平成１６年度　春期　ソフトウェア開発技術者　午後Ⅱ問題

問１　2 点間の経路の探索に関する次の記述を読んで，設問１～５に答えよ。

図１のように，水平線と垂直線でセルに区切られた領域の中で，始点セルから終点セルまでの経路を求めることを考える。ここで，経路とは次の条件を満たすものをいう。

・あるセルからは，その上下左右に隣接するセルに移動できる。経路とは，このような移動を繰り返すことによって，あるセルから別のセルに行く際に通るセルの列である。
・障害物があるセルを通ることはできない。
・一つのセルを 2 回以上通ることはできない。

図１において網掛けのセルが，避けるべき障害物を表している。S が始点セル（以下，始点という），T が終点セル（以下，終点という）を表す。

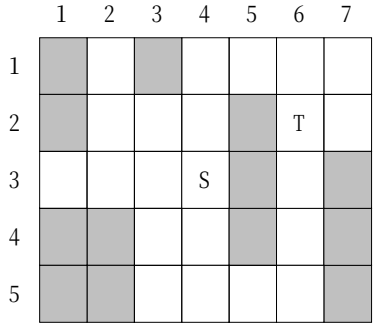
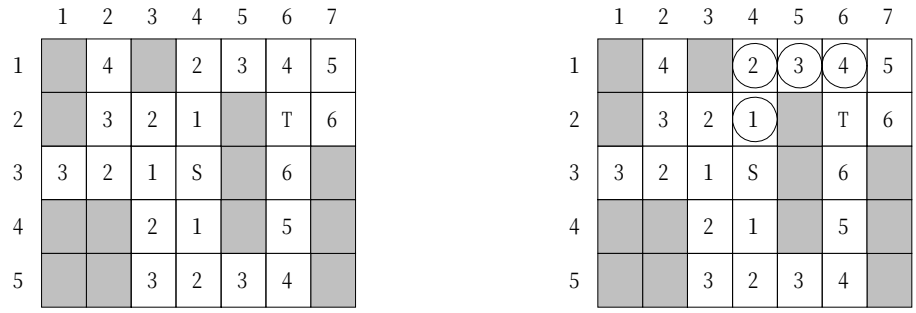


図１　経路の探索領域

図２（a）の例に示すように，始点 S からの距離（たどったセルの個数）を各セルのラベルとして付けていく（ラベル付け）。次に終点 T からラベルの逆順にセルをたどること（経路決定）によって，始点から終点までの経路を求めることができる。図２（b）では，発見した経路中の最短経路上にあるセルのラベルを○で囲っているが，アルゴリズムによっては，必ずしも最短経路を発見できるわけではない。



（a）ラベル付けの例　　（b）経路決定

図２　ラベル付けと経路決定

ここで，始点 S から終点 T に向かってラベル付けを行うときに，ラベルを付けたセルの集合を W とすると，この問題は次の手順で解くことができる。

(1) 集合 W を空集合に初期化する。
(2) 始点 S にラベル 0 を付け，W に始点 S を入れる。
(3) W が空でなく，終点 T のセルにラベルが付いていない間，次の操作を繰り返す。
　① W から一つの要素 p を取り出す。
　② セル p の上下左右にあるセルのうち，まだラベルが付いておらず，障害物でもない各セル q には，セル p のラベルに 1 を加えた新たなラベルを付け，W に入れる。
(4) 終点 T にラベルが付いていれば，始点と終点を結ぶ経路が存在する。その経路は次のようにして求める。
　① 初期値として，終点 T をセル p とする。
　② セル p が始点 S となるまで，次の操作を繰り返す。
　　i　セル p の上下左右のセルのうち，セル p のラベルから 1 を引いた値をラベルとしてもつセル q を求める経路上のセルとする。
　　ii　セル q をセル p とする。

探索領域が縦 n 個，横 m 個のセルに区切られているとする。縦が i 番目，横が j 番目のセルを（i，j）と表記する。このとき，i は上から下へ，j は左から右へ数えるものとする。探索領域は（1，1）から（n，m）までと考えられるが，ここでは探索領域の外側にダミーのセルを置くことにする。ダミーのセルも含めると，考えるセルの範囲は（0，0）から（n＋1，m＋1）までとなり，これらのセルを一次元配列で表す。

なお，配列のインデックスは 1 から始まる。

このとき，探索領域内のセル（i，j）は，配列の i×（m＋2）＋（j＋1）番目の要素に対応させる。すると，セル（i，j）の上下左右の隣接セルは，それぞれ配列の［　ア　］，［　イ　］，［　ウ　］，［　エ　］番目の要素になる。したがって，配列の k 番目の要素に対応するセルの隣接セルをそれぞれ，（上）k＋d1 番目，（下）k＋d2 番目，（左）k＋d3 番目，（右）k＋d4 番目と表す場合，d1～d4 は次のようになる。

d1＝［　オ　］，d2＝［　カ　］，d3＝［　キ　］，d4＝［　ク　］

この問題を解くアルゴリズムを考える。ここで考えるアルゴリズムでは，表１に示す定数，変数及び関数を使用する。

表１　定数，変数及び関数の説明

名称	内容
AVAIL	セルにラベルが付けられていないことを表す負の定数
OBST	セルに障害物があることを表す負の定数
n	探索領域の縦方向のセルの個数
m	探索領域の横方向のセルの個数
MAXSIZE	配列の要素数((n+2)×(m+2))
cell[MAXSIZE]	サイズが MAXSIZE の配列で，各セルの情報をもつ。
d[4]	サイズが 4 の配列で，隣接するセルとのインデックスの差分に関する情報をもつ。
S	始点に対応する cell の要素のインデックス
t	終点に対応する cell の要素のインデックス
insert(k)	cell のインデックス k を集合 W に入れる。
pick_cell()	集合 W から cell のインデックスを一つ取り出す。ただし，集合 W に cell のインデックスがない場合は 0 を返す。
print(str)	文字列 str を表示する。

初期化関数 init()では，次の探索領域の状態設定を行う。

（1）集合 W を空集合に初期化する。

（2）探索領域内の障害物のセルに対応する配列 cell の要素に OBST を代入する。

（3）外周上のダミーのセルに対応する配列 cell の要素に OBST を代入する。

（4）通過可能なセルに対応する配列 cell の要素には AVAIL を代入する。

（5）始点と終点を表す配列 cell の要素のインデックスを，それぞれ s，t に代入する。

（6）配列 d の要素 d[1]～d[4]に，d1～d4 の値をそれぞれ代入する。

次に，init()で初期化を終えた後に呼ばれる関数 find_path()と，find_path()の中で呼ばれて経路を決定する関数 back_trace()のアルゴリズムを図３に示す。

```
function find_path(s, t)
   cell[s] ← 0;
   insert(s);
   p ← pick_cell();
   while(p > 0 かつ cell[t]が AVAIL に等しい)
      for(i を 1 から 4 まで)
         if(cell[p + d[i]]が AVAIL に等しい)
            cell[p + d[i]] ← [  ケ  ] ;
            insert( [  コ  ] );
         endif
      endfor
      p ← pick_cell();
   endwhile
   if(cell[t]が AVAIL でない)
      back_trace(t, s);
   else
      print(" [  サ  ] ");
   endif
endfunction

function back_trace(x, y)
   x を発見した経路に加える;
   p ← x;
   while(p が y に等しくない)
      for(i を 1 から 4 まで)
         if(cell[p + d[i]])が [  シ  ] に等しい)
            p ← p + d[i];
            p を発見した経路に加える;
            break;
         endif
      endfor
   endwhile
endfunction
```

図３　ラベル付けと経路決定のアルゴリズム

設問１　図４で示す始点 S と終点 T を与えるときに, すべてのセルに始点から最短の距離でラベル付けを行い，見つかった経路の中での最短経路を示せ。そのとき，図２で示したように，最短経路のセルのラベルを○で囲め。

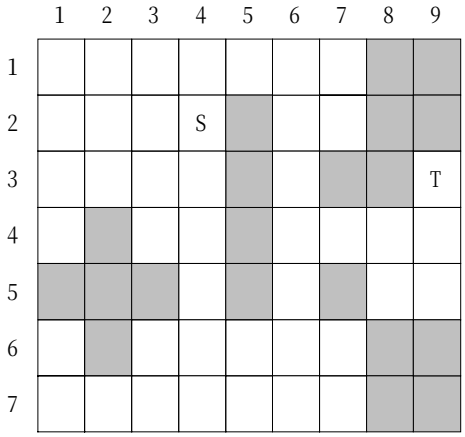


図４　セルと障害物の配置

設問２　本文中の［　ア　］～［　ク　］に入れる適切な式を答えよ。

設問３　図３で示したアルゴリズムに関する次の問いに答えよ。

（1）図３中の［　ケ　］，［　コ　］，［　シ　］に入れる適切な式を答えよ。

（2）図３中の“`print("`［　サ　］`")`”では［　サ　］に示すエラーメッセージを表示する。どのような内容のメッセージが表示されるか。15 字以内で答えよ。

設問４　集合Ｗの実現として,スタック,キューの２通りの方式で考え,それぞれの方式で insert(), pick_cell()の二つの関数を実現する。関数の実現法に関する次の問いに答えよ。ここで, stack[MAXSIZE], queue[MAXSIZE]は, それぞれ集合Ｗのデータを保持するための配列であり, 初期化関数 initW()は全体の初期化関数 init()から呼び出される。

(1) スタックを用いて実現する場合のアルゴリズムを図５に示す。［　ス　］, ［　セ　］に入れる適切な式を答えよ。

なお, 変数 top はスタック内の要素の個数を示している。

```
function initW()
   top ← 0;
endfunction

function insert(x)
   if(top ≧ MAXSIZE)
      print("スタックが満杯です");
   else
      top ← top + 1;
      [  ス  ] ;
   endif
endfunction

function pick_cell()
   if(top ≦ 0)
      return(0);
   else
      top ← [  セ  ] ;
      return(stack[top + 1]);
   endif
endfunction
```

図５　スタックによる実現法

（2）キューを用いて実現する場合のアルゴリズムを図６に示す。［　ソ　］～［　チ　］に入れる適切な式を答えよ。

なお，キューは循環配列で実現されていて，nq はキューの要素の数，rear はキューに要素を入れる場所，front はキューから要素を取り出す場所をそれぞれ示している。また，%は剰余演算子である。

```
function initW()
    nq ← 0;
    rear ← 0;
    front ← 0;
endfunction

function insert(x)
    if(nq が MAXSIZE に等しい)
        print("キューが満杯です");
    else
        nq ← nq + 1;
        rear ← [  ソ  ] ;
        queue[rear] ← x;
    endif
endfunction

function pick_cell()
    if(nq が 0 に等しい)
        return(0);
    else
        nq ← [  タ  ] ;
        front ← (front % MAXSIZE) + 1;
        return( [  チ  ] );
    endif
endfunction
```

図６　キューによる実現法

設問５　始点 S と終点Tの両方から同時に探索した方が，早く解が求まる可能性がある。その手順は，次のとおりである。ここで，S から T に向かってラベル付けを行うときに，ラベル付けしたセルの集合を Ws とする。逆に，T から S に向かってラベル付けを行うとき，ラベル付けしたセルの集合を Wt とする。

（1）集合 Ws，Wt を空集合に初期化する。
（2）S，T には，それぞれラベル 1，－n×m を付ける。
（3）Ws に S を入れ，Wt に T を入れる。
（4）Ws，Wt が空でない間，ラベル付けが合流するまで次の①，②を繰り返す。
　① Ws から一つの要素 a を取り出す。セル a の上下左右にあるセルのうち，まだラベルが付いておらず障害物でもないセル b に，セル a のラベルに 1 を加えた新たなラベルを付け，Ws に入れる。セル a の上下左右のセルのどれかに既に－n×m～－1 のラベルが付いていれば，ラベル付けが合流した。
　② Wt から一つの要素 c を取り出す。セル c の上下左右にあるセルのうち，まだラベルが付いておらず障害物でもないセル d に，セル c のラベルに 1 を加えた新たなラベルを付け，Wt に入れる。セル c の上下左右のセルのどれかに既に 1～n×m のラベルが付いていれば，ラベル付けが合流した。
（5）S からのラベル付けと，T からのラベル付けが合流したセルから，それぞれ S と T までの経路決定を行う処理を実行する。

アルゴリズムを図７に示す。［　ツ　］～［　ト　］に入れる適切な式を答えよ。ただし，［　サ　］は図３と同じである。また，図７を実行する際の AVAIL 及び OBST の値，並びに図７で初めて出てくる関数の意味は表２のとおりである。

表２　図７の定数及び関数の説明

名称	内容
AVAIL	セルにラベルが付けられていないことを表す負の定数。ただし，－n×m より小さい値
OBST	セルに障害物があることを表す負の定数。ただし，－n×m より小さく，AVAIL とは異なる値
Ws_insert(f)	cell のインデックス f を集合 Ws に入れる。
Wt_insert(g)	cell のインデックス q を集合 Wt に入れる。
Ws_Pick_cell()	集合 Ws から cell のインデックスを一つ取り出す。ただし，集合 Ws に cell のインデックスがない場合は 0 を返す。
Wt_Pick_cell()	集合 Wt から cell のインデックスを一つ取り出す。ただし，集合 Wt に cell のインデックスがない場合は 0 を返す。

```
function find_path(s, t)
   cell[s] ← 1; cell[t] ← −n×m; met_s ← 0; met_t ← 0;
   集合 Ws, Wt を空集合とする;
   Ws_insert(s); Wt_insert(t);
   while(真)
      p ← Ws_pick_cell();
      if(p が 0 でない)
         for(i を 1 から 4 まで)
            if(−n×m ≦ cell[p + d[i]] ≦ −1)
               met_s ← p;
               break;
            elseif(cell[p + d[i]]が AVAIL に等しい)
               [  ツ  ] ;
               Ws_insert(p + d[i]);
            endif
         endfor
      endif
      if(met_s が 0 でない)
         break;
      endif
      q ← Wt_pick_cell()
      if(q が 0 でない)
         for(j を 1 から 4 まで)
            if(1 ≦ cell[q + d[j]] ≦ n×m)
               met_t ← q;
               break;
            elseif(cell[q + d[j]]が AVAIL に等しい)
               cell[q + d[j]] ← [  テ  ] ;
               [  ト  ] ;
            endif
         endfor
      endif

      if(met_t が 0 でない)
         break;
      endif
      if(p が 0 に等しくかつ q が 0 に等しい)
         break;
      endif
```

```
    endwhile
    if(met_s > 0)
        back_trace(met_s + d[i], t);
        back_trace(met_s, s);
    elseif(met_t > 0)
        back_trace(met_t, t);
        back_trace(met_t + d[j], s);
    else
        print(" [  サ  ] ");
    endif
endfunction
```

図7　SとT両方からの探索