Home assignment 3

Oil consists mostly of hydrocarbons. Hydrocarbon molecules consist of carbon and hydrogen atoms. Each carbon atom forms 4 bonds with other atoms, and each hydrogen atom forms one. Instead of a single bond, two carbon atoms can also be connected to each other by a double or triple bond. The length of the hydrocarbons in oil varies greatly; gasoline for example contains mostly hydrocarbons with 5 to 10 carbon atoms[1].

The structure formed by the carbon atoms may be represented as a connected multigraphs, whose vertices correspond to carbon atoms and whose edges correspond to the bonds between carbon atoms. A multigraph is a graph, where an edge may occur more than once. In modeling hydrocarbons the degree of no vertex may exceed four, and double and triple bonds may be represented by having up to three edges between a pair of vertices. Obviously this model ignores, among others, the limitations of three-dimensional geometry.

Let $d(v)$ be the degree of vertex v. When $G = (V, E)$ represents a hydrocarbon molecule with $|V|$ carbon atoms, the molecule will have $\sum_{v \in V} (4 - d(v)) = 4|V| - 2|E|$ hydrogen atoms. Two hydrocarbons are isomers, if they have the same number of hydrogen and carbon atoms.

Assignment

We consider two hydrocarbons isomorphic, if the multigraphs corresponding to them are isomorphic. Design a computer program and use it to generate the nonisomorphic hydrocarbons with n carbon atoms for n as large as possible. Partition the nonisomorphic hydrocarbons into isomer classes and report the number of nonisomorphic hydrocarbons in each isomer class. For each isomer class indicate additionally, how many of the hydrocarbons in it contain no cycle. In evaluating the reports the efficiency of the method is of high importance, but excessive use of computational resources is not necessary. Approximately 2 hours of CPU time on a PC is a suitable order of magnitude. Evaluate the efficiency of your solution.

Hints and an auxiliary program

In solving the problem, computing the canonical form of a multigraph is likely to prove useful. On the home page of the course in the file `kotit3.c` there is a C function for computing the canonical form of a multigraph:

```
void canon(int vertices, char *s, char *canon);
```

[1] http://www.lloydminsterheavyoil.com/refining.htm

`canon` computes the canonical form of a multigraph. The number of vertices in the input multigraph is given as the parameter `vertices`, and `s` is a string that describes the adjacency relation of the multigraph itself. The vertices are numbered `1...vertices` and the string

```
s=b_2,1 b_3,1 b_3,2 b_4,1 b_4,2 b_4,3 b_5,1 ... b_vertices,vertices-1
```

where `b_j,i` is a character that describes the number of arcs between the vertices i and j. The character '0' means 0 edges, the character '1' one edge, etc. The function converts the input string into a graph, computes its canonical form and writes into `canon` the string that corresponds to the canoninal form. The caller must allocate memory for the string `canon`.

The function uses the excellent graph automorphism package `nauty`[2] by Brendan McKay. In the beginning of `kotit3.c` the row

```
#include "nauty22/nauty.h''
```

assumes that `nauty` is placed in the subdirectory `nauty22` of the directory where `kotit3.c` is located. in a directory whe The row

```
#define MAXN 60
```

assumes that the graphs that will be handled with nauty will have at most 60 vertices. The subprogram `canon` converts a multigraph into a graph so that every node and edge of the multigraph is made into an edge of the graph, so the multigraph under consideration may have at most `MAXN` vertices and edges in total—this should suffice at least for hydrocarbons with up to 20 carbons, so it should not be a limiting factor.

The file `kotit3.c` also contains a minimal main program, which calls `canon` once. At least on a Linux based system compilation can be carried out by the command:

```
cc -o kotit3 kotit3.c nauty22/nauty.c nauty22/nautil.c nauty22/naugraph.c
```

Due date

Return the assignment according to the instructions on the course page to Harri Haanpää (Harri.Haanpaa@tkk.fi) no later than on Wednesday April 19.

[2] http://cs.anu.edu.au/~bdm/nauty/