2024年　7 月 19日

まず、ホームディレクトリの下にC_PROG/15というディレクトリを作ってその中で作業する

課題に必要なデータを以下の様にダウンロードしておく

```
git clone https://github.com/NGS-maps/PracticeTemplate
git clone https://github.com/NGS-maps/PracticeRing
```

課題提出ファイル名、tar,gzする場合はディレクトリ名はいつもの方法で123nakamura15_6の様に、

メールのタイトルは、中村15_6の様に１～6のうちのどれか一つを８月１８日までにご提出ください（早い方が良い）

課題1. 15_1

以下のプログラムの###部を補って、再帰関数を使ってグラフをトレースするプログラム(trace_graph.c)を完成するひな形のプログラム、trace_graph_template.cを以下の手順で自分のディレクトリにtrace_graph.cと言う名前でコピーし編集する。ここで、再帰関数の動きを良く理解しておく必要がある。

```
% cp trace_graph_template.c trace_graph.c
```

========== trace_graph.c ==========

```
#include <stdio.h>
#include <stdlib.h>
#define  N  8

int a[N+1][N+1] = {{0,0,0,0,0,0,0,0,0},
                   {0,0,1,0,0,0,0,0,0},
                   {0,1,0,1,1,0,0,0,0},
                   {0,0,1,0,0,0,0,1,0},
                   {0,0,1,0,0,1,0,0,0},
                   {0,0,0,0,1,0,1,0,0},
                   {0,0,0,0,0,1,0,1,1},
                   {0,0,0,1,0,0,1,0,1},
                   {0,0,0,0,0,0,1,1,0}};

int v[N+1];

void visit(int);


int main(void)
{
int i;
for(i=1;i<=N;i++)
 {
 v[i] = 0;
 }
visit(1);
printf("\n");
}


void visit(int now)
{
int next;
v[###] = 1;
for(next = 1; next <= ###; next ++)
 {
 if((a[###][###] == 1) && (v[###] == 0))
  {
  printf("%d->%d ",now,next);
  visit(###);
  }
 }
}
```

==========

```
gcc -o trace_graph trace_graph.c
./trace_graph
1->2 2->3 3->7 7->6 6->5 5->4 6->8
```

課題2. 15_2

以下のプログラムの###部を補って、再帰関数を使って全てのノードを起点としてグラフをトレースするプログラム(trace_graph2.c)を完成する。

% cp trace_graph2_template.c trace_graph2.c

```
====================================== trace_graph2.c ============================
#include <stdio.h>
#include <stdlib.h>
#define  N  8

int a[N+1][N+1] = {{0,0,0,0,0,0,0,0,0},
                   {0,0,1,0,0,0,0,0,0},
                   {0,1,0,1,1,0,0,0,0},
                   {0,0,1,0,0,0,0,1,0},
                   {0,0,1,0,0,1,0,0,0},
                   {0,0,0,0,1,0,1,0,0},
                   {0,0,0,0,0,1,0,1,1},
                   {0,0,0,1,0,0,1,0,1},
                   {0,0,0,0,0,0,1,1,0}};

int v[N+1];

void visit(int);


int main(void)
{
int i,j;
for(j=1;j<=N;j++)
 {
 for(i=1;i<=N;i++)
  {
  v[i] = 0;
  }
 visit(j);
 printf("\n");
 }
}


void visit(int now)
{
int next;
v[###] = 1;
for(next = 1; next <= ###; next ++)
 {
 if((a[###][###] == 1) && (v[###] == 0))
  {
  printf("%d->%d, ",now,next);
  visit(###);
  }
 }
}
================================================================================
```

```
gcc -o trace_graph2 trace_graph2.c
./trace_graph2
1->2 2->3 3->7 7->6 6->5 5->4 6->8
2->1 2->3 3->7 7->6 6->5 5->4 6->8
3->2 2->1 2->4 4->5 5->6 6->7 7->8
4->2 2->1 2->3 3->7 7->6 6->5 6->8
5->4 4->2 2->1 2->3 3->7 7->6 6->8
6->5 5->4 4->2 2->1 2->3 3->7 7->8
7->3 3->2 2->1 2->4 4->5 5->6 6->8
```

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

課題3. 15_3

再帰関数を使って非連結グラフをトレースし、それぞれの連結グラフの番号と、含まれるノードのリストを表示するプログラム(trace_graph3.c)を完成する
ひな形のプログラム、trace_graph3_template.cを以下の手順でコピーし編集する。

% cp trace_graph3_template.c trace_graph3.c

このひな形はグラフが課題２、３のものと違い非連結グラフになっているので注意！

```
% gcc -o trace_graph3 trace_graph3.c
% ./trace_graph3
1 : 1 2 3 5 4
2 : 6 7 8
```

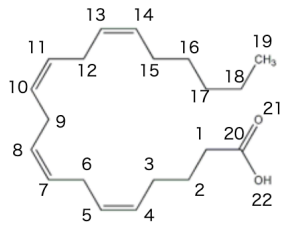
実行結果が下記のようでも可

```
% ./trace_graph3
1->2, 2->3, 3->5, 5->4
6->7, 7->8
```

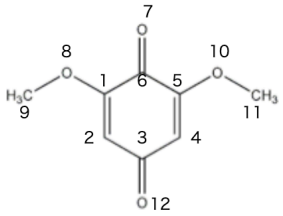
========== ==========

課題３までのグラフトレースのアルゴリズムを使って、以下の図の様な分子構造を含むmolファイルについて、リングになっている部分の数をカウントするプログラムを作成する。　ファイル名の最後の数値が答えのリングの数になっている。例えばc_1_0.molはリングがなく、c_4_3.molは三つのリングがある。下の図の原子位置にmolファイル中の原子の番号がふられている。

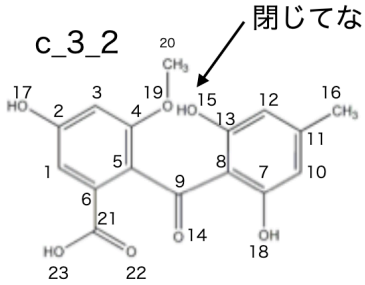
c_1_0



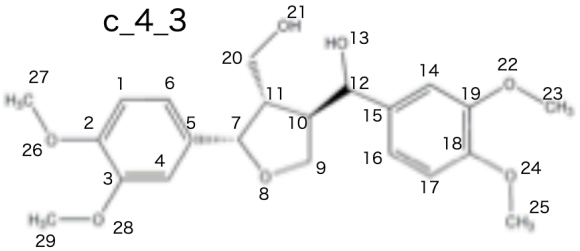
c_2_1



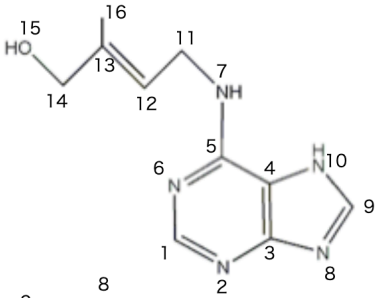
c_3_2



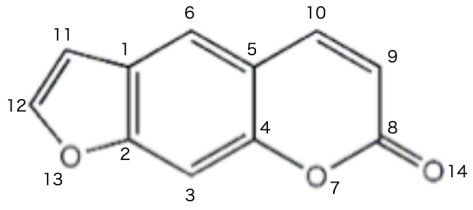
c_4_3



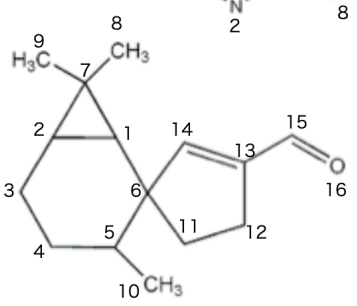
c_5_2



c_6_3

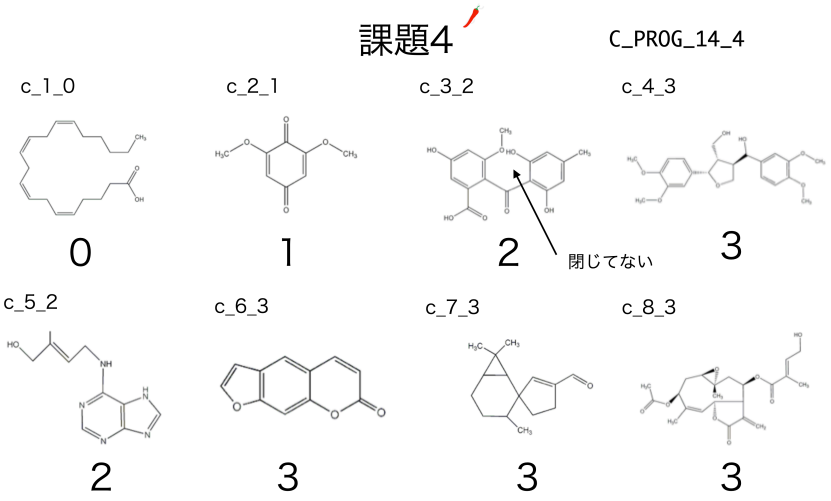


c_7_3



提出課題４　15_4

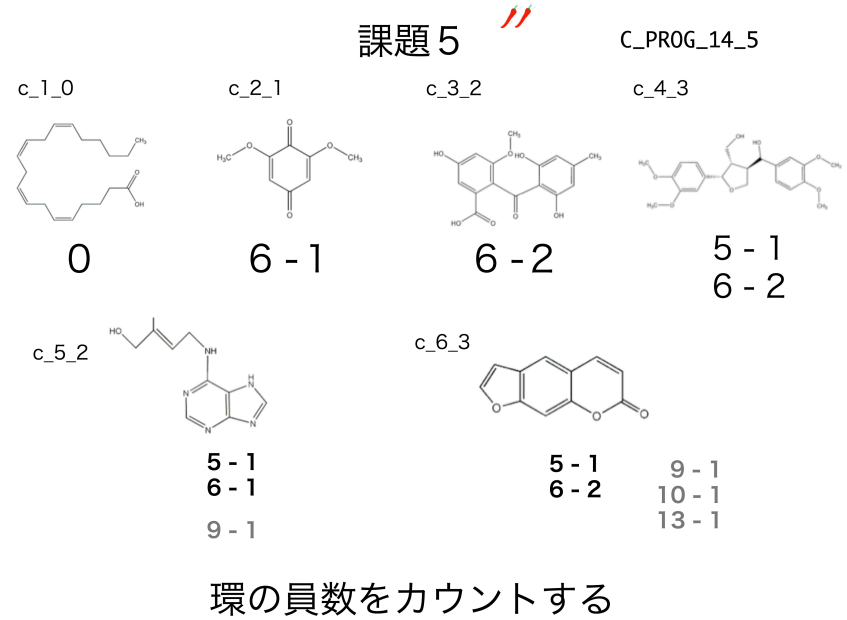
それぞれのファイルについて、リング（環）の数をカウントして表示するプログラムを作成する。



提出課題5　15_5

それぞれのリング（環）について、リングを構成する原子の数をカウントして表示するプログラムを作成する。

c_4_3 では、５個の原子からなる中央の環が一つ、６個の原子からなる両端の環が２つなので、5-1, 6-2といった出力になる。



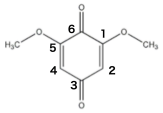
提出課題6　15_6

それぞれのリング（環）について、リングを構成する原子の数をカウントして表示するプログラムを作成する。

C_4_3 では、５個の原子からなる中央の環が一つ、６個の原子からなる両端の環が２つなので、5-1, 6-2といった出力になる。

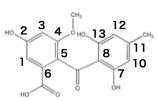
課題６　　C_PROG_14_6

c_2_1



ring1 : 1-2-3-4-5-6

c_3_2



ring1 : 1-2-3-4-5-6

ring2 : 7-8-13-12-11-10

環を構成する原子をリストアップする

1から3の課題についてはソースファイルのみ123NAME15_1.cといった名前に変更してそのまま添付、4から6の課題についてはコンパイルするためのmakefileを作成し、123NAME15_2（NAMEは自分の名前で置き換える） というディレクトリに、makefileとそれぞれのソースファイル「だけ」を入れ、tar, gzip したファイルをメールで提出。

環のカウント

```
int n_ring = 0;
```

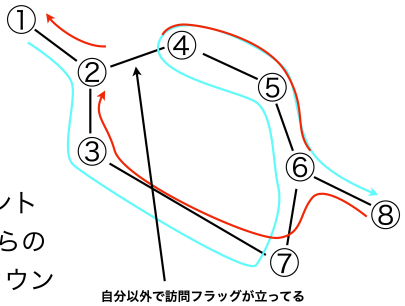
深さ優先探索で

④から②を試す時に

自分以外で訪問フラグが立っている場合

環の数をひとつとカウント

ただし、4から、と2からの様にリングごとに2回カウントするので最後に2で割る




```
n_ring ++;
```

mol 形式のファイルを読み込む

原子間の隣接行列を作る

先週はハードコードしてあった、今回はMOL fileを読んで結合表から作る
２次元配列をmalloc、または原子数最大100x100程度で決めておいてもよい

隣接行列をトレースして環の部分を検出する

先週のプログラム：再帰による深さ優先探索を応用

隣接行列

二次元配列

ポインタの配列をとって malloc

```
int **connect;
```

または、最大の原子数で決めうち

```
connect[MAX_ATOM][MAX_ATOM];
```

MAX_ATOM = 100 程度