード間の最短経路

6-2 節に示した計算の結果を，最短経路に含まれるエッジをハイライト表示する。
それと同時に，経路の始点ノードの下に“S”を，それ以外の最短経路上にある全ノードの下に，始点から各ノードまでの経路長を表示することにより提示する（Fig. 9）。

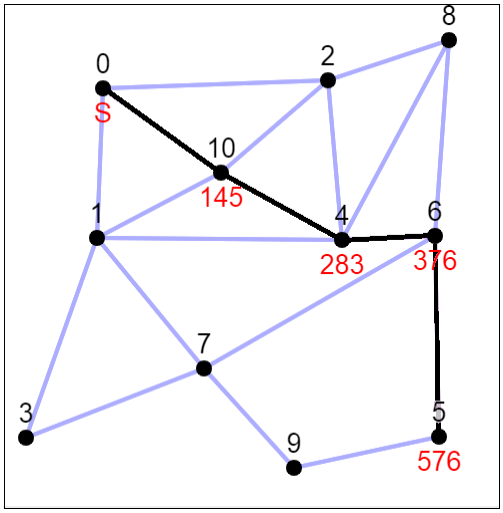

Fig.9 Visual display of the calculated shortest route

F-T	0	1	2	3	4	5	6	7	8	9	10
0	0	149	225	360	283	576	376	317	352	451	145
1	149	0	281	211	245	450	338	168	408	302	140
2	225	281	0	492	160	453	253	449	127	583	141
3	360	211	492	0	456	473	457	191	619	325	351
4	283	245	160	456	0	293	93	359	226	441	138
5	576	450	453	473	293	0	200	282	396	148	431
6	376	338	253	457	93	200	0	266	196	348	231
7	317	168	449	191	359	282	266	0	462	134	308
8	352	408	127	619	226	396	196	462	0	544	268
9	451	302	583	325	441	148	348	134	544	0	442
10	145	140	141	351	138	431	231	308	268	442	0

Fig.10 Length of shortest route between any two nodes

7-3 全ノード間の最短経路長

6-3 節に示した計算の結果を，始点ノードの番号を行に，終点ノードの番号を列にとった表に各２ノード間の最短経路長を示すことで提示する（Fig. 10）。

８　複数の地図データの比較

8-1 地図データの比較処理

ユーザにより作成された地図データを複数取り扱い，そのそれぞれにおいて経路計算を実行し，その結果を比較することによって複数の地図データの比較を可能にする。

8-2 複数の地図データの読み込み

4 節に示した操作によって作成した地図データを 3-3 節に示した形で文字列化したものを複数並列して入力することにより複数の地図データを読み込むことができるようにする（Fig. 11）。

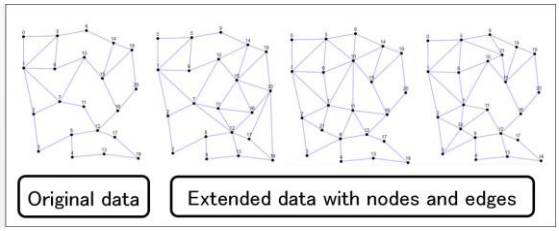

Fig.11 Example of the graphs on which shortest routes can be compared

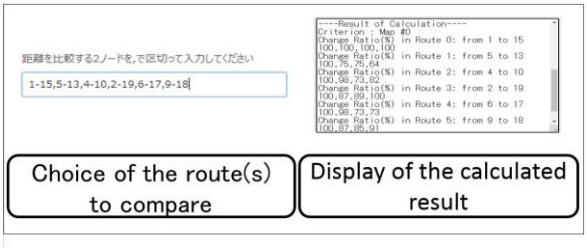

Fig.12 Interface for choosing routes to compare and displaying the calculated result

8-3 最短経路長の比較

ユーザがノードの番号によって指定した特定の２ノード間の最短経路長を，入力された複数の地図データのそれぞれにおいて計算する。

計算された各地図データ上での最短経路長の値を単に提示するほか，ユーザが指定した基準とする地図データでの値に対する差または割合を計算し提示できるようにする（Fig. 12）。

９　プログラムの利用例

既存のデータベース上に存在しない地図データの作成と，その地図データ上での最短経路の情報提示を可能にする目的に従い，プログラム利用の一例として岩手県立一関第一高等学校・附属中学校の校舎内の地図データを作成する（Fig. 13）。

9-1 地図データの入力

背景画像としてキャンバスに校舎の平面図を表示した上から，廊下の各教室の扉の前にノードを配置し，通行可能な部分にエッジを作成する。このとき，異なる階を繋ぐ階段部

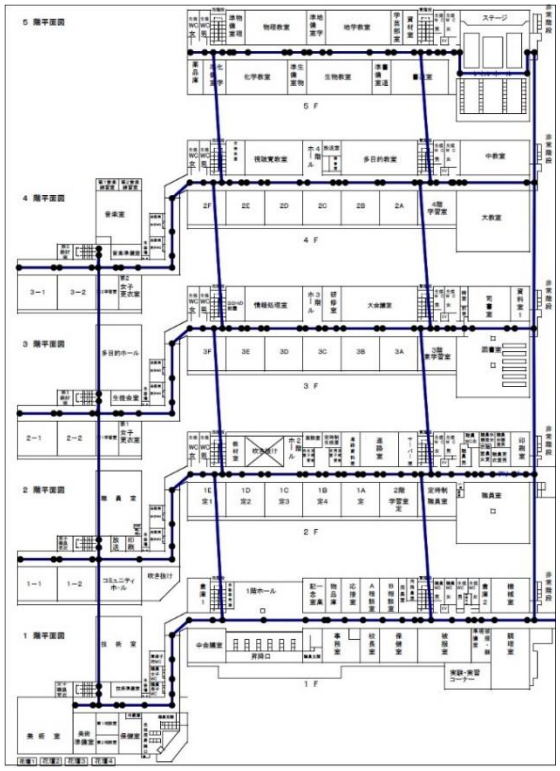

Fig.13 Input data of the map of Ichinoseki First S.H.S. and combined J.H.S.

分のエッジについては，平面画像上の距離が現実に通行する場合の距離に等しくないため，エッジのコストを修正する。

9-2 経路計算の実行

作成した地図データ上で最短経路や最短経路長を計算することができる。

9-3 地図データの比較計算

作成した地図データから部分的にノード，エッジを取り除くことにより，災害時に建物設備の破損・倒壊などにより通行できない場所がある場合の最短避難経路を検証することができる。

１０　考察・課題

10-1 地図データの入力処理

4 節及び 5 節に示したような地図データの入力方法によって，ユーザがデータベース化されていない地図について地図データを作成することができると考えられる。一方，平面図上に複数の階層を表現できる。9 節での例のような限られた場合を除いては，建物の内部や高低差のある地形などの立体的な地図を扱うことが困難であるため，そのような３次元の地図データを処理できる入力方法が必要であると考えられる。

10-2 地図データにおける経路計算処理

6 節に示した経路計算の実行により，作成された地図データにおける正確な最短経路を計算することができると考えられる。

大規模な地図データを作成して取り扱う場合には，経路計算のアルゴリズムの高速化が必要になることが予想されるが，6-5 節に示したように，ユーザが開発したプログラムを利用して入力する規模の地図データでは，ダイクストラ法およびワーシャル・フロイド法を使用することにより，十分に短い実行時間で経路計算を行うことができると考えられる。

10-3 経路計算結果の表示

7 節に示した計算結果の表示方法により，経路計算の結果をユーザが視覚的に理解できる形式で表示できると考えられる。

10-4 地図データの比較処理

複数の地図の比較によって，最短経路や最短経路長の変化を検証することができると考えられる。

このような機能によって，9-3 節で示したような災害時の避難経路の検証，または都市の交通網の改善案の提示などができると考えられる。

１１　結論

地図データの入力，最短経路の計算，計算結果の出力からなるプログラムによって，データベース化されていない地図データにおいて，最短経路情報の取得や災害時の避難経路等の検証を容易にすることができる。

謝辞

本研究に取り組むにあたってご指導していただいた柿崎朗先生に厚く感謝申し上げます。

参考文献

E. W. Dijkstra (1959): A Note on Two Problems in Connexion with Graphs, *Numerische Mathematik,* 1: 269-271

Stephen Warshall (1962): A theorem on Boolean matrices, *Journal of the ACM*, 9(1): 11-12

Robert W. Floyd (1962): Algorithm 97: Shortest Path, *Communications of the ACM*, 5(6): 345