アルゴリズムとデータ構造II Ex09 Strassen

Strassen's Algorithm

スケルトンコード`strassen.c`を用いて、Strassen's Algorithmによる行列乗算を行うプログラムを作れ。

ただし、

- 入力行列サイズは2の冪乗の正方行列であるとする。
  - それ以外は未定義のままで良い。
- 正方行列の各要素はint型とする。

スケルトンコードの中で、完成させるのは、

- Strassen's algorithmを適用する`void strassen(int \*\*, int \*\*, int \*\*, int)`の中身
- 通常の行列乗算を行う`void n3_product(int \*\*, int \*\*, int \*\*, int)`の中身

だけである。

使い方

プログラム実行時に、引数を与えることで行列の初期化方法を指定できる。

- 指定したサイズの正方行列を自動生成して計算(引数は自然数)

  `./a.out 8`

- 使用者が自分で用意したファイルを読み込んで計算(-1を引数とする。)

  ```
  $ ./a.out -1
  Enter the filename of matA:
  =>mat1
  Enter the filename of matB:
  =>mat2
  ```

  自分で入力する場合は、以下のよう記述形式でファイルを作ること。

  - 1行目に正方行列のサイズ
  - 2行目以降に行列の各要素

```
4
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

実行結果が正しいかどうかを目で確認するのは現実的ではないので、通常の行列乗算とStrassen's Algorithmの結果をそれぞれ保存しておいて、`diff`コマンドなりを使って、ファイルの差分が0であるかを確認すればよい。

また、例えば、以下のように値が一致しない要素の差分を表示するようにコードを書いてある。

- int型で計算しているので誤差は生まれない。(正しい結果になっているならば、何も表示されない)

```
$ ./a.out 32
Error at(1,3) : 4
Error at(2,5) : 24
Error at(4,6) : -33
Error at(8,7) : -18
```

作成したプログラムに以下の2つの行列を入力して、計算があっているかの確認をしてから、課題に取り組むこと。

- 行列1つ目

```
8
1 2 3 4 5 6 7 8
2 3 4 5 6 7 8 1
3 4 5 6 7 8 1 2
4 5 6 7 8 1 2 3
5 6 7 8 1 2 3 4
6 7 8 1 2 3 4 5
7 8 1 2 3 4 5 6
8 1 2 3 4 5 6 7
```

- 行列2つ目

```
8
1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1
```

2つ目が単位行列なので、計算結果は、1つ目の行列に一致する。