Data Structures and Algorithms (INFO-F413)
Assignment 1: Karger's Algorithm

Charlotte Nachtegael

October 4, 2016

1 Source code of the program in Python 3

```
from random import choice
from copy import deepcopy
from math import log


def make_graph(file):
    """
    Create a dictionary from the data file
    :param file: string of the name of the data file
    :return: dictionary with vertex as key and a set of the vertices they
        are connected to as values
    """
    dico = {}
    with open(file, 'r') as data:
        for line in data:
            line = line.strip().split(' ')
            dico[line[0]] = line[1:]
    return dico


def contract(vertex1, vertex2, graph):
    """
    Contraction of an edge
    :param vertex1: string of the vertex 1
    :param vertex2: string of the vertex 2
    :param graph: dictionary with the vertices as keys and the list of the
        vertices they are connected to as values
    :return: dictionary of the graph with the vertices 1 and 2 combined
    """
    new_edge_vertex = graph[vertex1] + graph[vertex2]

    # remove the edge between vertex 1 and 2
    while vertex1 in new_edge_vertex:
        new_edge_vertex.remove(vertex1)
    while vertex2 in new_edge_vertex:
        new_edge_vertex.remove(vertex2)
```

```
35
36      # remove the vertex 2, update the content to only have the vertex 1,
            combination of vertex 1 and 2
37      del graph[vertex2]
38      graph[vertex1] = new_edge_vertex
39      for vertex in graph:
40          edges = graph[vertex]
41          while vertex2 in edges:
42              edges.remove(vertex2)
43              edges.append(vertex1)
44
45      return graph
46
47
48  def minimum_cut(graph):
49      """
50      Look for the minimum cutset by contracting randomly two vertices of the
            graph until only two vertices are left
51      :param graph: dictionary with vertices as keys and list of neighbours
            as values
52      :return: integer of number of edges to cut to disconnect the graph
53      """
54      while len(graph) > 2:
55          vertices = list(graph.keys())
56          vertex1 = choice(vertices)
57          vertex2 = choice(graph[vertex1])
58          graph = contract(vertex1, vertex2, graph)
59
60      return len(graph.popitem()[1])
61
62
63  def summary(results, n, runs):
64      """
65      Print the summary of the results
66      :param results: list of the results obtained with the different runs of
            the karger's algorithms
67      :param n: size of the graph
68      :param runs: number of runs done
69      """
70      final = min(results)
71      print('The minimal cut is of %s edges' % final)
72      right = 0
73      for res in results:
74          if res == final:
75              right += 1
76
77      observed_success = right/runs
78      lower_bound = 2/(n*(n-1))
79
80      if observed_success >= lower_bound:
81          print('The lower bound of the probability of success is respected
                !\n'
82                'Observed success frequency = %s\n'
```

```
83              'Lower bound probability of success = %s\n' % (
                    observed_success, lower_bound))
84
85
86  def main():
87      """
88      Run several times the Karger's algorithm for a graph
89      """
90      file = str(input('Name of the file with the vertex and their neighbours
            : '))
91      graph = make_graph(file)
92      n = len(graph)
93      res = []
94      run = n*(n-1)*log(n)
95      i = 0
96
97      while i < run:
98          working = deepcopy(graph)
99          res.append(minimum_cut(working))
100         i += 1
101
102     summary(res, n, i)
103
104 if __name__ == "__main__":
105     main()
```

2 Observation of the results obtained

The program was run with different graphs of different size and different maximal and minimal degrees of the vertices of the graph.

|V|	Minimal degree	Maximal degree	Lower bound	Observed success
8	3	4	0.03571428571428571	0.3162393162393162
8	3	6	0.03571428571428571	0.3504273504273504
8	2	4	0.03571428571428571	0.41025641025641024
8	2	6	0.03571428571428571	0.26495726495726496
10	3	4	0.022222222222222223	0.4182692307692308
10	3	6	0.022222222222222223	0.23076923076923078
10	2	4	0.022222222222222223	0.27403846153846156
10	2	6	0.022222222222222223	0.09134615384615384
12	3	4	0.015151515151515152	0.3009118541033435

We observed that the observed frequency of success is always higher than the lower bound of probability of success.

It was shown that increasing the number of the vertices would decreased the lower

bound of probability of success, as its value is inversely proportional to the number of vertices. This is logical because if you increase the number of vertices, the number of edges is also increased, which means that the probability of picking a edge belonging to the set of edges to resolve the minimal cut problem is decreasing. However, we did not observe a consistent increase of the observed frequency of success with the increase of the number of vertices.

We tried to see if changing the minimal and maximal degree of the vertices in the graph would have a significant effect, but no consistent results were found.