104291 - Combinatorial Algorithms

Spring 2025

Assignment 1 (v1.1)

General Considerations

The due date for this assignment is April 25 at 23:59 (There will be a 2-hour time window after the deadline; submissions delivered during this time frame will be accepted but penalized. After that, submissions will not be accepted.)

This assignment is to be done in a group of 2 people. It consists of 2 problems to implement in python. It will be graded with a score in the range of 0-100 points.

Each group must turn in:

The source code of the solution in python: Code should be well structured and documented. It should run on a basic Unix system, and it should be easy to test. Test cases as described in each exercise should be provided.

A report on the assignment: Mainly, the report should describe how the provided code works, explaining the underlying algorithm and the thought process that led to it. This description should not be a mere listing of the code. The test cases provided should be used as examples of how the code arrives at the correct answer. The report should not be longer than 6 pages.

The source code for both exercises and the report should include the names and IDs of the members of the group, and all documents should be delivered as a single .zip file.

The presentation of the assignment is important, and lack of clarity, examples, or proper identification of the students will be penalized.

Also, if we consider it necessary, we may ask some groups to further explain their assignment personally in the teacher's office. In that case, students will be informed so an appropriate date and time for this can be arranged. Each member of the group is expected to be able to explain the whole assignment. Otherwise, they risk failing the assignment (0 points).

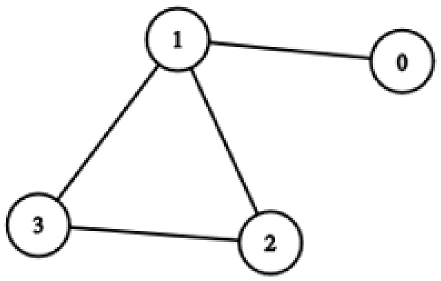
We remark that we expect every assignment to be coded and have its report written by the students that turn it in, and not by someone else (human or otherwise). Any suspicion of plagiarism will be elevated to discipline to be dealt with.

You ARE allowed to use code from the book or from the uploads on class, but they must be PROPERLY CITED as a comment in the code and as a listed source in the report. Failing to cite the class material as source will be penalized. Failing to cite other sources, especially for source code, *will be taken as plagiarism*.

You may use the *math* package to solve these exercises. The use of the *itertools* package is strictly forbidden. If there is any other package you think you need to use, ask first, the use of a package without previous consent will be penalized.

Exercise 1 (50 points)

A *paw* is a 4 vertex graph that consists of a triangle and a vertex which is adjacent to just one of the vertices of that triangle, like in the following figure:



Write a python program that takes an undirected graph from standard input, in the following format:

- A line with two numbers, n, m separated by a space
- m lines with two numbers v w, separated by a space, both **between $\mathbf{0}$ and $\boldsymbol{n-1}$**

The first line represents the number of vertices and edges of the graph. The following m lines describe its edges.

The program should print a list of all sets of 4 vertices from the graph that form an **induced paw**, that is, such that they form a paw and there are no other edges between them. The list should have one set by line (to represent the subset you can use the set type or a list in ascending order), and they should be ordered lexicographically (note that a solution that generates solutions in an arbitrary order and then uses python's **sort** function will be allowed but with a lower mark – however, using an alternate lexicographic generation method that the one presented in class, **provided it is correctly justified in the report**, is valid). If the provided graph has less than 4 vertices or doesn't have an induced paw, the program should print that it found no paws in the graph.

For instance, for input:

```
7 10

1 2

1 3

2 3

3 4

2 4

0 1

0 4

6 4

5 3

2 6
```

The program should print:

```
[0,1,2,3]

[0,2,3,4]

[0,2,4,6]

[1,2,3,5]

[1,2,3,6]

[1,2,4,6]

[2,3,4,5]
```

While for input:

```
6 12

0 1

0 3

1 3

2 1

2 4

4 1

5 3

5 4

3 4

0 2

0 5

2 5
```

The program should print that it found no paws in the graph.

The program should be able to calculate these examples in less than one minute.

Provide 2 further examples of graphs of 10 vertices, one should have at least 4 paws and the other should be connected and have no paws (and explain in the report how your program solves them). The example graph should not be trivial (i.e., be complete graphs or have no edges at all).

Exercise 2 (50 points)

Write a python program that takes the following from standard input:

- A line with two numbers, n, m, separated by a space
- m lines with two numbers a b, separated by a space, both **between 1 and n**

We want to find a permutation of the numbers from 1 to n such that for each of the m lines a b in the input, the number a is before b in the permutation. That is, if we have

```
10 8
```

```
2 3

4 5

1 2

1 4

2 4

7 8

4 7

3 6
```

It means we are looking for a permutation of the numbers 1 to 10 in which 2 is before 3, 4 is before 4, 1 is before 2, 1 is before 4, 2 is before 4, 7 is before 8, 4 is before 7 and 3 is before 6.

The program should print the rank in the **Trotter-Johnson ordering** of a permutation that satisfies all the conditions, if it exists, or a message indicating that they cannot be satisfied, if it doesn't (If more than one permutation exists, you can provide the rank of any permutation that does). Provide two examples of permutations for numbers from 1 to 8, one in which the restrictions are possible and one in which they are not.

For instance, for the example above a possible number to print is `244529` (Although your program may print another valid ranking number).

For the following example:

```
5 5

1 2

2 3

5 3

3 4

4 1
```

The program should print that it is impossible to satisfy.

The program should be able to calculate these examples in less than one minute.

Additionally, provide 2 further examples of permutations for numbers from 1 to at least 8, one should have at least 6 restrictions and be possible to satisfy, and the other should be impossible to satisfy (Please avoid just considering all possible restrictions toghether). Explain in the report how your program reaches each solution.

Bibliography

[1] *Combinatorial Algorithms: Generation, Enumeration, and Search,* by Donald L. Kreher and Douglas R. Stinson.

[2] *Introduction to Algorithms,* by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.