塗り壁プログラム

目次

1　はじめに

私は、趣味で離散最適化に関連した問題の考察、及びプログラムによる問題の解の列挙や数え上げを行っている。

プログラミングの土台として、独自に数学（特にグラフ理論や順列の理論)的な機能を持つクラス群（以下，クラスライブラリーと言う)を C++で開発している。

本発表では、私が取り組んでいる問題やクラスライブラリーについて前半で述べ、後半は応用例としての塗り壁プログラムについて述べる。

2　グラフプログラミング

2.1 グラフの利点

私がグラフ理論（を中心とする離散最適化)を研究対象に選んだのは,題材が身近であり,問題や論理展開が直観的でわかりやすいからである。例えば,次の問題は私の Web サイトで出題しているものである。

なお、本発表では、グラフに関する基本的な知識は仮定し、必要に応じて補足する。(巻末に用語集を付記)

川渡りの問題：ある大きな川のほとりに A,B,C,D の４人が佇んでいる。この川を渡りたいが,船は二人乗りのものが１艘しかない。しかも、船を動かせるのはリーダーの A 君だけである。B 君と C 君、C 君と D 君は仲が悪く A 君がいないとすぐ喧嘩をはじめてしまう。喧嘩がおきないようにするには、A 君は、他の人をどのような順番で船に乗せたらよいであろうか。

この問題を解くには,適切なグラフを設定しそれを幅優先探索で辿ればよい。グラフを使うことにより、より一般化された問題が解決できるのも利点である。

2.2　クラスライブラリー

クラスはグラフの構造に関連するクラス及び､グラフに直接関係しない一般的な数学の機能を提供するクラスに大きく分けることができる。夫々についての抜粋を以下に示す。(網掛けは、塗り壁プログラムの中心的な機能を提供している部分)

・グラフ構造関連クラス

No	クラス名	機能、特徴、用途等
1	基本グラフクラス	グラフの基本機能（頂点や辺の追加、削除、次数列の算出等）や特殊なグラフの生成機能を提供する。頂点数は最大 400、辺数は 79800 まで取り扱いが可能。
2	グラフ表示クラス	GDI を使ってグラフ（No１のオブジェクト)を表示する。頂点は円周上に等間隔に配置し、辺は直線で表示。表示例は次ページ参照。
3	グラフファイルクラス	グラフファイルの入出力を行う。入力はテキスト形式、バイナリー形式の二つ、出力はバイナリー形式のみである。
4	部分グラフクラス	他のグラフの部分グラフを表現するクラスであるが、単独のグラフとしても使用可能であり､その場合は No１の強化版と成る。頂点数は 20000,辺数は 2 億弱まで取り扱い可能であり、連結成分、近傍、最大クリークの算出機能を持つ。、
5	重み付部分グラフクラス	No４から派生し、頂点または辺に重みを持つグラフを表現する。特殊成分や特殊近傍の算出機能を持つ。
6	ボードグラフクラス	ボードパズルを表現するクラスで、No１のクラスから派生している。
7	ボードグラフ表示クラス	ボードの表示、及びマスの彩色、アイコンやビットマップのマスへの描画を行う。

表 1 クラスの概要

グラフの表示例を下図に示す。これは、グラフエディターを使って表示したものである。

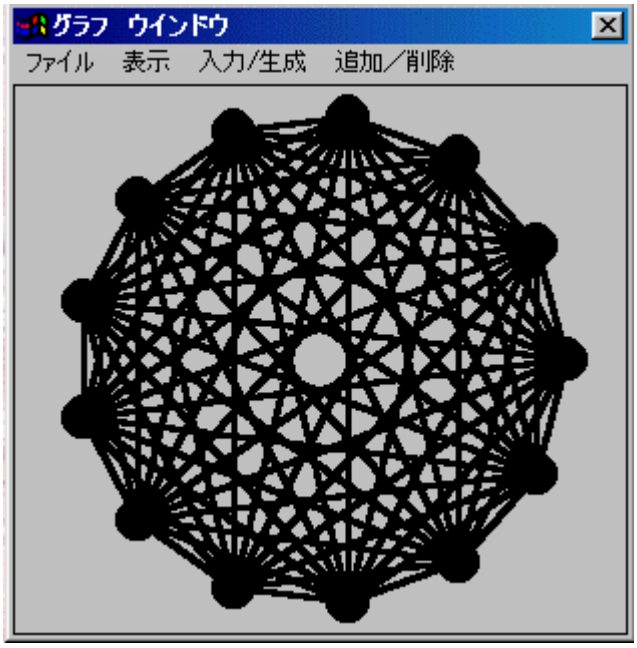


図 1 グラフの表示例（K13）

・一般数学クラス（及び関数）

グラフ構造関連クラスやアプリケーションに対して数学的な機能やデータ構造を提供する。

提供しているデータ構造は、可変長配列、ビットベクトル、スタック、キュー、集合、群、準群等がある。

◆順列クラス　順列と順列番号とのマッピングを与えるクラス。順列と文字列との相互変換、順列の型の算出、クイーン問題の解かどうかの判定、２面体群の作用、誘導順列の算出（グラフ同型分類問題で使用)の機能も持つ。

その他の機能として、すれ違い数の計算、組合せと組合せ番号との相互変換等がある。前者はラテン方陣標準形の列挙、後者はカークマン問題の解の列挙に関連する。

2.3 ボードグラフ

ボードグラフとはボードゲームやボードパズルの問題構造を表現するために導入したクラスであり、ボードの桝目を頂点とするグラフである。ボードグラフの辺の設定は対象とするパズルやゲームにより異なる。それを次表に示す。

なお、対象として考えているゲームは囲碁、オセロ、将棋などがある。

No	名称	辺を持つ条件
1	塗り壁グラフ	二つのマスが縦または横に隣接している。
2	マインスイーパーグラフ	二つのマスが縦、横、または斜めに隣接している。
3	ラテン方陣グラフ	二つのマスが縦、または横の同じ列上に無い。
4	クイーングラフ	二つのマスが縦、横、または斜めの同じ列上に無い。
5	ナイト（騎士）グラフ	一方のマスから他方へチェスのナイトで移動可能である。

表 2 ボードグラフの種類

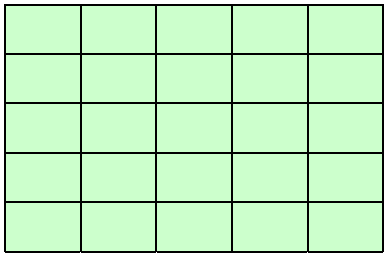
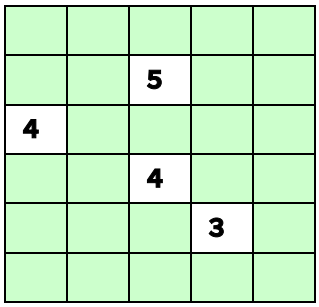


図 2 ボードのイメージ

3 塗り壁プログラム

3.1 塗り壁パズルの紹介

開始図



終了図

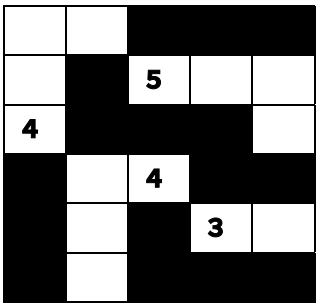


図 3 開始図と終了図（部分図）

塗り壁とは、ボードのいくつかのマスに数字が与えられている状態（開始図参照)から出発し、下記の条件を満たすように全てのマスを白または黒で塗分けるというパズルである。

なお、縦または横に連なった白マスの極大な塊（白マス部分グラフの連結成分)をシマと言う。

・塗り壁の条件

① 数字の入ったマス（以下、数字マスと言う)は白マスになる。

② 各シマには、一つ且つ唯一つの数字マスが属する。

③ シマの大きさは、それが含む数字マスの数字（以下、そのシマの容量という)に等しい。

④ 全ての黒マスは縦または横につながっている。(黒マス部分グラフは連結である)

⑤ 黒マスからなる２マス×２マスの正方形（４サイクル)は存在しない。

3.2 解答アルゴリズム群

ここで言うアルゴリズムとは、現在の状況を踏まえ、次に白マスになるべきマス或いは黒マスになるべきマスを決定するアルゴリズムのことを言う。

（1）準備

塗り壁の問題は重み付塗り壁グラフで表現する。即ち、重みでマスの状態を管理する。状態は次の４種類がある。

・未確定状態（初期値）

・白マス半確定状態（白マスであることは確定したが、どのシマに属するか未定の状態)

・白マス全確定状態（属するシマが確定している白マス）

・黒マス確定状態

また、全確定マスのみからなるシマ、半確定マスのみからなるシマを夫々全確定シマ、半確定シマと称する。

（2）初級アルゴリズム

アルゴリズムは初級、中級の２つに分けたが、この区分けはあまり厳密なものではない。また、残念ながら、上級と言えるようなアルゴリズムはまだ得られていない。

初級アルゴリズムとは概ね次の条件を満たすものであり、それ以外を中級とした。

・使用頻度が高い

・原理が簡単で実装が容易

・処理が高速（確定検索)

◆黒マス確定法

A) シマ確定法―容量が現在の大きさに一致するシマを探し、その近傍を黒マスで確定させる。

B) 未確定近傍法（黒）―各黒マス成分の未確定近傍を求め、それが唯一点からなるとき、その未確定点を黒マスで確定させる。

C) 重複近傍法―二つの全確定シマの近傍の共通点を、黒マスに確定させる。
また、全確定シマと半確定シマが重複近傍を持つ場合、その現在の大きさの合計＋１が容量を越えるとき、黒マスに確定させる。

下図に於ける記号の意味は次の通り

◎：全確定白マス　〇：半確定白マス　×：黒マス

▲：黒マスに確定するマス　　空白：未確定マス

5	×	×	◎	×	◎				
◎	▲	◎	6	×	5				
				▲	◎				
	▲				◎			◎	
▲	◎	▲	▲					4	▲
▲	◎	4	◎	×				▲	〇
	▲	×	▲						〇
					▲	5	◎		
				3	◎	▲			
					▲	2			

図 4 黒マス確定アルゴリズム 参考図

◆白マス確定法

D) ４サイクル法―4サイクルで、３マスが黒マス、残り１マスが未確定マスとなっているものを探し、未確定マスを白マスで確定させる

E) 未確定近傍法（白）―現在の大きさが容量に達していない全確定シマ、または全ての半確定シマの未確定近傍を求め、それが唯一点からなるとき、その未確定点を白マスで確定させる。

F) 容量一致法―全確定シマの白未確定成分の大きさがそのシマの容量と等しいとき、この成分に含まれるマスを白マス全確定させる。

△は、白マスに確定するマスを表わす。

5	◎	×	◎	8	◎	◎	×		
×	◎	×	◎	◎	◎	△			
	△		×	×	×				
						×	×	×	
					×	△	5	◎	×
	△	×	△			×	×	△	△
	×	×	×					×	×

図 5　白マス確定アルゴリズム　参考図

◆両マス確定法

G)　孤立マス法―未確定マスの周りが全て黒マスの場合は黒マスに、白マスの場合は白マスに確定させる。白マスの場合、未確定マスを囲む白マスは全て同一のシマに属さねばならない事に注意。

（3）中級アルゴリズム

中級アルゴリズムを１、２の２種類に分ける。中級２は処理が重く使用上注意すべきものを入れた。

・中級１　アルゴリズム

A)　残り容量１法―次の条件を満たす全確定シマを探す。

・現在の大きさ＝シマ容量―1

・未確定次数が２のものが１個あり、残りは全て未確定次数が０

・未確定次数が２の白マスの二つの未確定隣接点が斜めに連なっている。

このとき、その二つの未確定隣接点の共通未確定隣接点を黒マスに確定する。

B)　桂馬跳び法―次の条件を満たす全確定シマを探す。

・未確定次数が２のものが１個あり、残りは全て未確定次数が０

・未確定次数が２の白マスの二つの未確定隣接点が斜めに連なっている。

・未確定次数が２の白マスの桂馬跳びの位置に他のシマの全確定白マスがある。

このとき、未確定次数が２の白マスの斜めマスを黒マスで確定する。

C)　拡張容量一致法―他の全確定シマの近傍が全て黒マスと仮定して、容量一致法を行う。

D)　孤立未確定成分法―全確定シマの近傍が全て白マスと仮定した上で、次の条件を満たす全確定シマを探す。

・未確定隣接点の未確定成分の大きさは一つを除いて１。

・現在の大きさ＋未確定成分１のものの個数がシマの容量より小さい。

この条件を満たすシマに対し、未確定成分の大きさが２以上の未確定隣接点を白マスに確定する。

－は黒マスと仮定するマスを表わす

4	◎	◎					×	×	
×	×		▲					△	6
7	×							－	
◎	×							5	－
◎									
	▲	◎	4						
							－	3	－
					2	－	△	－	×
					－	△	△	△	5

図 6　中級１アルゴリズム参考図　その１

E)　未到達マス法―全確定シマの近傍が全て黒マスと仮定した上で、どの全確定シマからも到達できないマスを黒マスとして確定する。但し、容量 N の全確定マスの現在の大きさを M とするとき、その到達できる範囲とは、そのシマの未確定閉近傍を取る操作を N-M 回続けて得られる範囲を言う。

①、②・・は１回目、２回目・・・の閉近傍算出で付け加えられたマスを示す。

		②	①	②		×	②	×	
	×	①	◎	①	×	×	①	×	③
②	①	5	◎	①	②	×	◎	①	②
	②	①	①	×		×	6	×	③
		②	②		×	×	◎	×	
					×	②	①	②	③
						③	②	③	
							③		

図 7　中級１アルゴリズム参考図　その２

・中級２　アルゴリズム

F)　拡張未確定次数法１（黒）―未確定次数１の黒マスに対し、その未確定隣接点が白マスと仮定したとき黒未確定成分が分断されるとき、その点を黒マスに確定する。分断されるかどうかは、元の黒マスを含む黒未確定成分の大きさが黒数（＊）以上になるかどうかで判定する。
（＊）マス数－シマ容量の合計

G)　拡張未確定次数法２（黒）―未確定次数２の黒マスを対象とする以外は前項と同様。

H)　拡張未確定次数法３（黒）―未確定次数３の黒マスを対象とする以外は前々項と同様。

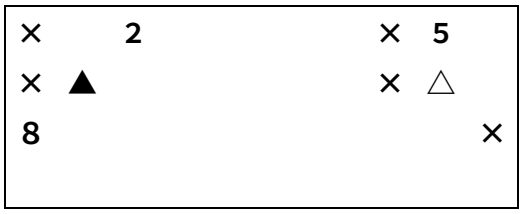


図 8　中級２アルゴリズム　参考図

I)　拡張未確定次数法１（白）―他の全確定シマの近傍が全て黒マスと仮定した上で、全確定シマの未確定隣接点が黒マスと仮定したときの白未確定成分の大きさが、そのシマの容量より小さいときその未確定隣接点を白マスで確定する。

J)　未確定成分法―孤立未確定成分法において、成分の大きさが１と言う制限をはずしたもの。

K)　切断点検索法―他の全確定シマの近傍が全て黒マスと仮定した上で、全確定シマの切断点を求め、その切断点及び前後のマスを白マスで確定する。（未実装）

3.3　実装

(1)基本的な方針

・Windows　ダイアログアプリケーションとして作成する。

・プログラムは塗り壁に関するさまざまな機能（問題の解答、評価、作成等）を持たせる。（但し、本発表時点では、解答関連の機能しか実装できていない）

・解答関連機能として次のものを入れる

問題入力及び表示､ユーザーが選択した解答アルゴリズムの実行とその結果の画面への反映

表示例は下記の通り。（ボードの部分のみ抜粋）

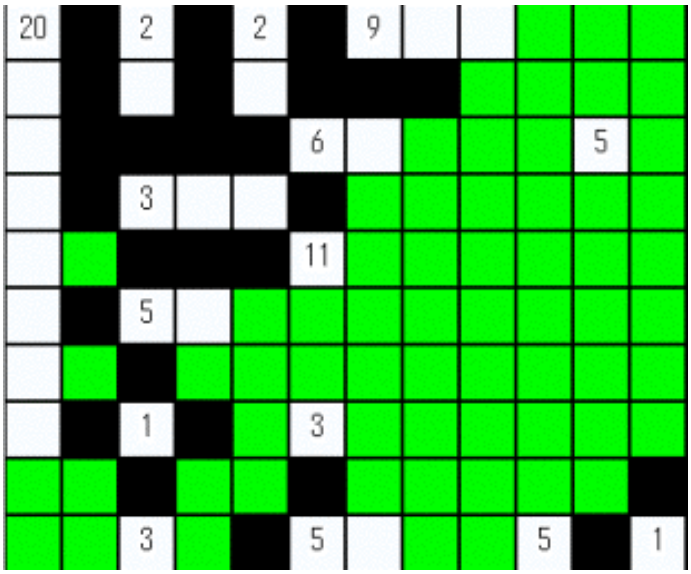


図 9　画面表示例（部分図）

(2)クラスの実装

中心となるのが、部分グラフクラス及び重み付部分グラフクラスの連結成分や近傍の取扱の機能である。

・部分グラフクラス

関連するメンバー関数は次の通り

①int GetDegree(int vn,int sz=0,int* list=0);
②int GetNeighborhood(int sz1,int* vn,int sz2,int*out);
③int GetComponent(int v,int sz=0,int* vlist=0,int* dist=0);

①は指定された頂点の隣接点のリストを返す。②は指定された頂点列の近傍を、①を用いて次のように求めている。

```
for (cnt=0;cnt<sz1;cnt++)
    bit.SetBit(vn[cnt]); //頂点列をビットベクトル
                                        bitに退避
for (cnt=0;cnt<sz1;cnt++)
{  //各頂点の近傍を求め、bitに含まれない点をビット
                                    ベクトルnhdに設定
    if  (ret =  GetDegree(vn[cnt],size,list),ret<0)
        return -1;
    for (cnt1=0;cnt1<ret;cnt1++)
        if  (!bit.IsSet(list[cnt1]))
            nhd.SetBit(list[cnt1]);
}
return  nhd.GetList(sz2,out);
// nhdの内容をoutに吐き出す。GetListの戻り値は全データの
                                        個数
```

③はキューを用いて幅優先探索を行い、連結成分を求めている。

```
q.Enqueue(v);
while   (bit.GetNum()<vnum)
{
    level++;
    ret=q.GetList(size1,list);
```

```
        q.Clear();
        for (cnt=0;cnt<ret;cnt++)
        {
            ret1=GetDegree(list[cnt],size2,list1);
            for (cnt1=0;cnt1<ret1;cnt1++)
            {
                if  (!bit.IsSet(list1[cnt1]))
                {//キューにまだ入っていない点をｴﾝｷｭｰ
                    q.Enqueue(list1[cnt1]);
                    bit.SetBit(list1[cnt1]);
                }
            }
        }
        if  (!q.GetSize())
            break;
    }
```

・重み付部分グラフクラス

関連するメンバー関数は次の通り

```
①int GetVWeight(int vn);
②int SetVWeight(int vn,ushort vw);
③int GetSpecialNeighborhood(int vn, int sz1,int* vl1,
      int sz2, int * vl2);
④int GetSpecialComponent(int vn,int sz1,int* val,
      int sz2, int * vl);
```

③④のコードは部分グラフクラスの該当する関数に重みの処理を入れただけである。

(3)アプリケーションの実装

①未確定近傍方（黒）

```
    for (cnt=1;cnt<=size;cnt++)
    {
        if  (bit.IsSet(cnt))
            continue;
        if  (model.GetVWeight(cnt)!=3)
            continue;   //  黒マス以外は対象外
        ret = model.GetSpecialComponent(cnt,3,
              size,list);
        for (cnt1=0;cnt1<ret;cnt1++)
            bit.SetBit(list[cnt1]);
        if  model.GetSpecialNeighborhood(ret,list,
              1,wtb,size,list1)==1)
            if  (size < yoko*tate-sirosuu)
                KuroKakutei(list1[0]);  //  黒マス確定
    }
```

②拡張容量一致法

```
for (cnt=0;cnt<simasuu;cnt++)
{   //  各シマの近傍を黒マスに仮設定
     ret = model.GetSpecialComponent(sima[cnt].masuno,
           2,siro,size,list);
     ret1 = model.GetSpecialNeighborhood(ret,list,1,mi,
           size,list1);
    for (cnt1=0;cnt1<ret1;cnt1++)
         w1.SetVWeight(list1[cnt1],3);
}
for (cnt=0;cnt<simasuu;cnt++)
{   //  シマ毎に近傍を未確定に戻す
    w2 = w1;
//白成分を求める
    ret = model.GetSpecialComponent(sima[cnt].masuno,2,
          siro,size,list);
    ret1 = model.GetSpecialNeighborhood(ret,list,1,mi,
          size,list1);
    for (cnt1=0;cnt1<ret1;cnt1++)
          w2.SetVWeight(list1[cnt1],0);
//白未確定成分を求める
    ret = w2.GetSpecialComponent(sima[cnt].masuno,3,
          siromi,size,list);
    if  (ret==sima[cnt].weight)
        for (cnt1=0;cnt1<ret;cnt1++)
            ZenKakutei(list[cnt1],cnt+1);
    }
```

(4)正答率

問題サイズ別の正答率は以下の通り

サイズ	問題数	正答数	正答率	カバー率
(10、10)	９６	７４	７７．１%	７９．６%
(10、18)	４８	１９	３９．６%	７７．９%
(14、24)	１３	１	７．７%	６８．５%
(20、36)	１	０	０%	４３%

4　　おわりに

(1)塗り壁プログラムに関して

解答機能だけの実装を行ったが、アルゴリズムが不十分であるので、今後も追加していく。また、性能強化にも努めたい。

(2)より一般的なボードグラフに関して

本発表で、ゲームやパズルのプログラミングへのグラフ理論の適用の有効性が示されたと考える。今後、さまざまなゲームやパズルに関して適用範囲を広げていきたい。

参考文献

無し

参考サイト

点と線の部屋

http://plaza16.mbn.or.jp/~graph_puzzle/

点と線の部屋　別館

http://star.endless.ne.jp/users/graph/

用語

・グラフ――ある集合 V 及び、V の冪集合の部分集合 E の対(V,E)のことを言う。V の要素を頂点（または、単に点)、E の要素を辺と言う。

注：本資料では、無向、有限且つ単純なグラフのことを単にグラフと言う。

・近傍及び次数

グラフにおいて、頂点ｖの近傍とは、ｖに接続する頂点全体の集合を言い、それにｖを付け加えたものを閉近傍と言う。このとき、近傍に含まれる頂点数のことをｖの次数と言う。

また、頂点集合 A に対し、A の何れかの点と接続し、且つ A に含まれない点全体の集合を A の近傍と言い、それに A を付け加えたものを A の閉近傍と言う。

・連結成分

グラフにおいて、頂点ｖの連結成分とは、ｖ及び、これと直接または間接に接続する頂点全体の集合を言う。連結成分のことを単に成分とも略称する。

・重み付グラフ ――グラフであり、且つ頂点、辺のいずれか或いは両方に値が割当てられているグラフを重み付グラフと言う。

・特殊近傍

G を重み付グラフ、X を重みから成るある集合とする。このとき、頂点ｖの近傍に含まれ、且つその重みが X に含まれる点の全体を X-近傍と言う。頂点集合 A についても同様。なお、ｖまたは A の各頂点の重みは X に含まれるかどうかは問わない。

・特殊成分

G を重み付グラフ、X を重みから成るある集合とする。頂点ｖの重みが X に含まれるとき、ｖを含み重みが X に含まれる頂点からなる極大連結部分グラフを X-成分と言う。また、A が、重みが X に含まれる頂点からなる連結部分グラフのとき、A を含み重みが X に含まれる頂点からなる極大連結部分グラフを X-成分と言う。