

COMP 3031 Assignment 3

PROLOG Programming

Fall 2020

Due: 5PM on Nov 20 Friday

Instructions

- There are five problems in this assignment. Each problem counts for 2 points.
- This is an individual assignment. You can discuss with others and search online resources, but your submission should be your own code.
- Write your prolog program according to the specification, with the same predicate name and number of arguments as specified. Write all your solutions in a single file named "ass3.pl". In this file, put down your **name, ITSC account, and student ID** as a *comment* (surrounded by "/*" and "*/") on the first line.
- Submit your code through Canvas before the deadline.
- **No** late submissions will be accepted.
- Your submission will be run on a lab 2 machine with the following command in swipl:
?- [ass3].

Please make sure your submission is executable. If it is not, a significant number of points will be deducted.

NOTE: We will perform code similarity checks. In case a submission is confirmed to have code similarity issues, we will deduct partial marks or full marks on a case-by-case basis.

PART I:

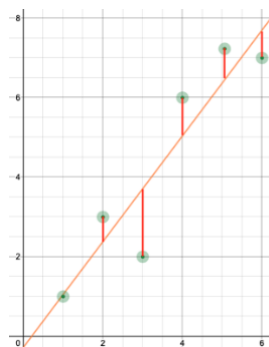
Question 1.

Define a /2 predicate `linear_regression (A, B)` to represent the regression line $y = Ax + B$ that is of the **least-squares** for a set of points (x_i, y_i) . The regression line on a set of points is of the least-squares if the sum of the squares of the vertical deviations from each data point to the line is the **minimal** out of all possible lines.

We will give the set of points in two /1 facts, `xs` and `ys`, where x_i , and y_i are corresponding elements of the list in `xs` and `ys`. Example facts and query:

```
xs([1, 2, 3, 4, 5.06, 6]).
ys([1, 3, 2, 6, 7.23, 7]).

?- linear_regression(A, B).
A = 1.3298931176836508,
B = -0.29625817640294727.
```



You can check the correctness of your program output up to four digits accuracy as well as learn more about linear regression and the least-square method at the following web pages:

<https://ncalculators.com/statistics/linear-regression-calculator.htm>

https://en.wikipedia.org/wiki/Simple_linear_regression

Question 2.

Write a predicate `interesting(X)` to represent **all** interesting vertices `X`, if any, in a given undirected graph. In other words, after issuing the query `interesting(X)`, we can get all interesting vertices one by one by keeping typing `;` until all answers are returned.

A vertex is **interesting** if it satisfies the following three conditions:

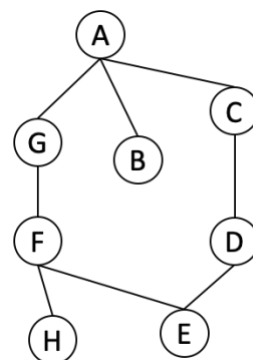
- (1) It is sparse, i.e., its degree is less than or equal to 2.
- (2) It is connected to at least two dense vertices (vertices whose degrees are greater than 2), and its shortest paths to its two closest dense vertices are of the same length.
- (3) Its shortest path length to its two closest dense vertices is the shortest among all other sparse vertices that satisfy condition (2).

Example:

In the graph given by facts `edge`, vertex `A` and `F` are dense, and `B`, `C`, `D`, `E`, `G`, and `H` are sparse. `B`, `C`, `E`, `H` are not interesting vertices, because their shortest paths to their two closest dense vertices `A` and `F` are not of equal length. `G` is an interesting vertex in this graph, of which the shortest path is of length 1 to `A` and `F`. `D` is not interesting, because its shortest path length to `A` and `F` is 2, which is greater than length 1 of `G`.

Example facts and query:

```
edge('A', 'C').
edge('A', 'B').
edge('A', 'G').
edge('C', 'D').
edge('D', 'E').
edge('E', 'F').
edge('F', 'H').
edge('F', 'G').
```



```
?- interesting(X).
X = 'G'.
```

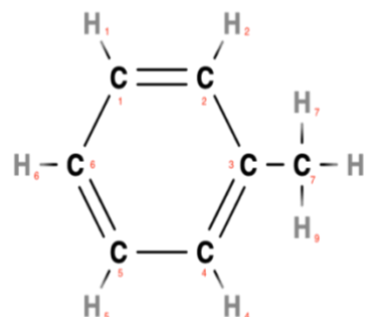
PART II:

We represent the structure of a chemical compound in the following /3 predicate:

`atom_elements(name_of_atom, element_type, bonded_atoms)`

The following example represents the compound *toluene*. It is resulted from bonding a benzene (**C6H6**) and a **CH3** group. The graph contains 15 vertices, including seven **C** atoms and eight **H** atoms, and 15 edges, including eight **C-H** edges and seven **C-C** edges.

```
atom_elements(h1,hydrogen,[c1]).
atom_elements(h2,hydrogen,[c2]).
atom_elements(h4,hydrogen,[c4]).
atom_elements(h5,hydrogen,[c5]).
atom_elements(h6,hydrogen,[c6]).
atom_elements(h7,hydrogen,[c7]).
atom_elements(h8,hydrogen,[c7]).
atom_elements(h9,hydrogen,[c7]).
atom_elements(c1,carbon,[c2,c6,h1]).
atom_elements(c2,carbon,[c1,c3,h2]).
atom_elements(c3,carbon,[c2,c7,c4]).
atom_elements(c4,carbon,[c3,c5,h4]).
atom_elements(c5,carbon,[c4,c6,h5]).
atom_elements(c6,carbon,[c1,c5,h6]).
atom_elements(c7,carbon,[c3,h7,h8,h9]).
```



Question 3.

CH3 is a structure consisting of one **C** atom bound with three **H** atoms. Define a predicate `ch3(X)` in which `X` is a list containing all **C** atoms that belong to the **CH3** structures given in `atom_elements`. The order of the elements in the list is unimportant.

Example query on the given compound *toluene*:

```
?- ch3(X).
X=[c7].
```

Question 4.

Define a predicate `c6ring(X)` in which `X` represents a list containing all six-**C**-atom rings in the given `atom_elements`. Each six-**C**-atom ring is a list containing the six **C** atoms that form the ring. The order of the **C** atoms in the list is unimportant, so each six-**C**-atom ring occurs **exactly once** in the list of rings.

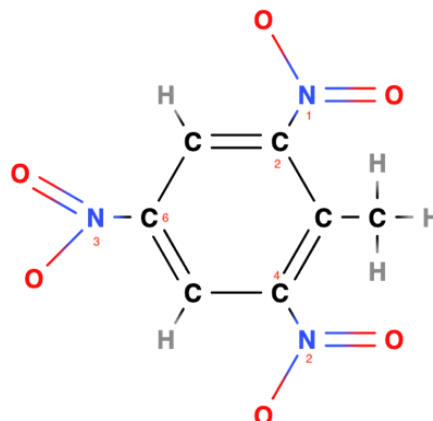
Example query on the given compound *toluene*:

```
?- c6ring(X).
X=[[c1,c2,c3,c4,c5,c6]].
```

Question 5.

The compound *2,4,6-Trinitrotoluene (TNT)*, shown in the following figure, is formed with the *H* atom at *C2*, *C4* and *C6* of the benzene ring replaced with an *NO2* structure. The structure *NO2* consists of one *N* atom bound with two *O* atoms.

Define a predicate `tnt(X)` in which *X* is a list containing all TNT structures. Each TNT structure is represented as a list containing all six *C* atoms on the ring, of which three *C* atoms, at position 2,4,6 on the ring respectively, are each represented as a list containing itself and the *NO2* structure it is bound to. The order of the six *C* atoms in the list is unimportant, and each TNT structure occurs **exactly once** in the list of TNT structures.



The facts that give the example *2,4,6-Trinitrotoluene* figure:

```
atom_elements(h1,hydrogen,[c1]).
atom_elements(n1,nitrogen,[o1, o2, c2]).
atom_elements(o1,oxygen,[n1]).
atom_elements(o2,oxygen,[n1]).
atom_elements(n2,nitrogen,[o3, o4, c4]).
atom_elements(o3,oxygen,[n2]).
atom_elements(o4,oxygen,[n2]).
atom_elements(h5,hydrogen,[c5]).
atom_elements(n3,nitrogen,[o5, o6, c6]).
atom_elements(o5,oxygen,[n3]).
atom_elements(o6,oxygen,[n3]).
atom_elements(h7,hydrogen,[c7]).
atom_elements(h8,hydrogen,[c7]).
atom_elements(h9,hydrogen,[c7]).
atom_elements(c1,carbon,[c2,c6,h1]).
atom_elements(c2,carbon,[c1,c3,n1]).
atom_elements(c3,carbon,[c2,c7,c4]).
atom_elements(c4,carbon,[c3,c5,n2]).
atom_elements(c5,carbon,[c4,c6,h5]).
atom_elements(c6,carbon,[c1,c5,n3]).
atom_elements(c7,carbon,[c3,h7,h8,h9]).
```

Example query on the *2,4,6-Trinitrotoluene*:

```
?- tnt(X).
X = [[c1, [c2, n1, o1, o2], c3, [c4, n2, o3, o4], c5,
[c6, n3, o5, o6]]] .
```