計算機科学実験及演習2
ソフトウェア報告書1

谷　勇輝

入学年 平成27年
学籍番号 1029272870

提出日: 2016年10月11日

1　課題1

1.1　実施内容

1. ステージパラメータ（seed、難易度、敵の有無、その他地形等の変数）と既存エージェントを様々に変更し、動作を観察した。

2.ch.idsia.agents.controllers パッケージ内に用意された Agent プログラムのコードと動作を見比べた。

3. 上記エージェントプログラムに様々なログ (System.out.print) を加え変数の変化を観察した。

1.2　実行結果

1. それぞれのパラメータについて意味を理解した。また、概ね自分の望むタイプのコースを用意できるようになった。

2. マリオの動作を決定する配列について理解した。各エージェントプログラムの実装の趣旨と、その方法について理解を深めた。

3. マリオの情報についての変数に対しての理解を深めた。

1.3　結論と考察

2　課題 2

2.1　実施内容

2.2　実行結果

2.3　結論と考察

1. 始点の距離を 0 に、それ以外の頂点の距離を INF に初期化する。

2. 始点をキューに加える。

3. キューが空であれば到達不能、アルゴリズムを終了する。

4. 最短距離をもつ頂点 V をキューから取り出す
実際のコードでは java.util.PriorityQueue を用いている。このクラスはリストをヒープで管理し、要素の自然順序付けに従い最小の要素を取得できる。

5. V が終点なら V の距離が最短距離であり、アルゴリズムを終了する。

6. V が既に最短路確定済みなら 3 へ戻る。
このコードの実装では同一頂点が複数個キューに入ることが起こりうるので、既に探索済みの頂点は破棄する。

7. 取得した頂点がもつ辺 E を全て取得し、それぞれについて 8-9 を行う。

8. E で到達できる頂点 V_2 について距離の更新を行う。
距離が (V の距離)+(E の重み) よりも遠ければこの値に更新する。同じか近ければ更新は行わない。

9. 7 で更新が行われた時、V_2 をキューに追加する。

10. V を最短路確定済みとし、3 に戻る。

2.4　アルゴリズムの流れ図

2.5 実行例

```
$ java MyGraphTest
le1 ex12 ≪最短経路の探索≫
ファイル名を入力してください:test.in
始点の頂点番号を入力してください:0
終点の頂点番号を入力してください:4

【頂点 0 から頂点 4 までの最短経路】
経路: 0 1 11 12 5 4
経路長: 11

< 続けて探索する場合は"o"を入力 >
< 終了する場合はそれ以外のキーを入力 >
:o
始点の頂点番号を入力してください:4
終点の頂点番号を入力してください:9

【頂点 4 から頂点 9 までの最短経路】
経路: 4 5 12 13 9
経路長: 6

< 続けて探索する場合は"o"を入力 >
< 終了する場合はそれ以外のキーを入力 >
:fin
```

3 クラス仕様

3.1 MyGraphTest クラス

3.1.1 役割

main() メソッドを持ち、プログラムの起点となる。入力から対象のファイル名、探索の始点、終点を受け取り、他の 3 つのクラスを利用して最短経路を出力する。

3.1.2 メンバ変数

static boolean fileFoundFlag
ファイルの入力が既に正しく完了しているかどうかを示す。

static MyGraph graph
入力ファイルを元に構築した Graph クラスのオブジェクトを保持する。

static int[] inputVs

大きさ 2 の配列で、入力された始点と終点の番号を保持する。

static Scanner io

標準入力から入力を受け取る Scanner クラスのオブジェクトを保持する。

3.1.3 main() メソッド

機能

プログラムの起点となる。他のクラスとメソッドを使い、最短経路探索の実行の一連の動作を行う。

インタフェース

1. 入力 String[] args
   実行の際のオプションを受け付ける。今回のプログラムでは使用しない。

3.1.4 input() メソッド

機能

一連の入力を受け付け、探索に必要な値とクラスをメンバ変数に準備する。必要であればエラーを標準出力に表示する。

インタフェース

1. 出力 int[]
   static int[] inputVs をそのまま返す。他の課題のプログラムをそのまま利用している関係で残してあるが、本プログラムでは inputVs をそのまま利用しているため不要。

3.1.5 output() メソッド

機能

探索結果を標準出力に表示する。

インタフェース

1. 入力 LinkedList<Integer> list
   最短経路で通過する頂点列を順に保持した list を与える。探索結果の経路の表示に用いる。
2. 入力 int far
   最短経路長を表す整数を与える。探索結果の経路長の表示に用いる。

3.2 MyGraph クラス

3.2.1 役割

MyEdge クラスのオブジェクトを用いてグラフを表現する。メソッドによって最短経路探索を行う。

3.2.2 メンバ変数

static final int MAX_VERTEX_NUM
入力の際に受け付ける頂点数の最大値を保持する。

static final int MAX_EDGE_NUM
入力の際に受け付ける辺の数の最大値を保持する。

static final int MIN_COST
入力の際に受け付ける辺の重みの最小値を保持する。

static final int MAX_COST
入力の際に受け付ける辺の重みの最大値を保持する。

int V
このグラフが持つ頂点の数を保持する。

int E
このグラフが持つ辺の数を保持する。

ArrayList<LinkedList<MyEdge>> graph
各頂点から伸びる辺を格納した list を保持する。

boolean complete
このグラフが正しく構成されているかどうかを示す。

3.2.3 ReadFromFile() メソッド

機能
与えられたファイルを読みグラフを構成する。必要であれば標準出力にエラーを表示する。

インタフェース

1. 入力 String filename
   グラフを構成する際に参照する外部ファイルの名前を与える。

3.2.4 isPrepared() メソッド

機能
このグラフが正しく構成済みであるかどうかを調査する。

インタフェース

1. 出力 boolean
   メンバ変数 complete の値を返す。

3.2.5 getEdges() メソッド

機能

このグラフで、ある頂点から伸びている全ての辺を検索する。

インタフェース

1. 入力 int id
   検索する頂点番号を与える。
2. 出力 LinkedList<MyEdge>
   指定の頂点から伸びている辺を表す MyEdge クラスのオブジェクトの list を返す。

3.2.6 getV() メソッド

機能

このグラフの総頂点数を確認する。

インタフェース

1. 出力 int
   メンバ変数 V を返す。

3.2.7 getE() メソッド

機能

このグラフの総辺数を確認する。

インタフェース

1. 出力 int
   メンバ変数 E を返す。

3.2.8 isValidIndex() メソッド

機能

指定された整数がこのグラフの頂点番号として適切かどうかを調査する。

インタフェース

1. 入力 int index
   調査する整数を与える。
2. 出力 boolean
   与えられた整数が頂点番号の範囲内であれば true、そうでなければ false を返す。

3.2.9 waySearch() メソッド

機能

ある始点からある終点までの最短経路を探索する。

インタフェース

1. 入力 int startV
   探索の始点の頂点番号を与える。
2. 入力 int goalV
   探索の終点の頂点番号を与える。
3. 出力 LinkedList<Integer>
   探索結果の最短経路を示す頂点番号列の冒頭に、最短経路長を示す整数をつけた list を返す。すなわち、この list の 0 番目には最短経路長が、1 番目以降には最短経路の頂点列が格納されている。

3.3 MyEdge クラス

3.3.1 役割

一本の辺を表す。

3.3.2 メンバ変数

int v1

この辺の一端の頂点番号を保持する。

int v2

この辺のもう一端の頂点番号を保持する。

int cost

この辺の重みを保持する。

3.3.3 getCost() メソッド

機能

この辺の持つ重みを確認する。

インタフェース

1. 出力 int
   メンバ変数 cost の値を返す。

3.3.4 getOpponentV() メソッド

機能

ある頂点に対して、この辺のもう一端にある頂点番号を確認する。

インタフェース

1. 入力 int myV
   この辺の一端の頂点番号を与える。
2. 出力 int
   この辺のもう一端の頂点番号を返す。

3.4 MyVertex クラス

3.4.1 役割

一つの頂点を表し、キューの要素となる。頂点がキューから取り出される際の順序付けを規定する。

3.4.2 実装 Interface

1. java.lang.Comparable<MyVertex>
   このクラスには自然順序付けが定義される。

3.4.3 メンバ変数

int index

この頂点の頂点番号を保持する。

int far

最短路探索の際に使用され、始点からの現在の最短距離を保持する。
この値は探索が進むに連れて更新されていく。

int fromV

最短路探索の際に使用され、この頂点に至る現在の最短経路の 1 つ前の頂点の番号を保持する。この値は探索が進むに連れて更新されていく。

boolean complete

最短路探索の際に使用され、この頂点へ至る最短経路が確定したかどうかを保持する。

3.4.4 setFar() メソッド

機能

この頂点の始点からの最短距離を更新する。

インタフェース

1. 入力 int
   新たな最短距離を与える。

3.4.5 setFromV() メソッド

機能

最短路でこの頂点の 1 つ前の頂点が何番であるかを更新する。

インタフェース

1. 入力 int
   新たな 1 つ前の頂点の番号を与える。

3.4.6 setComplete() メソッド

機能

この頂点への最短経路が確定したかどうかを更新する。

インタフェース

1. 入力 boolean
   確定したなら true、未確定なら false を与える。

3.4.7 getIndex() メソッド

機能

この頂点の番号を確認する。

インタフェース

1. 出力 int
   メンバ変数 index を返す。

3.4.8 getFar() メソッド

機能

この頂点へ至る現最短経路での経路長を確認する。

インタフェース

1. 出力 int
   メンバ変数 far を返す。

3.4.9 getFromV() メソッド

機能

この頂点に至る現最短経路での 1 つ前の頂点の番号を確認する。

インタフェース

1. 出力 int
   メンバ変数 fromV を返す。

3.4.10 isComplete() メソッド

機能

この頂点の最短経路が確定しているかどうかを確認する。

インタフェース

1. 出力 boolean
   メンバ変数 complete を返す。

3.4.11 neededFarRenewal() メソッド

機能

ある最短路候補が示された時、この頂点への最短路を更新する必要があるかどうかを示す。

インタフェース

1. 入力 int far2
   示された最短路候補でのこの頂点への経路長を与える。
2. 出力 boolean
   最短路更新の必要があれば true、そうでなければ false を返す。

3.4.12 compareTo() メソッド

機能

他の MyVertex と自分とを比べ、どちらがどれだけ大きいかを示し、このクラスの自然順序付けを定義する。Comparable Interface に属する。

インタフェース

1. 入力 MyVertex v
   比較する他の頂点を示す MyVertex を与える。
2. 出力 int
   比較先にくらべ、自分がどれだけ大きいかを返す。
   自分のほうが大きければ正、小さければ負、等しければ 0 となる。

4　プログラムの評価

4.1　設計について

頂点を表すクラス MyVertex をつくり、これをリストの要素とする設計をした。PriorityQueue を利用するには要素のクラスに Comparable インターフェースが実装され自然順序が付けられるか、コンストラクタ実行時に Comparator インターフェースを実装したクラスを引数として渡す必要がある。今回のプログラムでは MyVertex に自然順序を定義することでキューから最小の距離を保持している頂点を取り出すよう設計した。代替案である Comparator インターフェースを用いる設計は、キューに入れるクラスに既存のクラスを使う場合には便利であるが、自ら新たにクラスを作成して管理する場合は今回の方法の方が、「頂点が距離順に並ぶ」という、より自然な概念に基づいてコーディングできると考えた。

辺のデータは MyEdge クラスで表現し、二重リストで保持する設計にしている。内側のリストは羅列することを考慮し LinkedList に、外側のリストは検索に特化するため ArrayList とした。

4.2　創意工夫した点

プログラムの制作ではクラス構造とオブジェクト指向を意識した。ただ数式や処理を羅列するのではなく、それぞれのクラスやメソッドが独立し、それが組み合わさる構造を心がけ、美しいコードが作成できるよう挑戦した。様々なエラーケースに対応した他、実用を考え連続で探索できるインターフェースを設計したところなども工夫した。

4.3　テストについて

プログラム作成後、様々なグラフを紙の上に書いた後データ化し、実際の最短経路と同様の出力が得られるかを確認した。2 頂点間に複数の辺がある場合、到達できない点がある場合、自分から自分に向かう辺がある場合など、特殊なケースについてもきちんと動作することを確認した。

4.4　バグについて

大きなバグの発生はなかった。処理を羅列するのではなく、メソッドの集合体として考えることでバグの発生しにくいコーディングが出来ていた成果であると思う。

4.5　効率について

時間的なオーダーはもちろんのこと、無駄なメモリを消費しないよう心がけた。全体として、効率の良いコーディングができたと思う。

4.6　機能的完成度

以上より機能的完成度は高く、与えられた仕様に対して十分な性能を発揮していると言える。一部メンバ変数の置き方など改良できる点はあるが、仕様に従おうとするならば難しいかもしれない。

5　プログラム開発の経過

1. 問題の分析と解法の検討 (10 分程度)
2. クラス設計 (30 分程度)
3. クラス内論理設計/プログラミング (140 分程度)
4. プログラムテスト，デバッグ (50 分程度)
5. 仕様書の作成 (720 分程度)

6　感想

オブジェクト指向を意識し、一つのプログラムが組み上がっていく感覚に手応えを感じた。メソッドの引数の与え方やメンバの持ち方などを試行錯誤し、より美しく機能的なコードを書けるよう努力していこうと思う。また、Latex による仕様書はまだまだ上手く書けていないと感じるので、自分でコードを書く際にも仕様書を書くことを心がけ、今よりもクオリティの高い仕様書を短時間で作成できる力を身に付けていきたいと感じた。より素早く正確にものづくりが出来るようにになりたい。