MATH6010-M02 Assignment02

Jia Mian Tan 120033990001

Find the dominating set using greedy approach

|  |  |
| --- | --- |
| **Algorithm** findDominatingSetGreedy | |
| **Input:** A graph with vertices and edges | |
| **Output:** A set , the dominating set of vertices | |
|  | adjacency matrix; [] |
|  | array used to tell if the vertex is already covered: 1 if covered, 0 otherwise. Initialized as 0 for all vertices. |
|  | array used to store the number of uncovered neighbors for each vertex. Initialized as the number of neighbors + 1(itself) |
|  | **while** sum() **is not**  **do** *(meaning there is still vertex(vertices) uncovered)* |
|  | the vertex with largest number of uncovered neighbors |
|  | add to tail of |
|  |  |
|  | **if**  has never been covered |
|  | *(minus itself)* |
|  | **for**  from **to** |
|  | **if**  is an uncovered neighbor of |
|  | remove the edge between and |
|  | **if**  has never been covered |
|  |  |
|  | **if**  has never been covered |
|  |  |
|  | **for**  from **to** |
|  | **if** there is still edge between and |
|  |  |
|  | **if**  has been covered before *(so now both and are covered)* |
|  | remove the edge between and |
|  | mark as covered |
|  | mark as covered |
|  | **return** |

p/s: the vertices which have been covered but not yet in the dominating set are still eligible to be picked into the dominating set

Driver code

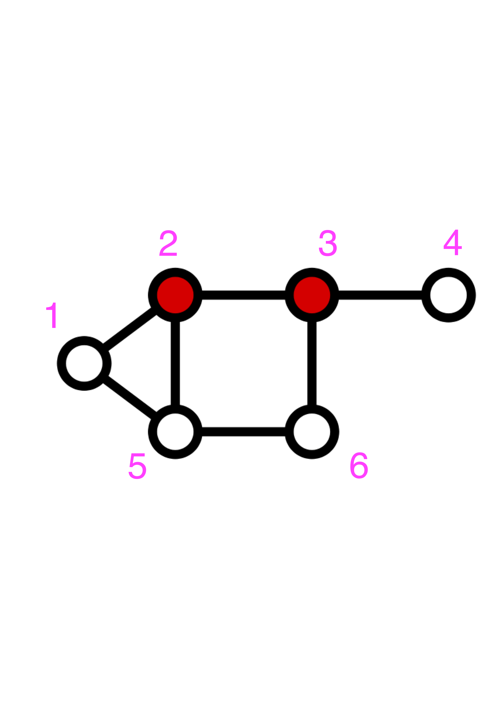
Sample 1: randomly generated 100x100 adjacency matrix, the number of minimum neighbor for each vertex is 3.

n = 100; = 3

Sample 2:

n = 6; = 1

Greedy Approach should be 2 🡪 3;

output in Python (index from 0): 1 🡪 2

Sample 3:

n = 10; = 3

Greedy Approach should be 1 🡪 2 🡪 9;

output in Python (index from 0): 0 🡪 1 🡪 8

