H21. 春. データ構造とアルゴリズム

問8

解答

問8	設問					
	a	b	c	d	e	f
	イ	ウ	イ	イ	ア	オ

解説

a　現在の画素（V, H）から，上下左右どの方向の画素をチェックしているかを検討します。この処理は，行番号30～33で行っています。

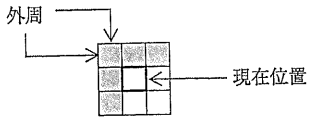
30　・CheckAndStack(V－1, H)
31　・CheckAndStack(V, H－1)
32　・CheckAndStack(V＋1, H)
33　・CheckAndStack(V, H＋1)

Vが縦（上下）の座標，Hが横（左右）の座標なので，各行番号の座標は，次のとおりです。

・行番号30：Vから1を引いているので上
・行番号31：Hから1を引いているので左
・行番号32：Vに1を加算しているので下
・行番号33：Hに1を加算しているので右

したがって，1の次の2は，上方向をチェックするので，「イ」が該当します。

b　塗る色の値は，1，2，3のいずれかであるため，外周に0の色を設定しておけば，必ず色が異なるので，範囲外とみなすことができます。

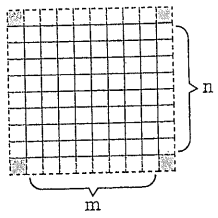


上記の例において，現在位置の上，左の色は，現在位置の色とは異なるので，必ず範囲外となるため，配列の上限や下限のチェックが不要です。

したがって，**“配列Imageの各添字の範囲チェックを省略できる”**が適切です。

c　行番号16～19では，Vを1から8まで変化させており，Wallの代入文が2つあるので，16画素にWallを代入しているとわかります。行番号20～23も同様，16画素にWallを代入しているので，計32画素にWallを代入しています。ここで，左上，左下，右上，右下の4つの画素には何も代入していないことに気をつけましょう。

次の図の網掛けの部分が，Wallの代入されない部分です。



したがって，表示領域がm×nの画素であると，Wallを代入する画素の数は，“2(m＋n)”となります。

d　8ビットの2進数は，符号なしとみれば0～255なので，負数は使えません。負数を考慮すると，－128～127の範囲の数値しか表せないからです。一方，dに関する解答群をみると，CCかNCを使えばよいことは見当がつきます。ここで，NCは塗り替えたい色で，行番号39で，画素に代入して色を塗り替えていることが確認できます。

行番号11で開始点の色をCCに代入し，行番号12で“CC＝NC”であればプログラムを終了しているので，以降の処理では，“CC≠NC”が成立します。そこで，WallをNCで初期設定しても，行番号38の判断では，“Image[Vt, Ht]＝CC”，すなわち，“Wall＝CC”は成立しないので，外周については色を塗り替えることはありません。

したがって，**“Wall ← NC”**が入ります。

e　行番号25でCheckAndStackを呼び出しています。また，CheckAndStackでは，行番号38で，引数Vt，Htで与えられた座標の画素を塗り替えるかどうかの判断をし，同じ色の領域であれば，行番号39で色を塗り替えています。さらに，行番号40～42の処理は，色を塗り替えたので，塗り替えた座標の上下左右を調べるため，添字Moreを1増やして，この座標のVtをVPos[More]，HtをHPos[More]にそれぞれ格納しています。

これらの処理が終わると，行番号26に戻り，Moreが0ではないので，以降の行番号27～33を実行します。この一連の処理では，CheckAndStackで色を変えた座標の上下左右の座標を塗り替えるかどうかの判断と塗り替えを行っています。

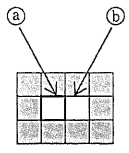
一方，CheckAndStackで塗り替えなかった場合は，Moreが0なので，行番号26で“More＝0”成立し，行番号27～33は実行しません。

ここで，行番号12～14を省略した場合を検討します。行番号12～14を省略すると，“CC＝NC”となる状態が発生するということです。

まず，行番号25で，CheckAndStackを呼び出し，行番号38～44で色を塗り替え，Moreは1となり，行番号26～33を実行します。しかし，上下左右ともに色が異なるので，行番号30～33で呼び出すCheckAndStackでは，行番号38の条件は成立せず，Moreは0のままです。なお，行番号29で，Moreは0となっています。

以上から，1か所の色は塗り替えられるため，**“仕様どおりに同じ色で塗り替えて正しく終了する。”**が適切です。

f　次の図のⓐを指定したとすると，まず，行番号25で，CheckAndStackを呼び出します。これは，CCと等しいので，色を塗り替え，ⓐの座標を配列HPosとVPosに格納するとともにMoreに1を加算するのでMoreは1になります。



次に，More＞0なので，行番号27～33を実行しますが，CCと一致するのはⓑ（ⓐの右）だけなので，行番号33のCheckAndStackから戻ったとき，Moreが1となっています。すなわち，行番号30～33の一連のCheckAndStackの呼び出しで，Moreは1になります。このとき，配列VPosとHPosに格納されている座標はⓑです。

再び，行番号26から実行しますが，このときは，ⓑの座標の上下左右を調べることになります。すると，行番号31のCheckAndStack呼出し（ⓑの左）で，行番号38の“Image[Vt, Ht]＝CC（＝NC）”が成立し，ⓐの座標を配列HPosとVPosに格納して行番号26から実行します。ⓐの座標が格納されると，今度は，ⓐの右が“Image[Vt, Ht]＝CC（＝NC）”を満たすため，ⓑの座標が格納されて行番号26から再び実行します。

以上から，Moreは0と1を繰り返し，CheckAndStackから戻ったときは常に1となり，行番号26～34を無限に繰り返すことになります。

したがって，**“変数Moreの値は一定値以下であるが処理が無限にループする。”**が適切です。