畳敷詰問題

2013.06.27

問題１：部分集合の生成　ta111.c

省略

問題２：畳敷詰プログラム　ta211.c

省略

問題３：畳敷詰プログラム（列優先）　ta311.c

省略

問題３a：畳敷詰プログラム（列優先）改良版　ta311a.c

省略

番外　：状態遷移行列による解法

プログラムリストは、別紙４－１を参照。
実行結果は、別紙４－２を参照。

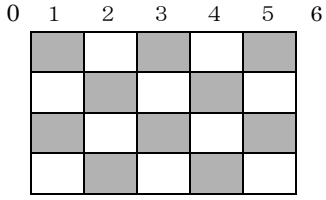
１．探索結果

```
----- PGen  M=20    38613965  38613965     6264
                                                                                    msec
===== M=20,N= 1   Ans =                                                        1     837
===== M=20,N= 2   Ans =                                                   10 946     817
===== M=20,N= 3   Ans =                                                  413 403     817
===== M=20,N= 4   Ans =                                              616 891 945     819
===== M=20,N= 5   Ans =                                           57 737 128 904     833
===== M=20,N= 6   Ans =                                       45 005 025 662 792     854
===== M=20,N= 7   Ans =                                    6 290 652 543 875 133     865
===== M=20,N= 8   Ans =                                3 547 073 578 562 247 994     879
===== M=20,N= 9   Ans =                              623 139 489 426 439 754 945     875
===== M=20,N=10   Ans =                          289 415 868 852 204 573 601 981     916
===== M=20,N=11   Ans =                       58 902 468 764 522 025 160 456 848     910
===== M=20,N=12   Ans =                   24 080 710 136 742 581 040 732 561 613     920
===== M=20,N=13   Ans =                5 422 065 916 132 126 528 319 352 874 496     934
===== M=20,N=14   Ans =            2 029 256 253 667 094 948 385 950 789 394 770     949
===== M=20,N=15   Ans =          490 984 130 367 164 806 905 167 493 235 118 259     954
===== M=20,N=16   Ans =      172 539 486 039 611 523 956 384 662 268 124 418 393     954
===== M=20,N=17   Ans =   43 983 420 324 451 571 797 442 416 459 108 745 607 592     958
===== M=20,N=18   Ans = 14 766 712 169 803 333 833 186 604 776 310 955 189 771 941     973
===== M=20,N=19   Ans = 3 911 222 696 776 012 972 239 561 518 359 338 259 546 415 835     997
===== M=20,N=20   Ans = 1 269 984 011 256 235 834 242 602 753 102 293 934 298 576 249 856     977
```

２．データ表現（遷移状態の管理）

2.1 畳を配置する位置と方向

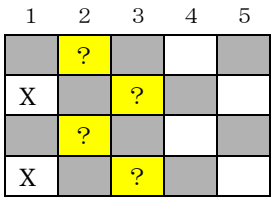
M x Nの部屋を以下の様に市松模様に塗り分ける。また、畳を配置する基準の位置を黒いマス目に限定する。



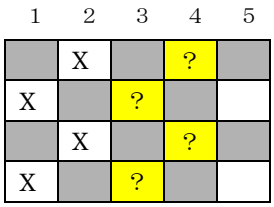
黒いマス目に配置する畳の方向は、左・右・上・下の４通りとなる。(テキストと同じ)

2.2 畳を配置した状態の管理

ある列ｎまでのすべての黒いマス目に畳を配置した状態を管理するビットマップを考える。



n=2の場合



n=3の場合

注：X：カバー済　　？：カバー済または未カバーの状態

ｎ 列まで畳を配置しているので、(n－1) 列までの白いマス目はカバーされている筈である。

ｎ列、(n+1)列の白いマス目はカバーされているものと(１) とカバーされていないもの(０) とが存在する。この１/ ０状態をビットマップとして管理する。各行には１つの0 / 1 が存在するので、それぞれの行に対応させてMビットの情報（状態マップ）として扱う。

なお、ｎの奇偶性により各ビットがどの列に対応するかが異なるので注意が必要である。

2.3 状態遷移行列

ｎ列の状態マップI から (n+1)列の状態マップJへの遷移行列 Tr (I, J) を以下の様に定義する。

Tr (I, J) ：ｎ列の状態マップIの状態から(n+1)列に畳を敷詰めて、状態マップJとなる配置の数。

特に、Tr (I, J) ＝ ０ の場合はそのようなパスが存在しないことを意味する。

ただし、Tr (I, J) は、ｎの奇遇により２つの行列のいづれかになる。

（プログラム上は、TrX, TrYとして分けている。）

３．状態遷移行列による パスの計算

3.1 ｎ 列から(n+1)列への状態遷移の計算

P (I) ：ｎ列が状態マップIとなるような畳の配置の数。

P　　：P(I)を要素とするベクトル(ｎ 列全体の状態マップに対する畳の配置の数)

Q (J) :（ｎ+1）列が状態マップJとなるような畳の配置の数。

Q　　：Q(J)を要素とするベクトル((n+1) 列全体の状態マップに対する畳の配置の数)

とすると、Q (J) ＝ P(I) ＊ Tr (I,J) となる。

また、ｎ列から (n+1) 列への全体の状態遷移を考えると

$$Q = Tr \cdot P \qquad i.e \qquad Q(j) = \sum_{i=0}^{i=2^M} Tr(i, J) * P(i)$$

となる。

3.2 初期値(第０列の状態マップ)

MASK1 ：奇数位置は１、偶数位置は０となるような状態マップとする。

ex. M=4なら、0101

MASK1の畳の配置の数を１とし、その他の状態マップに対応する畳の配置の数を０とする。

3.3 求める畳の配置数

N が奇数の場合：N 列計算後のMASK１の状態マップに対応する畳の配置数が求めるものである。

N が偶数の場合：N 列計算後のMASK０の状態マップに対応する畳の配置数が求めるものである。

ただし、MASK0：奇数位置は0、偶数位置は1となるような状態マップとする。

ex. M=4なら、1010

3.4 必要メモリ

本計算で必要とするメモリの大半は Tr 行列であり、2^M x 2^M バイトの奇数用と偶数用の2つである。

従って、M = 14とすると、2^{29} バイトとなり、これがメモリ制限からくる限界となる。つまり、上記の行列演算では、14 x N が限界となる。（N は、14 以上でもよいが）

しかし、Tr 行列の中身を考えると殆どは０である。そこで状態 I から状態 J への遷移ベクトル（TrList ）で Tr 行列の代用をさせる方式に変更した。本対策により、M=20 までの計算が可能となり、計算速度も向上した。（**20 x 20 の計算が約４１秒** ）

3.5 状態遷移行列の作成

これは、すべての入力パターン(状態マップのパターン）について通常の探索を１段進めることにより作成できる。ただし、奇数と偶数の2つが必要である。

探索の深さが１段（２ 回 ）なので、時間はほとんど問題にならない。

3.6 多倍長整数演算

Nが増えると畳の配置数は急激に増加し、６４ビット演算ではオーバーフローが発生するため、多倍長整数演算が必要となった。演算の種類が限定されているので自前で作成した。

以上