離散数学入門 最終レポート課題

1w192098-0 鏑木瑛司

2021 年 7 月 11 日

(1) この授業で学んだアルゴリズム（あるいはそれに類するアルゴリズム）を使って解決できる現実的な課題の例を一つ挙げて，分かりやすく説明してください（必要に応じて背景知識の説明や図表を適宜含めること）．

ある建物 (例えばショッピングモール) の中で, 現在地から目的地まで到達する最短経路と所要時間を建物の利用者に示すサービスを作ることを考える. 例えば, 図 1 のような建物 (ショッピングモール) 内の地図が与えられ,A 店から B 店まで行く最短経路と所要時間を知りたいとする. この問題に対し図 2 のような最短経路と所要時間 (例えば 3 分) を出力する.

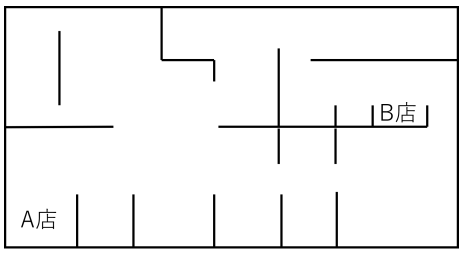

図 1.建物内の地図の例

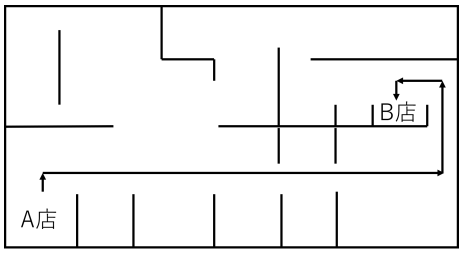

図 2.図 1 に対する最短経路の出力例

(2) (1) で挙げた現実的な課題が，数学的にはどのような問題として捉えられるかを簡潔に述べてください（何が入力として与えられ，何を出力する問題なのかを明記してください．問題に名称がついている場合は名称も述べてください）．

簡単のため, 以下のような単純な構造の地図を考える. ただし, 実線は壁を表す.

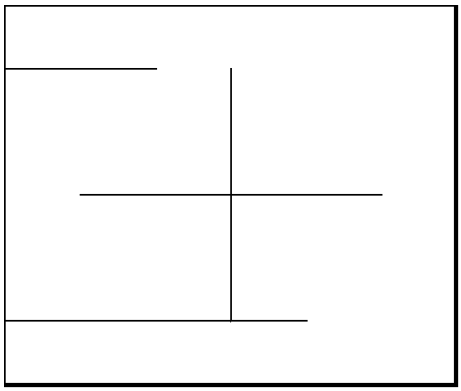

図 3.建物内の地図

次に, 図 3 の地図を, 平面上に以下のように表現する. ここで, 黒い部分は通れない場所（壁など）, 白い部分は通れる場所を表す.

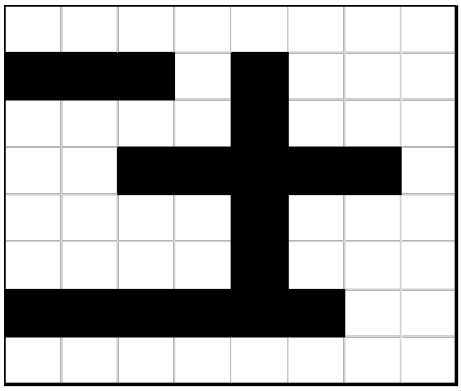

図 4.図 3 の地図をマス目状にして表した図

次に, 図 4 の白いマス目を頂点, 隣り合う白いマス目の間に重み 1 の辺があると考え, 図 4 をグラフ化する (図 5).

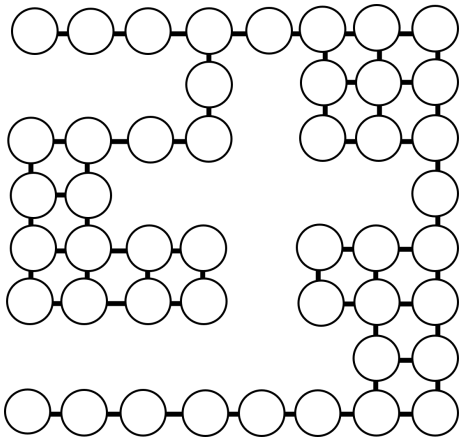

図 5.図 4 をグラフ化した図

建物の地図とスタート地点, ゴール地点を与えられた場合の最短経路と所要時間を求める本問題は, 図 5 のようなグラフとグラフ上のある 2 頂点を与え, その 2 頂点を結ぶ道の中で重みが最小のものとその重みを出力する問題ととらえることが出来る.

(3) (2) で述べた問題を解くために用いるアルゴリズムを一つ挙げ，アルゴリズムを正確に記述してください．そのアルゴリズムを選んだ根拠や妥当性などの特記事項があれば適宜補足してください．

(2) の問題を解くには深さ優先探索アルゴリズムを用いる方法が最良の手段であると考える. 幅優先探索は以下のようなアルゴリズムで各頂点の始点からの距離を求める.

(i) 始点を一か所設定する. 始点は, 始点からの距離を 0 とし記録する. 現在地を始点とし, 始点から探索を開始する.

(ii) まだ未探索の現在地に隣接する頂点の距離を, 現在地の距離に 1 を加えて記録する. 現在地の距離が n の場合, まだ未探索の現在地の隣接する頂点の距離は n+1 となる.

(iii) 現在地に隣接するすべての頂点の距離が記録されたら, 現在地と同じ距離のほかの頂点について (ii) と同様の操作を行う.

(iv) 現在地を, 探索済みの頂点の中で最も距離が大きい頂点のうちの一つに変える.

(v)(ii)〜(iv) を繰り返す. 到達可能なすべての頂点を訪問する, または始点から

の距離を求めたい頂点 (今回の問題では終点) を訪問したら探索を終了する.

幅優先探索は過去訪問した頂点の始点からの距離をすべて記録することが特徴である.2 頂点間の経路を求めるアルゴリズムに深さ優先探索も存在するが, 深さ優先探索によって得られた経路はそれが最短経路であるとは限らない. 一方幅優先探索では, 最短経路が存在すればそれを確実に見つけることが可能である (以下に最短経路を見つけるアルゴリズムについて記述する). よって, 本問題を解くには幅優先探索を用いるのが最適である.

続いて, 始点から終点への最短経路となるグラフ上の道を決定する方法について以下に記述する. 始点からの距離は幅優先探索で求めた値を用いる.
(i) 終点を現在地とし, 経路の探索を開始する.
(ii) 現在地の始点からの距離を n と置く. 現在地に隣接する頂点の中で始点からの距離が n-1 の頂点を一つ選び, 記録する. 現在地を記録した頂点に移す.
(iii) 現在地の距離が 0 になるまで (ii) を繰り返す.
(iv) 記録した頂点を, 記録した順序の逆の順序でたどる道が始点から終点への最短経路 (のうちのひとつ) となる. 幅優先探索は途中で頂点の距離が別の値に上書きされることはないので, この手法で必ず経路が求まる.

本来, 幅優先探索でグラフ上に BFS 木を形成すれば, 距離が 1 ずつ減るように終点から始点まで頂点をたどれば最短経路が 1 つに定まるのだが, 今回作成したプログラム（後述）では BFS 木の辺の情報を記憶しないので, 上記の方法で経路の探索を行う.

所要時間は, 終点の始点からの距離とする.

(4) (3) で挙げたアルゴリズムを用いて (2) の問題の正しい答えを求めてください．アルゴリズムが入力を受け取ってから解を出力するまでの途中過程も簡潔に記述してください．

図 5 のグラフに対し,(3) で述べたアルゴリズムに従い各頂点の始点から終点までの距離を求めた結果は図 6 のとおりである. ただし, 赤色の頂点を始点, 青色の頂点を終点とした. 始点から探索を開始し, 隣接する点の始点からの距

離を順番にすべて求めていった.

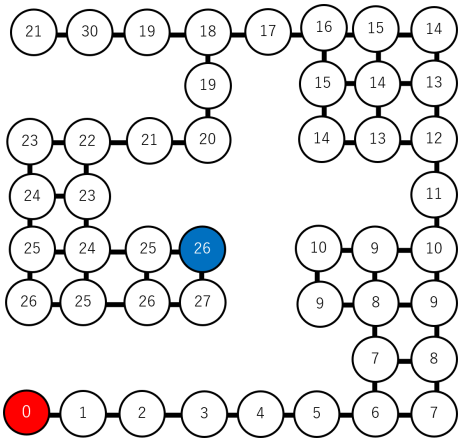

図 6.図 5 のグラフに対して幅優先探索を行った結果
(赤い頂点が始点, 青い頂点が終点を表す)

続いて, 図 7 は図 5 に対して幅優先探索を行った結果を利用して最短経路を求めた結果である. 赤い頂点を始点とし, 隣接する橙色の頂点を通り青色の終点まで行く道が最短経路を表す.(3) で述べたアルゴリズムに従い, 終点から開始して, 始点からの距離が 1 ずつ短くなるように始点までたどった.

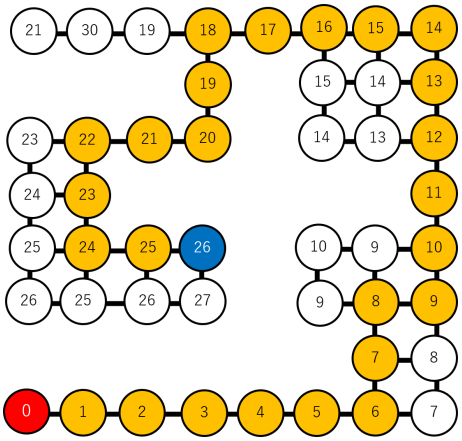

図 7.図 5 のグラフにおける最短経路

本問題では所要時間は始点から終点の距離で表すことにしているので, 図 5 に対する始点から終点までの所要時間は 26 である.

(5) [該当する希望者のみ] 現実のデータセットに対して (4) と同じことを行った計算機実験の結果などを自由に記してください．実験内容の詳細が分かるように，実験に使用したデータセット，自分で作成したプログラムがある場合はそのソースコード，実験の実行に必要な環境・ライブラリ等の情報も適宜含めてください．

建物の構造をテキストデータで表現し, そのテキストデータに対し最短経路と所要時間を出力するプログラムを python3 で作成した. ソースコードは以下のとおりである.

```
from collections import deque

def solve_maze(Maze):
  w=len(Maze[0])
  h=len(Maze)

  maze=[[] for i in range(h)]

  for i in range(h):
    for j in range(w):
      if Maze[i][j]=="S":
        s=[i,j]
      if Maze[i][j]=="G":
        g=[i,j]
      maze[i].append(Maze[i][j])
  print("以下の地図でスタートからゴールまでの最短経路を探索します")
  for i in range(h):
    for j in range(w):
      print(maze[i][j],end="")
    print(end="\n")
  print("")

  maze[s[0]][s[1]]=0

  queue=deque()
  queue.append(s)
  while len(queue):
    tmp=queue.popleft()
    x=tmp[1]
    y=tmp[0]

    for i,j in [[1,0],[0,1],[-1,0],[0,-1]]:
      if maze[y+i][x+j]==' ' or maze[y+i][x+j]=='G':
        queue.append([y+i,x+j])
        maze[y+i][x+j]=int(maze[y][x])+1

  x,y=g[1],g[0]
  n=maze[y][x]
  time=n
  if n=="G":
    print('この地図はゴールまで到達できません')
    print("\n=========================================\n")
    return
  maze[y][x]="G"

  while n>0:
    for i,j in [[1,0],[0,1],[-1,0],[0,-1]]:
      if maze[y+i][x+j]==0:
        maze[y+i][x+j]='S'
        n=0
        break
      if maze[y+i][x+j]==n-1:
        maze[y+i][x+j]='.'
        y+=i;x+=j;
```

```
        n-=1
        break

  for i in range(h):
    for j in range(w):
      if maze[i][j]!="#" and maze[i][j]!=" " and maze[i][j]!='.' and maze[i][
          j]!='S' and maze[i][j]!='G':
        maze[i][j]=' '
  print('最短経路は以下のとおりです')
  for i in range(h):
    for j in range(w):
      print(maze[i][j],end="")
    print(end="\n")
  print("所要時間は"+str(time)+"です")
  print("\n=========================================\n")



#地図の1番外側の壁を必ず書く
#地図は必ず長方形にする
#スタート(S)とゴール(G)を地図内に必ず1つずつ書く

Maze0=['############',
       '#G# # # #S#',
       '#          #',
       '############']

Maze1=['#########################',
      '#    #G  #    #   #     #',
      '#    #   #    #   #     #',
      '#                        #',
      '#   ################    #',
      '#                        #',
      '#  #   #    #     #     #',
      '#  #   #    #     #    S#',
      '#########################']

Maze2=['##################################################',
       '#     #      #                        #     #    G#',
       '#     #      #          #    #                   #',
       '#     #      #          #    #         #############',
       '#                       ######                   #',
       '###      #####          #    #         #         #',
       '#S       #              #    #         #         #',
       '###      #####          ######         #         #',
       '#        #                             #         #',
       '###      #####      ########           #         #',
       '#        #          ########           #         #',
       '##################################################']

Maze3=['##################################################',
       '#     #                        #         #     G#',
       '#     #                        #                 #',
       '#     #                        #         #############',
       '#                       ######                   #',
       '###      #####              #         #         #',
       '#        #                  #         #         #',
       '###      #####          ######         #         #',
       '#S       #                             #         #',
       '###      #####      ########           #         #',
       '#                   ########           #         #',
       '##################################################']


solve_maze(Maze0)
solve_maze(Maze1)
solve_maze(Maze2)
solve_maze(Maze3)
```

上記のプログラムに対する出力は以下のとおりである.

```
以下の地図でスタートからゴールまでの最短経路を探索します
###########
#G# # # #S#
#         #
###########

最短経路は以下のとおりです
###########
#G# # # #S#
#.........#
###########
所要時間は10です

=========================================

以下の地図でスタートからゴールまでの最短経路を探索します
#########################
#    #G  #    #   #     #
#    #   #    #   #     #
#                       #
#   ################    #
#                       #
#  #   #    #     #     #
#  #   #    #     #    S#
#########################

最短経路は以下のとおりです
#########################
#    #G  #    #   #     #
#    #.  #    #   #     #
#     ..............    #
#   ################.   #
#                   .   #
#  #   #    #     # .   #
#  #   #    #     # ...S#
#########################
所要時間は23です

=========================================

以下の地図でスタートからゴールまでの最短経路を探索します
#################################################
#     #      #                        #     #    G#
#     #      #          #    #                   #
#     #      #          #    #         ##########
#                       ######                   #
###       #####         #    #         #         #
#S       #              #    #         #         #
###       #####         ######         #         #
#         #                            #         #
###       #####      ########          #         #
#         #          ########          #         #
#################################################

最短経路は以下のとおりです
#################################################
#     #      #          ..............#     #    G#
#     #      #         .#    #       .............#
#     #      #         .#    #         ##########
#       ..............######                   #
###     .#####          #    #         #         #
#S......#              #    #         #         #
###       #####         ######         #         #
#         #                            #         #
###       #####      ########          #         #
#         #          ########          #         #
#################################################
所要時間は55です

=========================================

以下の地図でスタートからゴールまでの最短経路を探索します
```

```
73  ##################################################
74  #      #                         #         #     G#
75  #      #                         #                 #
76  #      #                         #         #############
77  #                              ######                #
78  ###       #####                  #         #         #
79  #         #                      #         #         #
80  ###       #####           ######           #         #
81  #S        #                                #         #
82  ###       #####       ########             #         #
83  #                     ########             #         #
84  ##################################################
85
86  最短経路は以下のとおりです
87  ##################################################
88  #      #                         #         #     G#
89  #      #                         #        ............#
90  #      #                         #        .############
91  #                              ######        .         #
92  ###       #####                  #         .#        #
93  #         #                      #         .#        #
94  ###       #####           ######           .#        #
95  #S.......#           ..................    .#        #
96  ###      .#####      .########             #         #
97  #        ...........########               #         #
98  ##################################################
99  所要時間は59です
100
101 ==========================================
```

本プログラムは, 建物内の構造を表したリストの入力に対し, 最短経路を上書きして出力し, さらに所要時間を出力するプログラムである. テキストデータは, 例えば以下のようなものを考える. これは (4) で用いた例と同じ構造である.

```
1   ##########
2   #        #
3   #    #   #
4   #    #   #
5   #  ##### #
6   #    #   #
7   #   G#   #
8   #######  #
9   #S       #
10  ##########
```

ここで, 空白は通行可能な場所,# は通行できない場所,S は始点,G は終点を表し,S から G への最短経路を求める. プログラム内で作成した関数 solve_maze() は, 建物内の構造を表すテキストデータを文字列の 1 次元配列で受け取り, 最短経路を二次元配列で出力する.

入力に対し幅優先探索を用いて各場所の始点からの距離を求める. 入力されたテキストデータの半角スペースの部分を各頂点, 隣り合っている空白はたがいに隣接している頂点と考え,S を始点,G を終点とし幅優先探索を実行する.

幅優先探索を以下の手順で行う.(ソースコード 27 行目から 35 行目)

(i) 始点の座標を両端キューに格納する. 始点の距離を 0 とし記録する.

(ii) 両端キューに格納されている座標を左側から一つ取り出す. この座標の始点からの距離が n であるとする. その座標に隣接するまだ未訪問且つ通行可能な座標をすべて探し, その座標を両端キューに右側から格納する. 今格納した

座標の始点からの距離を n+1 ととして記録する.
(iii) 両端キューが空になるまで (ii) を繰り返す.
上記のような幅優先探索を行った結果, 以下のように各頂点の始点からの距離が格納された二次元配列が得られる.

```
# #  #  #  #  #  #  #  #  #
# 21 20 19 18 17 16 15 14 #
# 22 21 20 19 #  15 14 13 #
# 23 22 21 20 #  14 13 12 #
# 24 23 #  #  #  #  #  11 #
# 25 24 25 26 #  10 9  10 #
# 26 25 26 27 #  9  8  9  #
# #  #  #  #  #  #  7  8  #
# 0  1  2  3  4  5  6  7  #
# #  #  #  #  #  #  #  #  #
```

上記の配列をもとに最短経路を求める (ソースコード 46 行目から 56 行目). そして, 求めた最短経路と S から G までの距離を出力する. 出力は, 入力で使用された文字に加え,'.' が最短経路を表している. なお, 終点にたどり着く経路が無い場合はその旨を出力する. 以下は (4) で扱った例と同じ構造の地図に対する結果である.

```
##########
#   ...  #
#   .#.  #
# ...#...#
# .#####.#
# ...#  .#
#   G#  .#
####### .#
#S.......#
##########
```

以上が与えられた地図の最短経路と所要時間を求めるプログラムの概要である. 本実験では４つの地図を入力として与えそれぞれに対する結果を出力している.

使用言語:python 3.8.2
使用ライブラリ:collections
実行環境:https://replit.com/

参考文献:なし