Discrete Optimization Assignment:
Set Cover

1 Preamble

This is an open source assignment and will not count toward your final grade in the course. This assignment looks-and-feels like all of the other assignments in the class, but you may ignore the typical collaboration policy guidelines and share your source code with your classmates.

To foster code sharing on this assignment, we provide a github repository including various solutions to this problem (`http://www.github.com/discreteoptimization`), which you are free to use. We encourage you to fork this repository and share your algorithms for the set cover problem.

2 Problem Statement

In this assignment you will design an algorithm to solve the *Set Cover Problem*. You are given a number of regions and need to decide where to place a set of fire-stations to assure that every region is covered in case of emergency. For each possible fire-station, we know the building costs, and the set of regions it covers. The goal is to select those fire-stations that cover all regions with minimal building costs. Figure 1 illustrates an example, showing a map of five regions and the region coverage and costs for four fire-stations, given in different colours. In the solution, three fire-stations have been selected that cover all regions and whose building costs amount to 24.

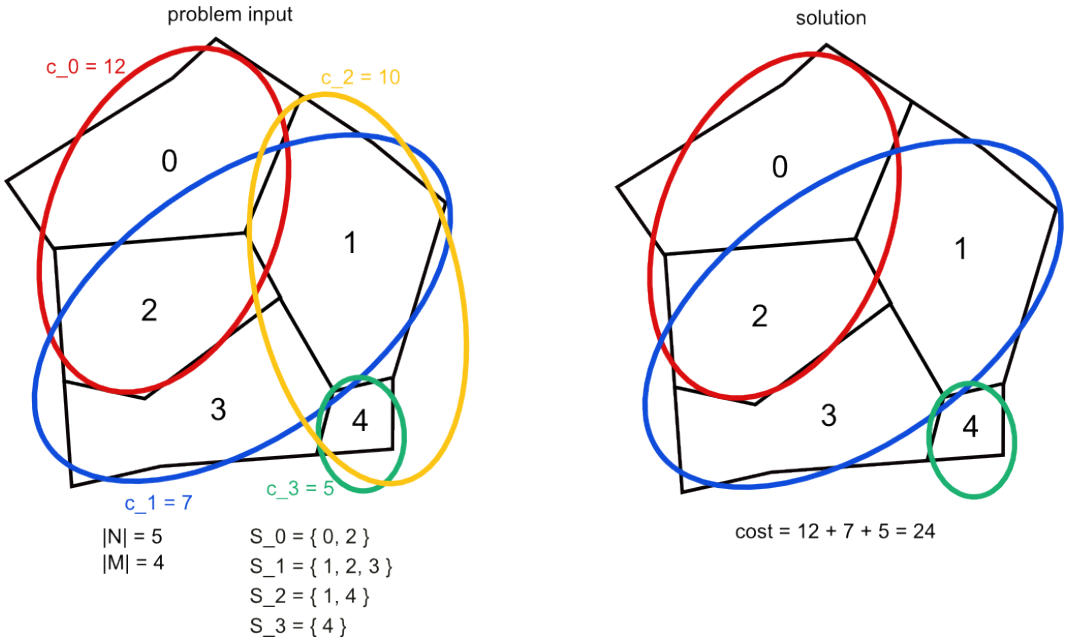

Figure 1: A Set Cover Example

3 Assignment

Write an algorithm to solve the set cover problem. We will now generalise the problem and therefore call the regions '*items*' and the fire-stations '*sets*'. The problem is mathematically formulated in the following way: We are given a number of items (regions) $N = \{0 \dots n-1\}$ and sets (possible fire-stations) $M = \{0 \dots m-1\}$. For each set $i \in M$, we know the cost c_i, and the set of items $S_i \subseteq N$ that it covers. For instance, in Fig. 1, the first possible fire-station has cost $c_0 = 12$ and covers regions $S_0 = \{0, 2\}$. We use variables $x_i \in \{0, 1\}$ to denote if we select set i. Our goal is to find a selection of sets with minimal cost that cover all items. The set cover problem is formalized as the following optimization problem,

$$\begin{aligned} \text{minimize:} \quad & \sum_{i \in M} c_i x_i \\ \text{subject to:} \quad & \\ & \sum_{i \in M} (j \in S_i) x_i \geq 1 \quad (j \in N) \\ & x_i \in \{0, 1\} \end{aligned}$$

where the constraint ensures that each item is covered at least once: for each item $j \in N$, it states that the sum over all sets $i \in M$ that contain item j, has to be greater or equal to 1.

4 Data Format Specification

The input consists of $|M| + 1$ lines, where $|M|$ is the number of sets. The first line contains 2 numbers: the number of items, $|N|$, and the number of sets, $|M|$. It is followed by $|M|$ lines, where each line represents the data for a set i: first comes the cost c_i, followed by the items $S_i = \{s_{i0}, s_{i1}, \dots\}$ that set i covers.

Input Format

```
|N| |M|
c_0 s_0_0 s_0_1 ...
c_1 s_1_0 s_1_1 ...
c_2 s_2_0 s_2_1 ...
...
c_|M|-1 s_(|M|-1)_0  ...
```

The output has two lines. The first line contains two values *obj* and *opt*. *obj* is the cost of the selected sets (i.e. the objective value). *opt* should be 1 if your algorithm proved optimality and 0 otherwise. The next line is a list of $|M|$ values in $\{0, 1\}$, one for each of the x_i variables that represent which sets have been selected. This line encodes the solution. Output Format

```
obj opt
x_0 x_1 ... x_|M|-1
```

Examples (based on Figure 1)

Input Example

```
5 4
12.0 0 2
7.0 1 2 3
10.0 1 4
5.0 4
```

Output Example

```
24.0 0
1 1 0 1
```

The output example illustrates the problem input and solution from Figure 1.

5 Instructions

Edit `solver.py` and modify the `solve_it(input_data)` function to solve the optimization problem described above. The function argument, `input_data`, contains the problem data in the format described above. The return value of `solve_it` is a solution to the problem in the output format described above. Your `solve_it` implementation can be tested with the command,

```
python ./solver.py ./data/<inputFileName>
```

You should limit the `solve_it` method to terminate within 5 hours, otherwise the submission will not be eligible for full credit. You may choose to implement your solver directly in python or modify the `solve_it` function to call an external application.

Resources You will find several set covering problem instances in the `data` directory provided with the handout.

Handin Run `submit.py` with the command, `python ./submit.py`. Follow the instructions to apply your `solve_it` method on the various assignment parts. You can submit multiple times and your grade will be the best of all submissions. However, it may take several minutes before your assignment is graded; please be patient. You can track the status of your submission on the *feedback* section of the assignment website.

Grading Infeasible solutions (i.e. those that do not conform to the output format or violate problem constraints) will receive 0 points. Feasible solutions will receive at least 3 points. Feasible solutions passing a low quality bar will receive at least 7 points and solutions meeting a high quality bar will receive all 10 points. The grading feedback indicates how much your solution must improve to receive a higher grade.

Collaboration Rules In open source assignments we encourage collaboration, exchange of ideas, and code! However, please refrain from the following:

1. Posting or sharing problem solutions.

The goal here is to share your algorithms, not just a list of high quality solutions.

Warnings

1. It is recommended you do not modify the `data` directory. Modifying the files in the data directory risks making your assignment submissions incorrect.
2. You cannot rename the `solver.py` file or the `solve_it()` method.
3. Be careful when using global variables in your implementation. The `solve_it()` method will be run repeatedly and it is your job to clear the global data between runs.
4. `solver.py` must remain in the same directory as `submit.py`.

6 Technical Requirements

You will need to have python 2.7.x installed on your system (installation instructions, `http://www.python.org/downloads/`).