平成29年春 データ構造及びアルゴリズム 最短経路の探索

問8　最短経路の探索（データ構造及びアルゴリズム）　(H29春・FE 午後問8)

【解答】

［設問1］　a－イ，b－オ，c－キ，d－イ

［設問2］　e－イ，f－カ，g－ア

【解説】

N個（N＞1）の地点から構成されるグラフにおいて，「出発地から目的地に至る最短経路とその距離を求める」アルゴリズムの問題である。始点を決め，始点に隣り合っている地点から一つずつ最短距離を確定し，徐々に範囲を広げていき，最終的に目的地に至るまでの最短経路を求めるダイクストラ法に基づいたアルゴリズムである。

アルゴリズムの説明は問題文に示されているとおりだが，内容をしっかり理解してから解答を考えていては時間がかかってしまう。このような問題は，手順に従って，プログラムとの対応関係を見極めながらその処理内容を把握していくと解きやすい。〔プログラムの説明〕にはプログラムの行番号が示されているので，これを手掛かりに行番号の範囲内で内容を吟味すると理解が早いだろう。また，設問2の設問文と表3には，手順に従った配列と変数の値の変化が示されているので，これも参考になる。

プログラムは短く，処理範囲も〔プログラムの説明〕に行番号が示されており，空欄の穴埋めも決して難しくはないが，擬似言語のプログラムに出てくるbreak命令や繰返し処理の記述の仕方に戸惑った人もいるだろう。

break命令は，プログラム中に注釈があるので，そのとおりに解釈すればよい。つまり，実行中の繰返し処理を抜けるので，行番号16のbreak命令を実行すると行番号20へ，行番号21のbreak命令を実行すると行番号40へ処理を移動する。行番号16のbreak命令は二重ループの内側にあり，注釈の「最内側の繰返し」は，行番号14～19の繰返しを指す。行番号6，23，30の繰返し処理の記述については「共通に使用される擬似言語の記述形式」に記載があるので，「変数：初期値，条件式，増分」の意味を確認してから考えるとよい。

〔プログラムの説明〕に従って，図1のグラフの各地点の最短距離の求め方を確認する。グラフにおいて，丸は地点（地点番号）を，線分は地点間を結ぶ経路を，線分の上の数字はその距離を表している。図1の場合，地点0からは地点1，2，3に進むことができ，地点0から地点1，2，3への距離は，それぞれ2，8，4である。また，経路上は双方向に移動できるので，地点1から地点0に進むこともでき，その場合の距離も同じく2である。

〔プログラムの説明〕(1)，(2)から，使用する引数の意味は次のとおりである。

- Distance[][]：地点間の距離が格納されている2次元配列で，地点iから地点jまでの距離をDistance[i][j]と表す。表2に「図1の例における配列Distanceの内容」が表示されているが，地点0から地点1までの場合，距離はDistance[0][1]＝2である。双方向に移動できるのでDistance[1][0]も2である。また，地点0から地点4のように地点iから地点jまで直接の経路がない場合は－1，2点が同一の地点の場合は0が格納されている。
- nPoint：地点数を表し，図1では7である。
- sp：出発地の地点番号を表し，地点0を出発地とする場合は0である。
- dp：目的地の地点番号を表し，目的地が地点6の場合は6である。
- sDist：出発地から目的地までの最短距離を求めるための変数で，初期値として∞（最大値を表す定数）が設定されている。〔プログラムの説明〕(3)も参照。
- sRoute[]：出発地から目的地までの最短経路上の地点の地点番号を，目的地から出発地までの順に設定する配列で，初期状態は全ての要素に地点番号が設定されていないことを意味する－1が設定されている。

	0	1	2	3	4	5	6
sRoute	－1	－1	－1	－1	－1	－1	－1

〔プログラムの説明〕(3)～(6)から，使用する配列の意味は次のとおりである。

- pDist[]：出発地から各地点までの最短距離を設定する配列で，初期状態は全ての要素に∞が設定されている（行番号8）。ただし，∞を設定後，行番号11で出発地から出発地自体への最短距離に0を設定していることに注意が必要である。これによって，地点0からの最短経路を求める場合，pDist[0]＝0となる。なお，配列の添字は地点番号と対応しており，例えば，pDist[4]には地点0から地点4の仮の最短距離が設定され，全ての地点が処理済みとなった時点で設定されている値が最短距離として確定する。図1ではpDist[4]＝5となる。
- pFixed[]：最短距離を求める過程で最短距離が確定している地点を識別する配列で，初期状態は全ての要素にfalseが設定されている（行番号9）。配列の添字は地点番号と対応しており，pFixed[　b　]がtrueになったとき，pDist[j]には地点0から地点jまでの最短距離が求められている。全ての要素がtrueになると全ての地点が処理済みとなる。
- pRoute[]：地点0から地点jの仮の最短距離を配列pDistに設定したときの，直前の経由地の地点番号を設定する配列である。配列の添字は地点番号と対応しており，図1でpDist[4]＝5と設定した場合，地点4は直前に地点1を経由しているので，pRoute[4]＝1となる。
- sPoint：出発地からの最短距離が未確定の地点中で，出発地からの距離が最も短い地点である。

最短経路を求める手順は，次のようになる。

- 行番号13～22：配列pFixedの要素がfalseの地点を求める。
- 行番号23～29：最短距離が未確定の他の地点と行番号13～22で求めた地点の距離を比較し，最も距離の短い地点をsPointに決定する。これによって，配列pFixedのsPointに指定された地点の要素がtrueになる。
- 行番号30～39：出発地から地点sPointを経由して地点jに到達する最短距離を決定する。sPointを経由しているので，行番号32が示すように，その計算式はpDist[sPoint]＋Distance[sPoint][j]であり，計算結果である距離はnewDistに格納される。このnewDistと既に計算済みの配列pDistの値を比較し，newDistの方が短ければ地点jまでの距離を仮の最短距離としてpDist[j]を更新し，このとき経由した直前の経由地の地点番号をpRoute[j]に設定する。
- 行番号40～48：出発地から目的地までの最短距離をsDistに，最短経路上の地点の地点番号を目的地から出発地までの順に配列sRouteに設定する。これについて，図1を使って実際にトレースして動作を確認したいところだが，配列pRouteの初期状態について，〔プログラムの説明〕にもプログラム中にも定義がない。断念して設問1にとりかかろう。

［設問1］

- 空欄a：既に確認したように，行番号23～29は，行番号13～22で求めた最短距離が未確定の地点と，それ以外の未確定地点の距離を比較し，最も距離の短い地点をsPointに決定する処理である。行番号14の注釈に「未確定の地点を一つ探す」とあり，配列pFixedの添字がiであることから，iに最初にみつかった未確定の地点番号が格納されていることが分かる。また，空欄aは，「最短距離がより短い地点を探す」ループの中の条件式の一部であり，条件式の残りの部分で距離の比較を行っていることから，未確定かどうかの判断と分かる。しかし，配列pFixedの初期値はfalseであり，条件式の優先順位を括弧で表すと（　a　）and（pDist[j]＜pDist[i]）である。つまり，空欄aが真かつpDist[j]がpDist[i]より小さいときに条件が成立し，地点iが地点jに更新される。このことから，pFixed[j]を否定したnot(pFixed[j])がここにあればよい。したがって，空欄aには，「not(pFixed[j])」（イ）が入る。

- 空欄b：注釈に「出発地からの最短距離を確定する」とあり，配列pFixedにtrueを代入している。行番号23～29では，sPointに指定された地点の最短距離が確定し，配列pFixedの地点sPointの要素も確定を意味するtrueになるので，配列pFixedの添字である空欄bには，「sPoint」（オ）が入る。

- 空欄c：〔プログラムの説明〕(6)に，行番号40～48では，最短距離をsDistに，最短経路を配列sRouteに設定することが記述されている。ここから，空欄cは，配列sRouteを作成する処理の一部と分かる。ここでは，行番号40で最短距離をsDistに格納した後，最短経路上の地点の地点番号を，目的地から出発地までの順に配列sRouteに設定する。そのため，行番号42では，ループ変数iに目的地の地点番号であるdpの値を代入して初期値とし，iがspになるまで繰り返している。繰返しの中でiを空欄cに代入しているが，図1の場合，1回目の繰返しではiの値は6であり，これが配列sRouteの先頭sRoute[0]に代入される。また，i＝spとなって繰返しを抜けた後，行番号48でsRoute[j]にspを代入することで，出発地の地点番号0が設定される。したがって，空欄cには，「sRoute[j]」（キ）が入る。

- 空欄d：繰返しの中でiの値を更新する処理である。配列sRouteには，目的地から出発地までの順に地点番号を格納するので，次のiの値は，地点iの直前に通過する地点の地点番号である。直前に通過する地点番号は行番号35で配列pRouteに設定しており，地点iの直前の地点はpRoute[i]に格納されている。例えば，地点6の場合，i＝6なので，直前の地点はpRoute[6]に格納されている。したがって，空欄dには，「pRoute[i]」（イ）が入る。

［設問2］

図1において，出発地の地点番号sp＝0，目的地の地点番号dp＝6の場合について，行番号28のαにおける sPoint の値，及び行番号39のβにおける配列 pDist と配列 pRoute の内容をトレースする。配列 pRoute の初期値について，〔プログラムの説明〕にもプログラム中にも定義がなかったが，設問文の最後に「全ての要素には初期値として0が格納されている」と記述されている。読み落とすとトレースが進まないので，注意したい。繰返しの3回目までをトレースすると次のとおりとなる。網掛け部分が更新された要素である。

初期状態

	0	1	2	3	4	5	6
pFixed	false	false	false	false	false	false	false
pDist	0	∞	∞	∞	∞	∞	∞
pRoute	0	0	0	0	0	0	0

1回目

配列pFixedの全ての要素がfalseなので，i＝0のとき，行番号15の判定が真となり，break命令が実行される。i≠nPointなので行番号20は偽となり，i＝0の状態で行番号23の繰返しを開始する。繰返し変数jは，地点iと同一の地点を比較しても意味がないので，i＋1を初期値としている。配列pDistは，pDist[0]のときだけ0，それ以外の要素は∞なので行番号24が真になることはなく，i＝0のまま繰返しを終了する。そして，行番号28でsPointに0が代入され，行番号29でpFixed[0]にtrueが設定される。

行番号30～39では距離の比較を行うが，行番号31の条件にnot(pFixed[j])があるので，j＝0では何も処理しない。j＝1では，not(pFixed[1])が真となり，Distance[0][1]＝2＞0なのでnewDistを計算する。pDist[0]＋Distance[0][1]＝0＋2＝2から，newDist＝2＜pDist[1]＝∞となり，行番号33が真となるので，pDist[1]を2，pRoute[1]を0に更新する。j＝2，3でもj＝1と同様に処理が実行され，配列pDistと配列pRouteが更新される。j＝4以上は，地点4，5，6が地点0と隣接していないため，行番号31でDistance[0][j]＝－1＜0となり，処理を実行しない。

	0	1	2	3	4	5	6
pFixed	true	false	false	false	false	false	false
pDist	0	2	8	4	∞	∞	∞
pRoute	0	0	0	0	0	0	0

平成29年春 データ構造及びアルゴリズム 最短経路の探索

問8　最短経路の探索（データ構造及びアルゴリズム）　(H29春・FE 午後問8)

【解答】

［設問1］　a－イ，b－オ，c－キ，d－イ

［設問2］　e－イ，f－カ，g－ア

【解説】

N個（N＞1）の地点から構成されるグラフにおいて，「出発地から目的地に至る最短経路とその距離を求める」アルゴリズムの問題である。始点を決め，始点に隣り合っている地点から一つずつ最短距離を確定し，徐々に範囲を広げていき，最終的に目的地に至るまでの最短経路を求めるダイクストラ法に基づいたアルゴリズムである。

アルゴリズムの説明は問題文に示されているとおりだが，内容をしっかり理解してから解答を考えていては時間がかかってしまう。このような問題は，手順に従って，プログラムとの対応関係を見極めながらその処理内容を把握していくと解きやすい。〔プログラムの説明〕にはプログラムの行番号が示されているので，これを手掛かりに行番号の範囲内で内容を吟味すると理解が早いだろう。また，設問2の設問文と表3には，手順に従った配列と変数の値の変化が示されているので，これも参考になる。

プログラムは短く，処理範囲も〔プログラムの説明〕に行番号が示されており，空欄の穴埋めも決して難しくはないが，擬似言語のプログラムに出てくるbreak命令や繰返し処理の記述の仕方に戸惑った人もいるだろう。

break命令は，プログラム中に注釈があるので，そのとおりに解釈すればよい。つまり，実行中の繰返し処理を抜けるので，行番号16のbreak命令を実行すると行番号20へ，行番号21のbreak命令を実行すると行番号40へ処理を移動する。行番号16のbreak命令は二重ループの内側にあり，注釈の「最内側の繰返し」は，行番号14～19の繰返しを指す。行番号6，23，30の繰返し処理の記述については「共通に使用される擬似言語の記述形式」に記載があるので，「変数：初期値，条件式，増分」の意味を確認してから考えるとよい。

〔プログラムの説明〕に従って，図1のグラフの各地点の最短距離の求め方を確認する。グラフにおいて，丸は地点（地点番号）を，線分は地点間を結ぶ経路を，線分の上の数字はその距離を表している。図1の場合，地点0からは地点1，2，3に進むことができ，地点0から地点1，2，3への距離は，それぞれ2，8，4である。また，経路上は双方向に移動できるので，地点1から地点0に進むこともでき，その場合の距離も同じく2である。

〔プログラムの説明〕(1)，(2)から，使用する引数の意味は次のとおりである。

- Distance[][]：地点間の距離が格納されている2次元配列で，地点iから地点jまでの距離をDistance[i][j]と表す。表2に「図1の例における配列Distanceの内容」が表示されているが，地点0から地点1までの場合，距離はDistance[0][1]＝2である。双方向に移動できるのでDistance[1][0]も2である。また，地点0から地点4のように地点iから地点jまで直接の経路がない場合は－1，2点が同一の地点の場合は0が格納されている。
- nPoint：地点数を表し，図1では7である。
- sp：出発地の地点番号を表し，地点0を出発地とする場合は0である。
- dp：目的地の地点番号を表し，目的地が地点6の場合は6である。
- sDist：出発地から目的地までの最短距離を求めるための変数で，初期値として∞（最大値を表す定数）が設定されている。〔プログラムの説明〕(3)も参照。
- sRoute[]：出発地から目的地までの最短経路上の地点の地点番号を，目的地から出発地までの順に設定する配列で，初期状態は全ての要素に地点番号が設定されていないことを意味する－1が設定されている。

	0	1	2	3	4	5	6
sRoute	－1	－1	－1	－1	－1	－1	－1

〔プログラムの説明〕(3)～(6)から，使用する配列の意味は次のとおりである。

- pDist[]：出発地から各地点までの最短距離を設定する配列で，初期状態は全ての要素に∞が設定されている（行番号8）。ただし，∞を設定後，行番号11で出発地から出発地自体への最短距離に0を設定していることに注意が必要である。これによって，地点0からの最短経路を求める場合，pDist[0]＝0となる。なお，配列の添字は地点番号と対応しており，例えば，pDist[4]には地点0から地点4の仮の最短距離が設定され，全ての地点が処理済みとなった時点で設定されている値が最短距離として確定する。図1ではpDist[4]＝5となる。
- pFixed[]：最短距離を求める過程で最短距離が確定している地点を識別する配列で，初期状態は全ての要素にfalseが設定されている（行番号9）。配列の添字は地点番号と対応しており，pFixed[　b　]がtrueになったとき，pDist[j]には地点0から地点jまでの最短距離が求められている。全ての要素がtrueになると全ての地点が処理済みとなる。
- pRoute[]：地点0から地点jの仮の最短距離を配列pDistに設定したときの，直前の経由地の地点番号を設定する配列である。配列の添字は地点番号と対応しており，図1でpDist[4]＝5と設定した場合，地点4は直前に地点1を経由しているので，pRoute[4]＝1となる。
- sPoint：出発地からの最短距離が未確定の地点中で，出発地からの距離が最も短い地点である。

最短経路を求める手順は，次のようになる。

- 行番号13～22：配列pFixedの要素がfalseの地点を求める。
- 行番号23～29：最短距離が未確定の他の地点と行番号13～22で求めた地点の距離を比較し，最も距離の短い地点をsPointに決定する。これによって，配列pFixedのsPointに指定された地点の要素がtrueになる。
- 行番号30～39：出発地から地点sPointを経由して地点jに到達する最短距離を決定する。sPointを経由しているので，行番号32が示すように，その計算式はpDist[sPoint]＋Distance[sPoint][j]であり，計算結果である距離はnewDistに格納される。このnewDistと既に計算済みの配列pDistの値を比較し，newDistの方が短ければ地点jまでの距離を仮の最短距離としてpDist[j]を更新し，このとき経由した直前の経由地の地点番号をpRoute[j]に設定する。
- 行番号40～48：出発地から目的地までの最短距離をsDistに，最短経路上の地点の地点番号を目的地から出発地までの順に配列sRouteに設定する。これについて，図1を使って実際にトレースして動作を確認したいところだが，配列pRouteの初期状態について，〔プログラムの説明〕にもプログラム中にも定義がない。断念して設問1にとりかかろう。

［設問1］

- 空欄a：既に確認したように，行番号23～29は，行番号13～22で求めた最短距離が未確定の地点と，それ以外の未確定地点の距離を比較し，最も距離の短い地点をsPointに決定する処理である。行番号14の注釈に「未確定の地点を一つ探す」とあり，配列pFixedの添字がiであることから，iに最初にみつかった未確定の地点番号が格納されていることが分かる。また，空欄aは，「最短距離がより短い地点を探す」ループの中の条件式の一部であり，条件式の残りの部分で距離の比較を行っていることから，未確定かどうかの判断と分かる。しかし，配列pFixedの初期値はfalseであり，条件式の優先順位を括弧で表すと（　a　）and（pDist[j]＜pDist[i]）である。つまり，空欄aが真かつpDist[j]がpDist[i]より小さいときに条件が成立し，地点iが地点jに更新される。このことから，pFixed[j]を否定したnot(pFixed[j])がここにあればよい。したがって，空欄aには，「not(pFixed[j])」（イ）が入る。

- 空欄b：注釈に「出発地からの最短距離を確定する」とあり，配列pFixedにtrueを代入している。行番号23～29では，sPointに指定された地点の最短距離が確定し，配列pFixedの地点sPointの要素も確定を意味するtrueになるので，配列pFixedの添字である空欄bには，「sPoint」（オ）が入る。

- 空欄c：〔プログラムの説明〕(6)に，行番号40～48では，最短距離をsDistに，最短経路を配列sRouteに設定することが記述されている。ここから，空欄cは，配列sRouteを作成する処理の一部と分かる。ここでは，行番号40で最短距離をsDistに格納した後，最短経路上の地点の地点番号を，目的地から出発地までの順に配列sRouteに設定する。そのため，行番号42では，ループ変数iに目的地の地点番号であるdpの値を代入して初期値とし，iがspになるまで繰り返している。繰返しの中でiを空欄cに代入しているが，図1の場合，1回目の繰返しではiの値は6であり，これが配列sRouteの先頭sRoute[0]に代入される。また，i＝spとなって繰返しを抜けた後，行番号48でsRoute[j]にspを代入することで，出発地の地点番号0が設定される。したがって，空欄cには，「sRoute[j]」（キ）が入る。

- 空欄d：繰返しの中でiの値を更新する処理である。配列sRouteには，目的地から出発地までの順に地点番号を格納するので，次のiの値は，地点iの直前に通過する地点の地点番号である。直前に通過する地点番号は行番号35で配列pRouteに設定しており，地点iの直前の地点はpRoute[i]に格納されている。例えば，地点6の場合，i＝6なので，直前の地点はpRoute[6]に格納されている。したがって，空欄dには，「pRoute[i]」（イ）が入る。

［設問2］

図1において，出発地の地点番号sp＝0，目的地の地点番号dp＝6の場合について，行番号28のαにおける sPoint の値，及び行番号39のβにおける配列 pDist と配列 pRoute の内容をトレースする。配列 pRoute の初期値について，〔プログラムの説明〕にもプログラム中にも定義がなかったが，設問文の最後に「全ての要素には初期値として0が格納されている」と記述されている。読み落とすとトレースが進まないので，注意したい。繰返しの3回目までをトレースすると次のとおりとなる。網掛け部分が更新された要素である。

初期状態

	0	1	2	3	4	5	6
pFixed	false	false	false	false	false	false	false
pDist	0	∞	∞	∞	∞	∞	∞
pRoute	0	0	0	0	0	0	0

1回目

配列pFixedの全ての要素がfalseなので，i＝0のとき，行番号15の判定が真となり，break命令が実行される。i≠nPointなので行番号20は偽となり，i＝0の状態で行番号23の繰返しを開始する。繰返し変数jは，地点iと同一の地点を比較しても意味がないので，i＋1を初期値としている。配列pDistは，pDist[0]のときだけ0，それ以外の要素は∞なので行番号24が真になることはなく，i＝0のまま繰返しを終了する。そして，行番号28でsPointに0が代入され，行番号29でpFixed[0]にtrueが設定される。

行番号30～39では距離の比較を行うが，行番号31の条件にnot(pFixed[j])があるので，j＝0では何も処理しない。j＝1では，not(pFixed[1])が真となり，Distance[0][1]＝2＞0なのでnewDistを計算する。pDist[0]＋Distance[0][1]＝0＋2＝2から，newDist＝2＜pDist[1]＝∞となり，行番号33が真となるので，pDist[1]を2，pRoute[1]を0に更新する。j＝2，3でもj＝1と同様に処理が実行され，配列pDistと配列pRouteが更新される。j＝4以上は，地点4，5，6が地点0と隣接していないため，行番号31でDistance[0][j]＝－1＜0となり，処理を実行しない。

	0	1	2	3	4	5	6
pFixed	true	false	false	false	false	false	false
pDist	0	2	8	4	∞	∞	∞
pRoute	0	0	0	0	0	0	0