技術社会システム

第12回：動的計画法

担当教員：蓮池 隆(はすいけ たかし)

連絡先：thasuike@waseda.jp

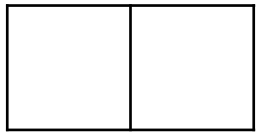
さっそく前回の演習の解答です

演習11-1

- 1から100のうち，99個の異なる数字がランダムに提示されるとする.
- 提示されなかった残された1つの数字Nを当てるためには，提示された99個を覚えておく必要がある

- 十進数何桁分のメモリがあれば十分か？100桁よりも少ない数となるアルゴリズムを求めなさい.
(少なければ少ないほどベターです)

解答



演習11-1

- 最初に0を2個並べ，提示された数字を足し上げていき，常に下二桁のみを保存する.
- 最後に保存された二桁の数をAとするとき，求める数Nは以下で計算される.

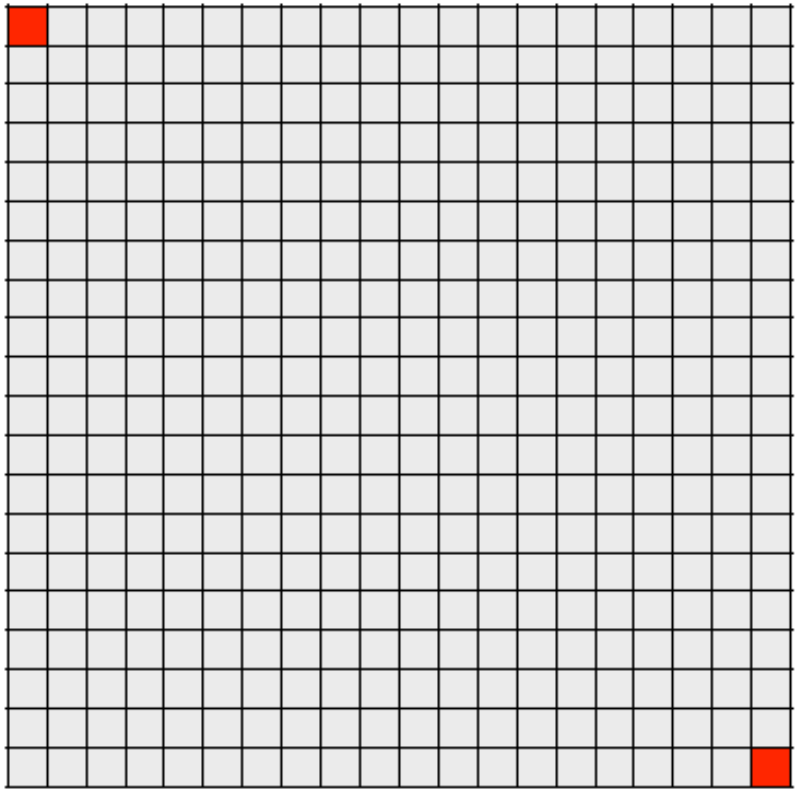
$$N=\begin{cases}50-A & \text{(ただし，}A<50\text{)}\\ 150-A & \text{(ただし，}A\geqq 50\text{)}\end{cases}$$

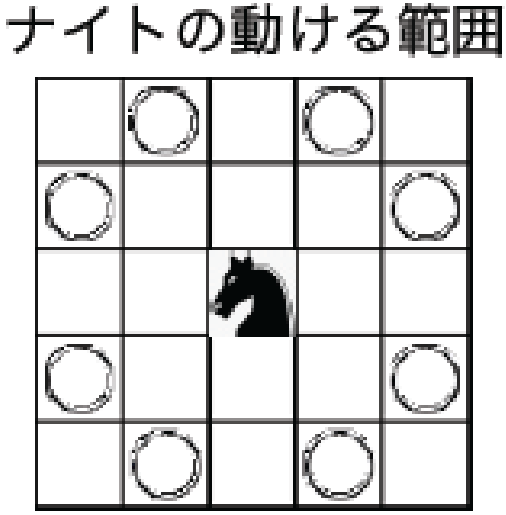
- 1～100まですべて足すと，5050．よって，**どれかの数が足りないとき，4950～5049**となる.
- Nが1～49までなら，50○○となり，5050から引き算で，百の位からの繰り下がりの必要なし.
- Nが50～100までなら，49○○となり，5050からの引き算で，百の位から繰り下がりの必要あり.

続いて解答です

演習11-3

- 100×100のチェス盤の左上のマスから右下のマスまで
  チェスのナイトを動かすのに必要な最小手数はいくつ？
  (下図はあくまでイメージ図)





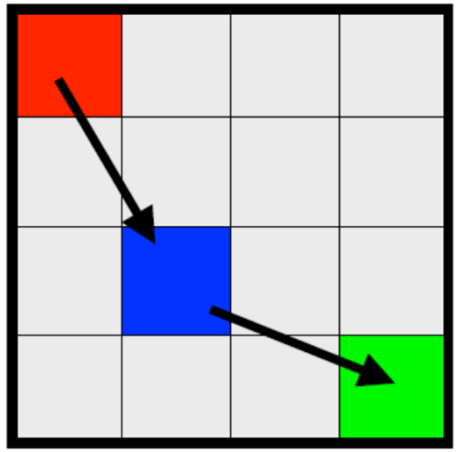
解答

演習11-3

- 100×100のチェス盤の左上のマスから右下のマスまで
  チェスのナイトを動かすのに必要な最小手数はいくつ？

最小手数は66手

- ナイトは2手で右3下3の位置へ移動できる(**マンハッタン距離を小さくするような貪欲法**)
- 2手×33回 = 66手 で 99右99下 へ移動できる
- これにより100×100のチェス盤の右下へ
  移動できる.
- さらに本来は，これ以上は手数を少なく
  できないことを示す必要がある.



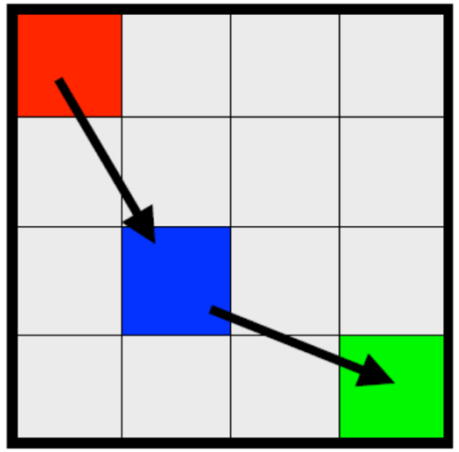
解答

演習11-3

- 100×100のチェス盤の左上のマスから右下のマスまで
  チェスのナイトを動かすのに必要な最小手数はいくつ？

最小手数は66手

- ナイトの初期位置とゴールとのマンハッタン距離は198
  (= 99+99)である.
- ゴールとナイトの位置のマンハッタン距離を一回のナイト
  の移動で3以下しか減らすことができない.
  (下2右1の3回の移動を，1回で移動可能)
- よって，198/3 = 66手未満にはできない.



動的計画法

(授業内のメモを参考に)

フィボナッチの計算

フィボナッチ数列

- 初期条件：fb[0]=0, fb[1]=1
- 漸化式：**fb[n] = fb[n-1] + fb[n-2] (n≧2)**

- フィボナッチ数列の計算プログラムをC言語で書くと…

```
int fb(int n){
    if(n==0) return 0;                  /* 初期条件 */
    else if(n==1) return 1;             /* 初期条件 */
    else return fb(n-2)+fb(n-1);    /* 漸化式 (再帰) */
}
```

フィボナッチの計算

フィボナッチ数列

- 初期条件：fb[0] = 0, = fb[1] = 1
- 漸化式：fb[n] = fb[n-1] + fb[n-2] (n≧2)

- 計算量c[n]：再帰で fb[n] を求めるために用いた漸化式の適用回数

例：c[5]

fb[5]
= (**fb[4]** + fb[3])
= (**fb[3] + fb[2]**) + **fb[3]**
= **fb[3]** + fb[2] + (**fb[2] + fb[1]**)
=(**fb[2] + fb[1]**) + fb[2] + fb[2] + fb[1]（ここまでで4回適用)
= … (3か所のfb[2]の部分に適用しないといけないので…)

フィボナッチの計算

フィボナッチ数列

- 初期条件：fb[0] = 0, = fb[1] = 1
- 漸化式：fb[n] = fb[n-1] + fb[n-2] (n≧2)

- 計算量c[n]：再帰で fb[n] を求めるために用いた漸化式の適用回数

例：c[5]

fb[5]
= **fb[4]** + fb[3]
= (**fb[3] + fb[2]**) + **fb[3]**
= fb[3] + fb[2] + (**fb[2] + fb[1]**)
= …
= fb[1] + fb[0] + fb[1] + fb[1] + fb[0] + (fb[1] + fb[0]) + fb[1]
⇨ **c[5] = 7**

それでは演習です

<u>演習12-1</u>

- 初期条件：fb[0] = 0, = fb[1] = 1
- 漸化式：fb[n] = fb[n-1] + fb[n-2] (n≧2)

- 計算量c[n]：再帰で fb[n] を求めるために用いた漸化式の適用回数

(1) c[4], c[5], c[6], c[7] の値を求めよ.
(2) c[n] を c[n-1] と c[n-2] を用いた漸化式で表せ.
(3) c[n] と fb[n] の関係を求めよ.
(ヒント：(2)の漸化式をうまく変形してみると…)

ちなみに…

- フィボナッチ数列の計算プログラムをC言語で書くと…

```
int fb(int n){
    if(n==0) return 0;                  /* 初期条件 */
    else if(n==1) return 1;             /* 初期条件 */
    else return fb(n-2)+fb(n-1);    /* 漸化式 (再帰) */
}
```

- c[n]の一般項 : $c[n] = \frac{1}{\sqrt{5}}\left\{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right\} - 1$

- c[n]のオーダー : **指数オーダー** → **計算量が膨大**

フィボナッチの計算(動的計画法)

フィボナッチ数列の計算の改良(nが小さいほうから計算)

```
int fb(int n) {
    int i;
    int memo[n];    /* 途中計算をメモする領域 */
    memo[0] = 0;   /* 初期条件 */
    memo[1] = 1;   /* 初期条件 */
    for (i = 2; i <= n; i++) {
        memo[i] = memo[i-1] + memo[i-2];
                                    /* 漸化式 (メモ化) */
    }
    return memo[n];
}
```

動的計画法での計算量

フィボナッチ数列

- 初期条件：fb[0] = 0, = fb[1] = 1
- 漸化式：fb[n] = fb[n-1] + fb[n-2] (n≧2)

- 計算量d[n]：動的計画法で fb[n] を求めるために用いた漸化式の適用回数

例：d[5]

初期条件より fb[0] = 0, fb[1] = 1 である.

値をメモする: memo[0] = fb[0] = 0, memo[1] = fb[1] = 1.

fb[2] = fb[1] + fb[0] = memo[0] + memo[1] = 1.

値をメモする: memo[2] = fb[2] = 1

動的計画法での計算量

フィボナッチ数列

- 初期条件：fb[0] = 0, = fb[1] = 1
- 漸化式：fb[n] = fb[n-1] + fb[n-2] (n≧2)

- 計算量d[n]：動的計画法で fb[n] を求めるために用いた漸化式の適用回数

例：d[5]

fb[3] = fb[2] + fb[1] = memo[2] + memo[1] = 2.
値をメモする: memo[3] = fb[3] = 2

fb[4] = fb[3] + fb[2] = memo[3] + memo[2] = 3.
値をメモする: memo[4] = fb[4] = 3

動的計画法での計算量

フィボナッチ数列

- 初期条件：fb[0] = 0, = fb[1] = 1
- 漸化式：fb[n] = fb[n-1] + fb[n-2] (n≧2)

- 計算量d[n]：動的計画法で fb[n] を求めるために用いた漸化式の適用回数

例：d[5]

fb[5] = fb[4] + fb[3] = memo[4] + memo[3] = 5.

値をメモする: memo[5] = fb[5] = 5

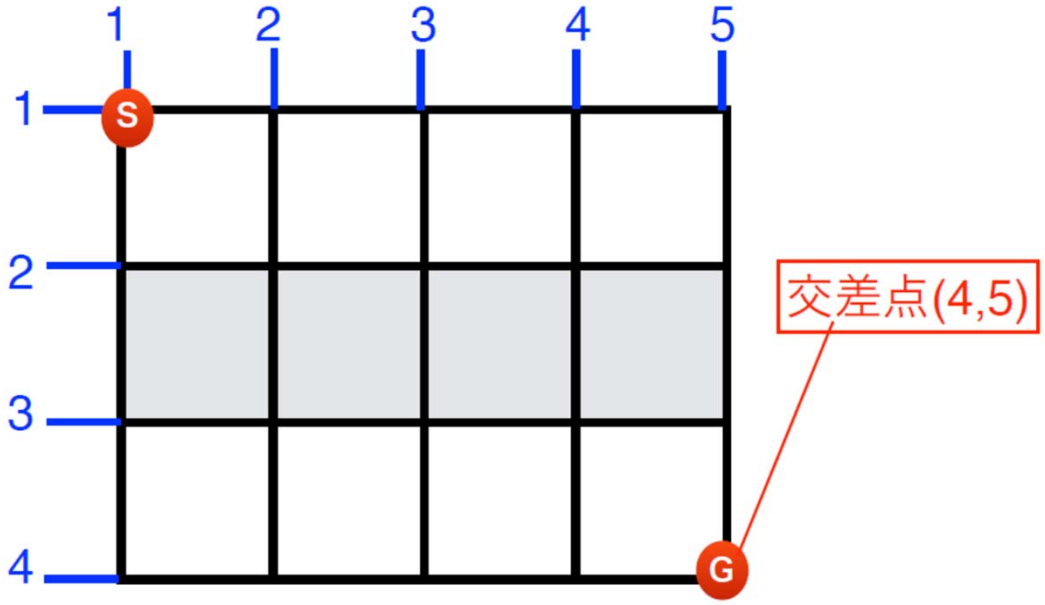
⇨ d[5] = 4

一般式 d[n] = n-1：線形オーダー → 計算時間が短い！

演習で慣れていきましょう

演習12-2

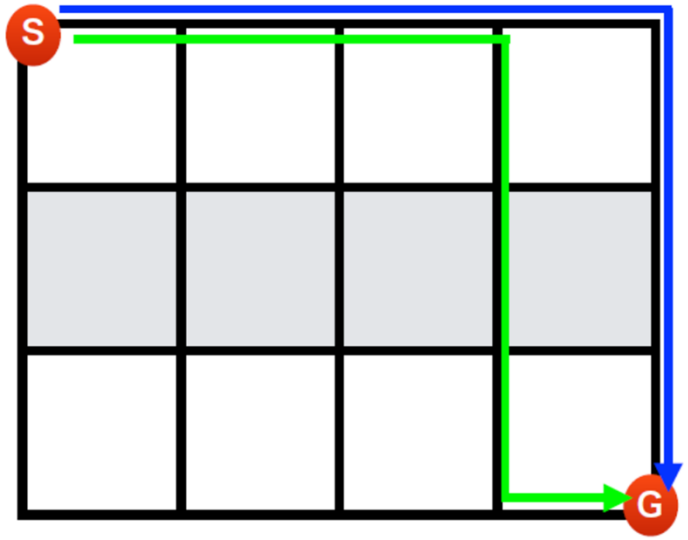
- 以下の水平方向の道を i (1≦ i ≦4) ，垂直方向の道を j (1≦ j ≦5) で表す.
- 交差点(i, j) への最短経路の数 P[i, j]とおく.
- “最短経路は右方向もしくは下方向へ向かう経路からなる”ことを用いてよい.



演習で慣れていきましょう

演習12-2

- 以下の格子状の道を通る S から G への経路について
  (1) P[i, j]に関する初期条件(i=1またはj=1の場合)と
  漸化式を求めよ.
  (2)動的計画法を用いて最短経路の全数を求めよ.
  (3)組合せ的計算で最短経路の全数を求めよ.



解答

演習12-2

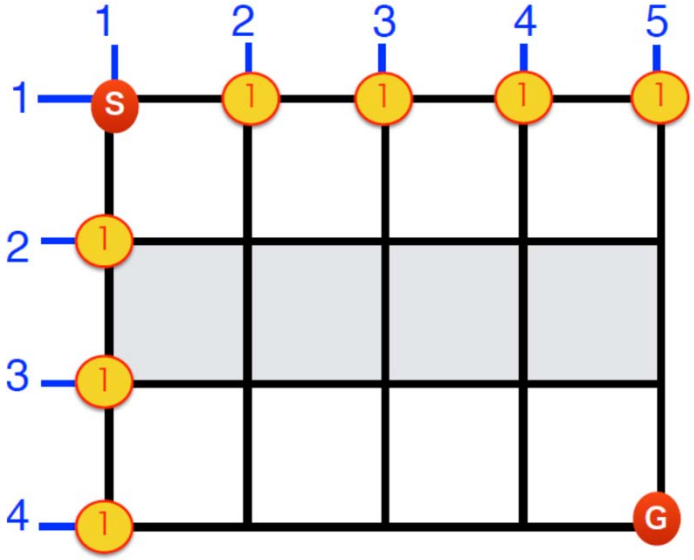
(1)P[i, j]に関する初期条件(i=1またはj=1の場合)と漸化式を求めよ.

(解答)

- 最短経路は右方向もしくは下方向へ向かう部分からなる.

⇨ 初期条件: P[1, j] = P[i, 1] = 1

⇨ 計算結果を交差点にメモしておく



解答

演習12-2

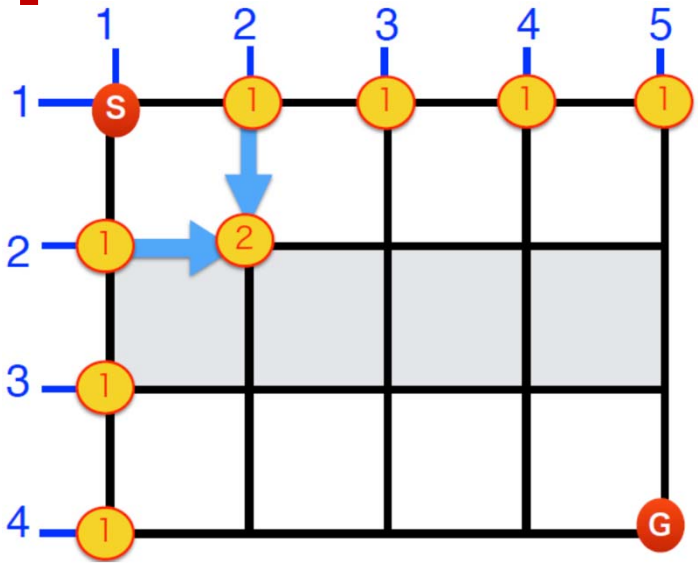
(1)P[i, j]に関する初期条件(i=1またはj=1の場合)と漸化式を求めよ.

(解答)

⇨ 交差点(i, j)を通る最短経路は，直前に(i-1, j)または(i, j-1)を通る

⇨ **漸化式: P[i, j] = P[i-1, j] + P[i, j-1]**

⇨ 計算結果を交差点にメモしておく



解答

演習12-2

(1)P[i, j]に関する初期条件(i=1またはj=1の場合)と漸化式を求めよ.

(解答)

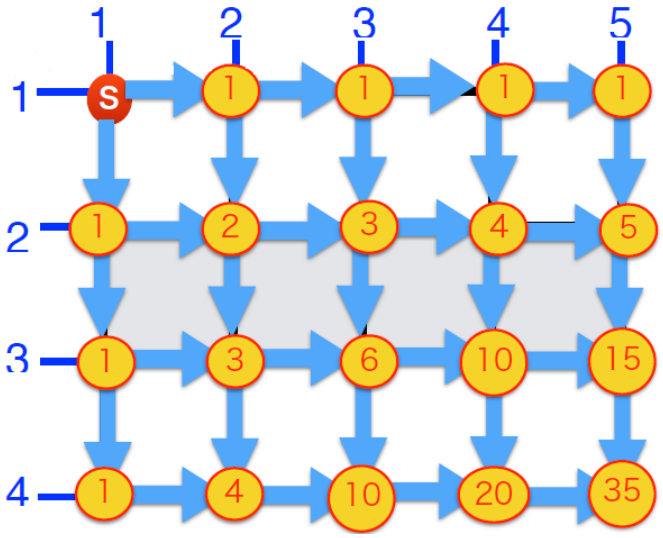
- P[i, j] = P[i-1, j] + P[i, j-1] かつ P[1, j] = P[i, 1] = 1

⇨ 1行目(もしくは1列目)の P[i, j] が分かっており，かつ P[2,1] が分かっているので，2行目の P[i, j] が計算できる.

⇨ …

⇨ 全ての P[i, j] が計算できる.

よって，**P[4,5]=35 (2)の答え**



解答

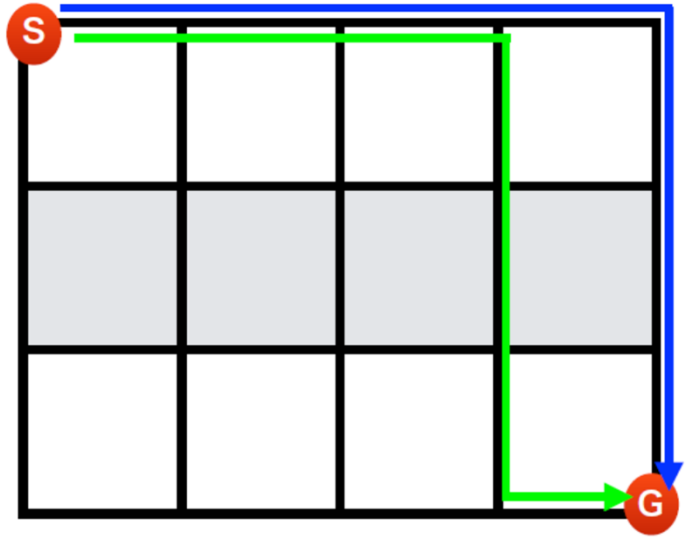
演習12-2

(3)組合せ的計算で最短経路の全数を求めよ.

(解答)

${}_7C_3 = 35$通り

(SからGへ行くには，右へ4、下へ3，合計で7動く必要があり，7動く中で下へ3動く場所を選択するため)



状況が変化した演習です

演習12-3

- 以下の格子状の道を通る S から G への経路について以下に答えよ．ただし×印の場所は通れないとする．

(1)P[i, j]に関する初期条件(i=1またはj=1の場合)と漸化式を求めよ.

(2)動的計画法を用いて最短経路の全数を求めよ.