配列処理Ⅲ

０．目次

１．グラフ表現

１．１　経路数

問題：隣接行列が与えられたとき、節点iから節点jへm回の
移動で移れる方法の数を求めよ。

１．２　最短距離

問題：距離行列が与えられたとき、節点iから節点jへの
最短距離を求めよ。

１．３　最小時間

問題：時間行列が与えられたとき、節点i(1≦i≦n-1)から節点nへの
最小移動時間を求めよ。

２．行列積

n個の行列の積を求めるのに必要な最小乗算回数を考察する。

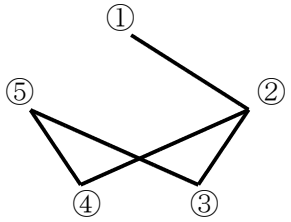
１．グラフ表現

１．１　経路数

n個の節点からなる無向グラフを考察する。
無向グラフは、節点の集合と節点と節点とを結ぶ辺の集合からなる。
節点iから節点jへの辺が存在するとき、配列aの要素a[i][j]を１、存在しないとき0とする。
ただし、自己ループ（節点iから節点iへの辺（1≦i≦n））はないとする。
このn×nの行列を隣接行列Ａとする。

問題：隣接行列が与えられたとき、節点iから節点jへm回の移動で移れる方法の数を求めよ。

グラフ



隣接行列

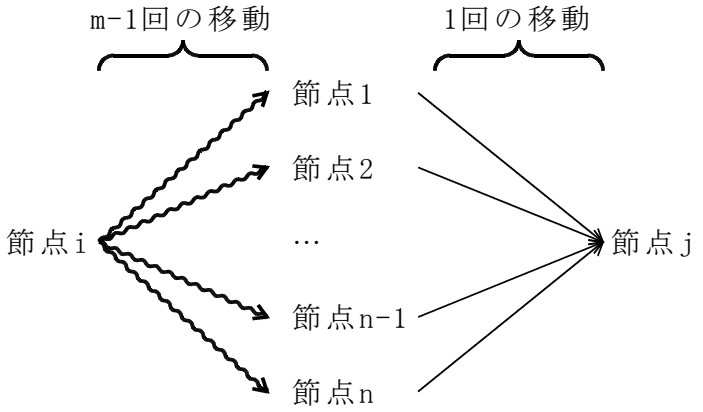
$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

●考え方

m-1回の移動で、節点iから節点jへ移る方法の数がb[i][j]のとき、
m回の移動で節点iから節点jへ移る方法の数c[i][j]は、

m-1回の移動で節点iから節点1へ移動後、節点1から節点jへの移動と、
m-1回の移動で節点iから節点2へ移動後、節点2から節点jへの移動と、
・・・
m-1回の移動で節点iから節点kへ移動後、節点kから節点jへの移動と、
・・・
m-1回の移動で節点iから節点n-1へ移動後、節点n-1から節点jへの移動と、
m-1回の移動で節点iから節点nへ移動後、節点nから節点jへの移動の

和になる。



すなわち、

$$
\begin{aligned}
c[i][j] &= b[i][1]\times a[1][j] \\
&\quad + b[i][2]\times a[2][j] \\
&\quad \cdots \\
&\quad + b[i][k]\times a[k][j] \\
&\quad \cdots \\
&\quad + b[i][n]\times a[n][j] \\
&= \sum_{1\leqq k\leqq n} b[i][k]\times a[k][j]
\end{aligned}
$$

で求められる。ただし、$1\leqq i,j\leqq n$。

A^1：1回の移動で、節点 i から節点 j へ移る方法。ただし、$1\leqq i,j\leqq 5$。

$$
A^1=\begin{pmatrix}
0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0
\end{pmatrix}
$$

A^2：2回の移動で、節点 i から節点 j へ移る方法。ただし、$1\leqq i,j\leqq 5$。

$$
A^2=\begin{pmatrix}
1 & 0 & 1 & 1 & 0 \\
0 & 3 & 0 & 0 & 2 \\
1 & 0 & 2 & 2 & 0 \\
1 & 0 & 2 & 2 & 0 \\
0 & 2 & 0 & 0 & 2
\end{pmatrix}
$$

A^3：3回の移動で、節点 i から節点 j へ移る方法。ただし、$1\leqq i,j\leqq 5$。

$$
A^3=\begin{pmatrix}
0 & 3 & 0 & 0 & 2 \\
3 & 0 & 5 & 5 & 0 \\
0 & 5 & 0 & 0 & 4 \\
0 & 5 & 0 & 0 & 4 \\
2 & 0 & 4 & 4 & 0
\end{pmatrix}
$$

A^4：4回の移動で、節点 i から節点 j へ移る方法。ただし、$1\leqq i,j\leqq 5$。

$$
A^4=\begin{pmatrix}
3 & 0 & 5 & 5 & 0 \\
0 & 13 & 0 & 0 & 10 \\
5 & 0 & 9 & 9 & 0 \\
5 & 0 & 9 & 9 & 0 \\
0 & 10 & 0 & 0 & 8
\end{pmatrix}
$$

●プログラム(d111.c)

```
/* << d111.c >> */
/* 隣接行列が与えられたとき、節点iから節点jへm回の移動で移れる
   方法の数を求める。*/
#include<stdio.h>
#include <stdlib.h>
#define N 99 /* 節点の最大個数。*/

int main() {
  int a[N+1][N+1], /* 隣接行列Ａ。*/
      b[N+1][N+1], /* 行列Ｂ＝Ａ**m。*/
      c[N+1][N+1], /* 作業用配列。*/
      i,j,k,
      m,           /* 移動回数。*/
      n,           /* 節点数。*/
      w;

 /* 節点数nと隣接行列Ａの読み込み。*/
  scanf("%d",&n);
  if( (n <= 0)||(n > N) ) { exit(0); }
  for( i=1; i<=n; i++ ) {
    for( j=1; j<=n; j++ ) { scanf("%d",&a[i][j]); }
  }

  /* 行列Ｂの初期化。*/
  for( i=1; i<=n; i++ ) {
    for( j=1; j<=n; j++ ) {
      b[i][j] = 0; if( i == j ) { b[i][j] = 1; }
    }
  }

  /* Ａ**mを計算する。*/
  for( m=1; m<=n-1; m++ ) {
    for( i=1; i<=n; i++ ) {
      for( j=1; j<=n; j++ ) {
        w = 0;
        for( k=1; k<=n; k++ ) {
          w = w + b[i][k]*a[k][j];
        }
        c[i][j] = w;
      }
    }
    for( i=1; i<=n; i++ ) {
      for( j=1; j<=n; j++ ) { b[i][j] = c[i][j]; }
    }

    /* 行列Ｂ＝Ａ**mの表示。*/
    printf("Ａ**%d¥n",m);
    for( i=1; i<=n; i++ ) {
```

```
49          for( j=1; j<=n; j++ ) { printf("%4d",b[i][j]); };
50          printf("¥n");
51        }
52      }
53  }
```

実行結果

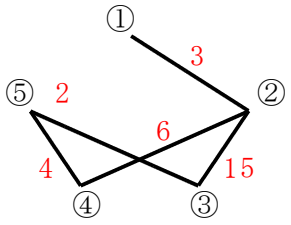
```
% cc d111.c
% ./a.out
5
0 1 0 0 0
1 0 1 1 0
0 1 0 0 1
0 1 0 0 1
0 0 1 1 0
A**1
   0   1   0   0   0
   1   0   1   1   0
   0   1   0   0   1
   0   1   0   0   1
   0   0   1   1   0
A**2
   1   0   1   1   0
   0   3   0   0   2
   1   0   2   2   0
   1   0   2   2   0
   0   2   0   0   2
A**3
   0   3   0   0   2
   3   0   5   5   0
   0   5   0   0   4
   0   5   0   0   4
   2   0   4   4   0
A**4
   3   0   5   5   0
   0  13   0   0  10
   5   0   9   9   0
   5   0   9   9   0
   0  10   0   0   8
```

１．２　最短距離

n個の節点からなる無向グラフを考察する。無向グラフは、節点の集合と節点と節点とを結ぶ辺の集合からなる。節点iから節点jへの辺が存在するとき、その距離dを要素a[i][j]の値とし、存在しないとき、∞を要素a[i][j]の値とする。
ただし、自己ループ（節点iから節点iへの辺（1≦i≦n））はないとする。
このn×nの行列を距離行列Aとする。

問題：距離行列が与えられたとき、節点iから節点jへの最短距離を求めよ。

グラフ



距離行列

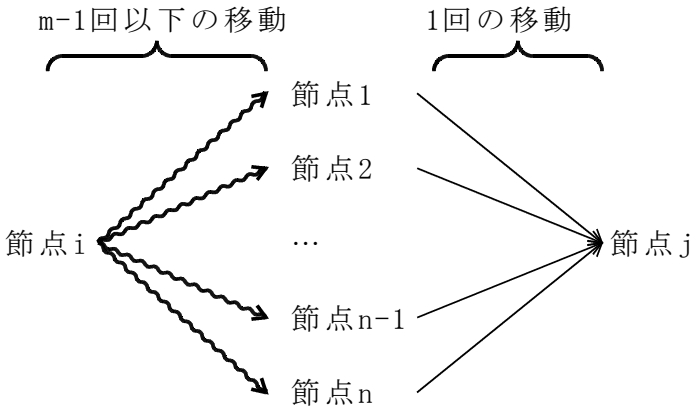
$$A = \begin{pmatrix} \infty & 3 & \infty & \infty & \infty \\ 3 & \infty & 15 & 6 & \infty \\ \infty & 15 & \infty & \infty & 2 \\ \infty & 6 & \infty & \infty & 4 \\ \infty & \infty & 2 & 4 & \infty \end{pmatrix}$$

●考え方

m-1回以下の移動での節点ｉから節点ｊへの最短距離がb[i][j]のとき、
m回以下の移動での最短距離c[i][j]は、

節点iから節点1への最短距離と、節点1から節点jへの距離の和 b[i][1]+a[1][j]
節点iから節点2への最短距離と、節点2から節点jへの距離の和 b[i][2]+a[2][j]
・・・
節点iから節点kへの最短距離と、節点kから節点jへの距離の和 b[i][k]+a[k][j]
・・・
節点iから節点n-1への最短距離と節点n-1から節点jへの距離の和 b[i][n-1]+a[n-1][j]
節点iから節点n-1への最短距離と、節点n-1から節点jへの距離の和 b[i][n]+a[n][j]

の中の最小値となる。



すなわち、

```
c[i][j] = min{ b[i][1]+a[1][j],
               b[i][2]+a[2][j],
               ・・・,
               b[i][k]+a[k][j],
               ・・・,
               b[i][n-1]+a[n-1][j],
               b[i][n]+a[n][j] }

c[i][j] = min { b[i][k] + a[k][j] }
          1≦k≦n
```

で求められる。ただし、1≦i,j≦n。

●プログラム(d121.c)

```
/* << d121.c >> */
/* 距離行列が与えられたとき、節点iから節点jへの最短距離を求める。*/
#include <stdio.h>
#include <stdlib.h>
#define N 99 /* 節点の最大個数。*/

int main() {
  int a[N+1][N+1], /* 距離行列Ａ。*/
      b[N+1][N+1], /* 最短行列Ｂ。*/
      c[N+1][N+1], /* 作業配列。*/
      i,j,k,
      m,           /* 移動回数。*/
      n,           /* 節点数。*/
      w;

  /* 節点数nと距離行列Ａの読み込み。*/
  scanf("%d",&n);
  if( (n <= 0)||(n > N) ) { exit(0); }
  for( i=1; i<=n; i++ ) {
    for( j=1; j<=n; j++ ) { scanf("%d",&a[i][j]); }
  }

  /* 最短行列Ｂの初期化。*/
  for( i=1; i<=n; i++ ) {
    for( j=1; j<=n; j++ ) { b[i][j] = a[i][j]; }
  }

  /* 最短行列Ｂの表示。*/
  printf("行列Ｂ（移動回数 %d 以下）¥n",1);
  for( i=1; i<=n; i++ ) {
    for( j=1; j<=n; j++ ) { printf("%5d",b[i][j]); };
    printf("¥n");
  }
```

```
34
35    /* 最短行列Ｂを計算する。*/
36    for( m=2; m<=n-1; m++ ) {
37      for( i=1; i<=n; i++ ) {
38        for( j=1; j<=n; j++ ) {
39          w = b[i][j];
40          for( k=1; k<=n; k++ ) {
41            if( b[i][k]+a[k][j] < w) {
42              w = b[i][k]+a[k][j];
43            }
44          }
45          c[i][j] = w;
46        }
47      }
48      for( i=1; i<=n; i++ ) {
49        for( j=1; j<=n; j++ ) { b[i][j] = c[i][j]; }
50      }
51
52      /* 最短行列Ｂの表示。*/
53      printf("行列Ｂ（移動回数 %d 以下）¥n",m);
54      for( i=1; i<=n; i++ ) {
55        for( j=1; j<=n; j++ ) { printf("%5d",b[i][j]); };
56        printf("¥n");
57      }
58    }
59  }
```

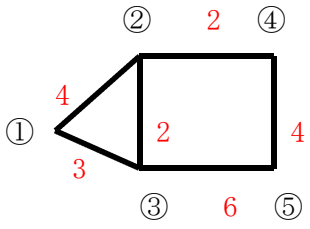
実行結果

```
% cc d121.c
% ./a.out
5
9999 3 9999 9999 9999
3 9999 15 6 9999
9999 15 9999 9999 2
9999 6 9999 9999 4
9999 9999 2 4 9999
行列Ｂ（移動回数 1 以下）
 9999    3 9999 9999 9999
    3 9999   15    6 9999
 9999   15 9999 9999    2
 9999    6 9999 9999    4
 9999 9999    2    4 9999
行列Ｂ（移動回数 2 以下）
    6    3   18    9 9999
    3    6   15    6   10
   18   15    4    6    2
    9    6    6    8    4
 9999   10    2    4    4
行列Ｂ（移動回数 3 以下）
    6    3   18    9   13
    3    6   12    6   10
   18   12    4    6    2
    9    6    6    8    4
   13   10    2    4    4
行列Ｂ（移動回数 4 以下）
    6    3   15    9   13
    3    6   12    6   10
   15   12    4    6    2
    9    6    6    8    4
   13   10    2    4    4
```

１．３　最小時間

n個の節点からなる無向グラフを考察する。
無向グラフは、節点の集合と節点と節点とを結ぶ辺の集合からなる。
節点iら節点jへの辺が存在するとき、その移動時間を要素t[i][j]の値とし、
存在しないとき、∞を要素t[i][j]の値とする。
ただし、自己ループ（節点iから節点iへの辺（1≦i≦n））はないとする。
このn×nの行列を時間行列Tとする。

問題：時間行列が与えられたとき、節点i(1≦i≦n-1)から節点nへの最小移動時間を求めよ。



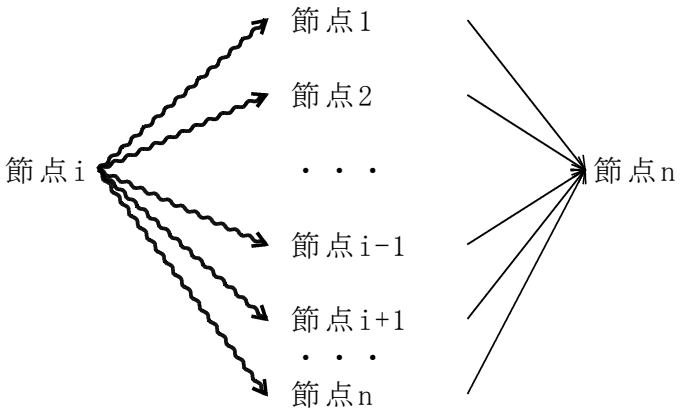
$$T = \begin{pmatrix} \infty & 4 & 3 & \infty & \infty \\ 4 & \infty & 2 & 2 & \infty \\ 3 & 2 & \infty & \infty & 6 \\ \infty & 2 & \infty & \infty & 4 \\ \infty & \infty & 6 & 4 & \infty \end{pmatrix}$$

時間行列

節点iから節点n(=5)までの最小移動時間をa(i)とする。節点iから節点1,2,i-1,i+1,…,n-1を経由して節点nへ至るとすると、

a(i) = min{節点iから節点1を通り,節点1から節点nまでの最小移動時間,
節点iから節点2を通り,節点2から節点nまでの最小移動時間,
・・・,
節点iから節点i-1を通り,節点i-1から節点nまでの最小移動時間,
節点iから節点i+1を通り,節点i+1から節点nまでの最小移動時間,
・・・,
節点iから節点n-1を通り,節点n-1から節点nまでの最小移動時間,
節点iから節点nまでの移動時間}

となる。



すなわち、節点i(1≦i≦n-1)に対して、

$$a(i)=\min_{j\neq i}\{\ t(i,j)+a(j)\ \}$$

が成り立つ。ただし、a(n)＝0。

●考え方１

具体的にはつぎのようになる。

```
a(5)＝0
a(4)＝min{ t(4,2)+a(2),t(4,5)+a(5) }
a(3)＝min{ t(3,1)+a(1),t(3,2)+a(2),t(3,5)+a(5) }
    ＝min{ 3+a(1),2+a(2),6 }
a(2)＝min{ t(2,1)+a(1),t(2,3)+a(3),t(2,4)+a(4) }
    ＝min{ 4+a(1),2+a(3),2+a(4) }
a(1)＝min{ t(1,2)+a(2),t(1,3)+a(3) }
    ＝min{ 4+a(2),3+a(3) }
```

（ア）まず、a(4)＝min{ t(4,2)+a(2),t(4,5)+a(5) }
　　＝min{ 2+a(2),4 }

a(2)≧2 より、a(4) = 4 が決まる。

（イ）a(4)(=4)をa(2)に代入すると、

a(2)＝min{ 4+a(1),2+a(3),6 }

となる。

（ウ）a(2)をa(1)に代入すると、

```
a(1)＝min{ 4+a(2),3+a(3) }
    ＝min{ 8+a(1),6+a(3),10,3+a(3) }
    ＝min{ 3+a(3),10 }
```

となる。

（エ）a(2)をa(3)に代入すると、

```
a(3)＝min{ 3+a(1),2+a(2),6 }
    ＝min{ 3+a(1),6+a(1),5+a(3),8,6 }
    ＝min{ 3+a(1),5+a(3),6 }
    ＝min{ 3+a(1),6 }
```

となる。

（オ）a(3)をa(1)に代入すると、

a(1)＝min{ 3+a(3),10 }
＝min{ 6+a(1),9,10 }
＝9

となる。すなわち、地点1から地点5への最小移動時間a(1)は9となる。

上式で、等号が成り立つ経路を選んで行けば、最小の経路が求められる。
この場合は、1->3->5となる。

a(3)＝min{3+a(1),6}　より、a(3)＝6

a(2)＝min{4+a(1),2+a(3),6}　より、a(2)＝6

となる。

●考え方２

近似解を繰り返し関係式に代入し、近似解が変わらなくなったら解とする。

手順１：a(1),a(2),a(3),a(4)を大きな数(例えば、9999)、a(5)を0にしておく。

手順２：つぎの関係式を使って、

b(5)＝a(5)
b(4)＝min{ t(4,2)+a(2),t(4,5)+a(5) }
b(3)＝min{ t(3,1)+a(1),t(3,2)+a(2),t(3,5)+a(5) }
b(2)＝min{ t(2,1)+a(1),t(2,3)+a(3),t(2,4)+a(4) }
b(1)＝min{ t(1,2)+a(2),t(1,3)+a(3) }

b(1),b(2),b(3),b(4),b(5)を求める。

手順３：a(i)＝b(i) (1≦i≦5)ならば、解 a(i) (1≦i≦5)が得られたとし、終了する。
その他の場合、b(i) (1≦i≦5)をa(i) (1≦i≦5)に代入し、手順２に戻る。

	a(1)	a(2)	a(3)	a(4)	a(5)
手順１	9999	9999	9999	9999	0
手順２，３	9999	9999	6	4	0
手順２，３	9	6	6	4	0
手順２，３	9	6	6	4	0

●プログラム(d131.c)

```
/* << d131.c >> */
/* 時間行列が与えられたとき、節点iから節点jへの最小時間を求める。*/
#include <stdio.h>
#include <stdlib.h>
#define N 99 /* 節点の最大個数。*/

int main() {
  int a[N+1],       /* 節点iから節点nまでの最小移動時間をa[i]。*/
      b[N+1],       /* 作業用配列。*/
      i,j,
      n,            /* 節点数。*/
      t[N+1][N+1], /* 時間行列T。*/
      w;

  /* 節点数nの読み込み。*/
  scanf("%d",&n);
  if( (n <= 0)||(n > N) ) { exit(0); }

  /* 時間行列Tの読み込み。*/
  for( i=1; i<=n; i++ ) {
    for( j=1; j<=n; j++ ) { scanf("%d",&t[i][j]); }
  }

  /* 配列aの初期設定。*/
  for( i=1; i<n; i++ ) { a[i] = 9999; };
  a[n] = 0;

  /* 配列aの表示。*/
  for( j=1; j<=n; j++ ) { printf("%5d",a[j]); }
  printf("¥n");

  while( 1 ) {
    /* 配列aから配列bを計算。*/
    for( i=1; i<=n; i++ ) {
      b[i] = a[i];
      for( j=1; j<=n; j++ ) {
          w = t[i][j] + a[j];
          if( w < b[i] ) { b[i] = w;}
      }
    }
    for( i=1; i<=n; i++ ) {
      if( a[i] != b[i] ) { break; }
    }

    /* 配列bの表示。*/
    for( j=1; j<=n; j++ ) { printf("%5d",b[j]); }
    printf("¥n");
    if( i > n ) { break; /* a[*]とb[*]が一致すると終了。*/ }

    /* 配列bを配列aに代入。*/
```

```
51      for( j=1; j<=n; j++ ) { a[j] = b[j]; }
52    }
53  }
```

実行結果

```
% cc d131.c
% ./a.out
5
9999 4 3 9999 9999
4 9999 2 2 9999
3 2 9999 9999 6
9999 2 9999 9999 4
9999 9999 6 4 9999
 9999 9999 9999 9999    0
 9999 9999    6    4    0
    9    6    6    4    0
    9    6    6    4    0
```

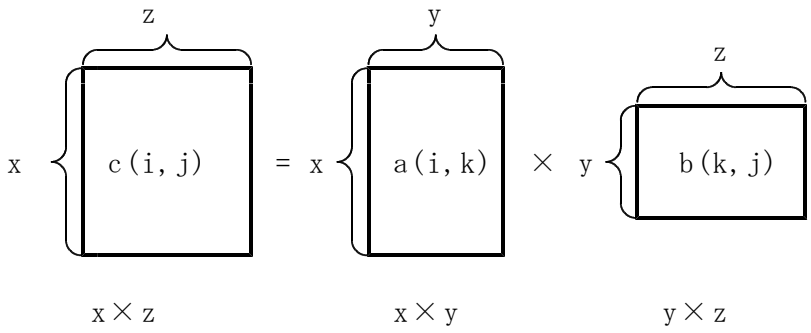
２．行列積

n個の行列の積を求めるのに必要な最小乗算回数を考察する。

x×yの行列Ａとy×zの行列Ｂの積Ｃは、x×zの行列で、

$$c(i,j) = \sum_{k=1}^{y} a(i,k) \times b(k,j) \quad (1 \leqq i \leqq x, 1 \leqq j \leqq z)$$

と定義される。このとき、乗算の回数は、x×y×z回となる。



●例

４個の行列（２×３の行列Ａ、３×４の行列Ｂ、４×５の行列Ｃ、５×６の行列Ｄ）を考える。

求める最小乗算回数をf(ABCD)とする。

ABCDの計算方法は、３通り（(ABC)D,(AB)(CD),A(BCD)）考えられる。

(ABC)D において、ABC は、2×5の行列
D　は、5×6の行列
したがって、(ABC)Dには、2×5×6の乗算が必要。

(AB)(CD) において、AB は、2×4の行列
CD は、4×6の行列
したがって、(AB)(CD)には、2×4×6の乗算が必要。

A(BCD) において、A　は、2×3の行列
BCD は、3×6の行列
したがって、A(BCD)には、2×3×6の乗算が必要。

まとめると、

f(ABCD) = min{ f(ABC)+２×５×６,　(ABC)Dに対応
f(AB)+f(CD)+２×４×６,　(AB)(CD)に対応
f(BCD)+２×３×６ }　A(BCD)に対応

が成り立つ。

ABCの計算方法は２通り（(AB)C,A(BC)）、BCDの計算方法は２通り（(BC)D,B(CD)）考えられる。

(AB)C において、AB は、2×4の行列
C は、4×5の行列
したがって、(AB)Cには、2×4×5の乗算が必要。

A(BC) において、A は、2×3の行列
BC は、3×5の行列
したがって、A(BC)には、2×3×5の乗算が必要。

まとめると、

f(ABC) = min{ f(AB)+２×４×５, (AB)Cに対応
f(BC)+２×３×５ } A(BC)に対応

(BC)D において、BC は、3×5の行列
D は、5×6の行列
したがって、(BC)Dには、3×5×5の乗算が必要。

B(CD) において、B は、3×4の行列
CD は、4×6の行列
したがって、B(CD)には、3×4×6の乗算が必要。

まとめると、

f(BCD) = min{ f(BC)+３×５×６, (BC)Dに対応
f(CD)+３×４×６ } B(CD)に対応

が成り立つ。AB,BC,CDはそれぞれ１通りであるから、

f(AB) = ２×３×４
f(BC) = ３×４×５
f(CD) = ４×５×６

が成り立つ。これらの結果を逆に見ていくと、

f(ABC) = min{ 64,90 } = 64
f(BCD) = min{ 150,192 } = 150

f(ABCD) = min{ 124,192,186 } = 124

を得る。

計算手順は、①②③④⑤⑥の順に行えばよい。

	A	B	C	D
A		① f(AB)	④ f(ABC)	⑥ f(ABCD)
B			② f(BC)	⑤ f(BCD)
C				③ f(CD)
D				

（参考）

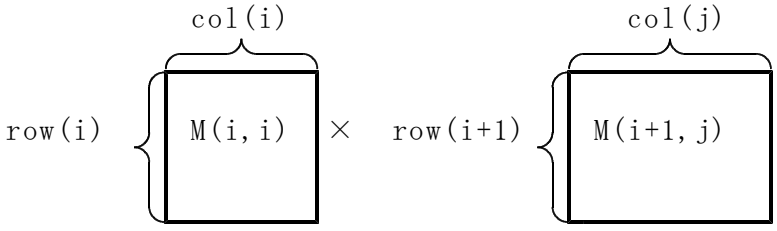
計算方法	乗算回数
((AB)C)D	124
(A(BC))D	150
(AB)(CD)	192
A((BC)D)	186
A(B(CD))	228

i番目の行列をM(i), p(i,j)をi番目の行列M(i)からj番目の行列m(j)の行列積で必要な最小乗算回数とする。

i番目の行列からj番目の行列までの行列積を、M(i,j)とすると、M(i,j)は、つぎのいずれかの方法で計算される。

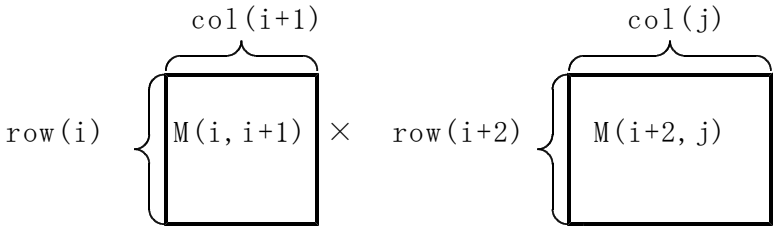
$$M(i,j) = M(i)\times M(i+1)\times\cdots M(j) = \begin{cases} M(i,i) \times M(i+1,j) \\ M(i,i+1) \times M(i+2,j) \\ \quad \cdot\cdot\cdot \\ M(i,j-2) \times M(j-1,j) \\ M(i,j-1) \times M(j,j) \end{cases}$$

・M(i,i)×M(i+1,j)の乗算回数
i番目の行列の行数をrow(i)、列数をcol(i)とする。



求める乗算回数は、row(i)×row(i+1)×col(j) + p(i,i) + p(i+1,j)

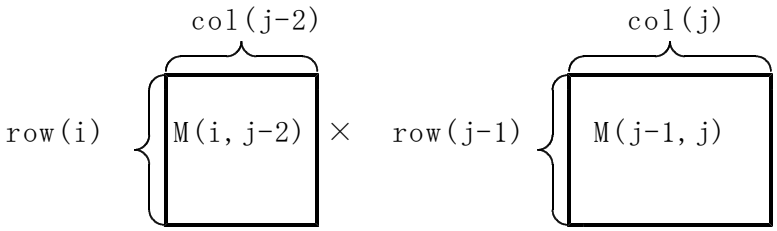
・M(i,i+1)×M(i+2,j)の乗算回数



求める乗算回数は、row(i)×row(i+2)×col(j) + p(i,i+1) + p(i+2,j)

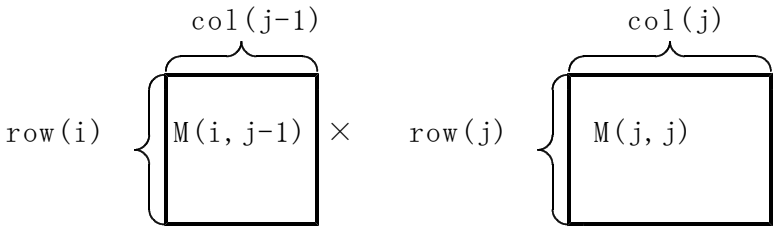
・・・

・M(i,j-2)×M(j-1,j)の乗算回数



求める乗算回数は、row(i)×row(j-1)×col(j) + p(i,j-2) + p(j-1,j)

・M(i,j-1)×M(j,j)の乗算回数



求める乗算回数は、row(i)×row(j)×col(j) + p(i,j-1) + p(j,j)

となる。

したがって、上の乗算回数のなかで、最小のものが求めるp(i,j)となる。

５個の行列をＡ，Ｂ，Ｃ，Ｄ，Ｅとすると、p(i,j)の計算手順は、①～⑩の順になる。

	A	B	C	D	E
A		①	⑤	⑧	⑩
B			②	⑥	⑨
C				③	⑦
D					④
E					

●プログラム(d211.c)

```
1  /* << d211.c >> */
2  /* n個の行列の積を求めるのに必要な最小乗算回数を考察する。*/
3  #include <stdio.h>
4  #include <stdlib.h>
5  #define N 99 /* 行列の最大個数。*/
6
7  int main() {
8    int p[N+1][N+1], /* a(i,j)をi番目の行列からj番目の行列の
9                        行列積で必要な最小乗算回数。*/
10       col[N+1],    /* col[i]:i番目の行列の列数。*/
11       h,i,j,k,
12       m,
13       min,         /* 最小乗算回数。*/
14       n,           /* 行列の個数。*/
15       row[N+1],    /* row[i]:i番目の行列の行数。*/
16       w;
17
18   /* 行列の個数を入力。*/
19   scanf("%d",&n);
20   if( (n <= 0)||(n > N) ) { exit(0); }
21
22   /* 行列の行と列の次数を入力。*/
23   for( i=1; i<=n; i++ ) { scanf("%d%d",&row[i],&col[i]); }
24
25   /* 行列の行と列の次数を表示。*/
26   printf("行列の個数：%d¥n",n);
27   for( i=1; i<=n; i++ ) {
28     printf("行列 %d  (行：%d 列：%d) ¥n",i,row[i],col[i]);
29   }
30
31   /* 初期設定。*/
32   for( i=1; i<=n; i++ ) { p[i][i] = 0 ; }
33
34   /* 最小乗算回数を求める。*/
35   for( k=1; k<=n-1; k++ ) {
36     for( i=1; i<=n-1; i++ ) {
37       j = i + k;
38       min = row[i]*row[i+1]*col[j] + p[i+1][j];
39       for( m=i+2; m<=j; m++ ) {
40         w = row[i]*row[m]*col[j] + p[i][m-1] + p[m][j];
41         if( w < min ) { min = w; }
42       }
43       p[i][j] = min;
44     }
45
46     /* 配列aの表示。*/
47     printf("配列Ａ¥n");
48     for( i=1; i<=n; i++ ) {
```

```
49          for( j=1; j<=i-1; j++ ) { printf("     *"); }
50          h = i+k;
51          if( h > n ) { h = n; }
52          for( j=i; j<=h; j++ ) { printf("%6d",p[i][j]); }
53          for( j=h+1; j<=n; j++ ) { printf("     *"); }
54          printf("¥n");
55        }
56      }
57      printf("最小乗算回数：%d¥n",p[1][n]);
58    }
```

実行結果

```
% cc d211.c
% ./a.out
4
2 3
3 4
4 5
5 6
行列の個数：4
行列 1  (行：2 列：3)
行列 2  (行：3 列：4)
行列 3  (行：4 列：5)
行列 4  (行：5 列：6)
配列Ａ
     0    24     *     *
     *     0    60     *
     *     *     0   120
     *     *     *     0
配列Ａ
     0    24    64     *
     *     0    60   150
     *     *     0   120
     *     *     *     0
配列Ａ
     0    24    64   124
     *     0    60   150
     *     *     0   120
     *     *     *     0
最小乗算回数：124
% ./a.out
4
5 6
4 5
3 4
2 3
＜＜途中省略＞＞
最小乗算回数：114
```