IT 328, Introduction to the Theory of Computation

Programming Assignment 1 (Team work)

3SAT $\leq_P$ Vertex-Cover & Clique $\leq_P$ Vertex-Cover

Due date: Oct. 9, 2022, Sunday, 11:55 PM 50 points (45 on programs, 5 on report)

Discuss with your classmates and find two partners to form a team of 3 students and decide who will be the corresponding student for this programming assignment. The corresponding student will be in charge of submitting the report and preparing the programs on his/her Unix directory, and answering my questions or comments, if any. After the team is formed, please use the ReggieNet to joint the team for my records. On ReggieNet, under Site Info, there is a selection labeled as *"Groups you can join"*. One of the team member should select an empty group first for the other two to join.

Every method developed by the team can be shared within the team, either in an independent classes or repeated codes in different programs (e.g., codes for reading and parsing the text files that contain graph descriptions can be reused). Let the team members be Students A, B, and C. The major programming responsibility is divided as follows.

- **Student A** should write a method that takes an undirected graph $G : (V, E)$ (in whatever presentation the team decided) and an integer $1 \leq k \leq |V|$ as inputs of the method and returns a k-vertex cover, if such k-vertex cover exists in G.
- **Student B** should write a method that takes an 3CNF φ (in whatever presentation the team decided) as the input and returns an assignment that makes φ true, if such assignment exists. However, Student B should not solve the problem directly. Instead, he/she should reduce the problem in polynomial time to a corresponding k-vertex cover problem for Student A's method to solve and use the results (some k-vertex cover, if exists) to answer the original 3CNF problem. In other words, Student B should implement the Cook-reduction that reduces 3SAT to Vertex-Cover.
- **Student C** should write a method that takes an undirected graph $G : (V, E)$ (in whatever presentation the team decided) and an integer $1 \leq n \leq |V|$ as inputs of the method and return a n-clique, if such n-clique exists in G. As Student B, this method should not solve the problem directly. Instead, he/she should reduce the problem in polynomial time to a corresponding k-vertex cover problem for Student A's method to solve and use the results (some k-vertex cover, if exists) to answer the original n-clique problem. In other words, Student C should implement the Cook-reduction that reduces Clique to Vertex-Cover.

It is clear that Student C's task is relatively easy. It is the team's agreement to distribute the work load fairly. for example, Student C may take more responsibility to implement the program interface such as reading and parsing the text files that contain graph descriptions, or taking care of the output format.

File formats: There are two text files needed for this project, one contains undirected graphs and the other contains 3CNF described as follows.

- **Graph file:** `graphs2022.txt`

  In this text file, there are a sequence of graphs. A graph $G : (V, E)$ with $|V| = n$ is stored in $(n + 1)$ lines. For each graph, the first line contains a number n to indicate the size of V. The following n lines represent the *adjacency matrix* m of G (i.e., m is an $n \times n$ two dimensional array). If the entry $m[i][j] = 1$, that means $(i, j) \in E$; if $m[i][j] = 0$ means $(i, j) \notin E$.

  One can see that all matrices in this file are symmetric, i.e., for every $0 \leq i, j < n$ we have $m[i][j] = m[j][i]$, which mean undirected. By convention of undirected graphs, if $m[i][j] = m[j][i] = 1$, we count as one edge in E. Also note that, the diagonal of the matrix is set to 1, i.e. $m[i][i] = 1$ as we consider every vertex links to itself but we do not count it as an edges, thus (v, v) does not contribute to the edge count, $|E|$. The file is ended by an empty graph, i.e., $n = 0$.

- **3CNF file:** `cnfs2022.txt`

  Each line in this file is a 3CNF, where positive numbers $1, 2, \ldots$ representing boolean variables a_1, a_2, $\ldots$ and negative numbers $-1, -2, \ldots$ representing negation of boolean variables, $\neg a_1, \neg a_2, \ldots$. These numbers are grouped 3 by 3, and each group is a clause. For example, a line

  ```
  1 3 -2 3 1 4 -4 5 1
  ```

  represents 3-CNF:

  $$(a_1 \vee a_3 \vee \neg a_2) \wedge (a_3 \vee a_1 \vee a_4) \wedge (\neg a_4 \vee a_5 \vee a_1).$$

  Hint: you can use the largest index to indicate the number of variables. In the example above, it is a_5, and therefore the number of variables is 5. It should not be a problem if some variables with smaller indices are missing from the formula.

What to do

1. Let $\sim$ denote your home directory in your Unix account. Make directory `~/IT328/npc/`. All files related to this assignment should be prepared in this directory.

2. From my `/home/ad.ilstu.edu/cli2/Public/IT328/`, copy `graphs2022.txt` and `cnfs2022.txt` to your `~/IT328/npc/`.

3. **Program:** Each team will write three Java programs: `findVCover.java`, `find3SAT.java`, and `findClique.java`. Students are free to create other programs or classes if that helps but the three main programs need to be named as required above (note that Unix is case sensitive) and take arguments from the command line as shown in the following.

   - `findVCover.java`

     This program finds and prints a minimum vertex cover of each graph in the input file. For example, on the Unix command line,

     ```
     java findVCover graphs2022.txt
     ```

     should print the results in the following format.

```
* A Minimum Vertex Cover of every graph in graphs2022.txt *
   (|V|,|E|) (size, ms used) Vertex Cover
G1 ( 5, 8) (size=3 ms=0) {0,3,4}
G2 ( 5, 7) (size=2 ms=0) {1,4}
G3 (10, 21) (size=6 ms=0) {0,1,5,6,7,9}
.............
.............
G50 (125,3882) (size=115 ms=435) {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,18,19,20,21,2
***
```

For `G1`, `(5, 8)` indicates the number of vertices (`|V|`=5) and the number of edges (`|E|`=8) and `(size=3, 0 ms) {0,3,4})` shows the program finds a 3-vertex cover (vertices 0, 3, and 4), which is the minimum one in graph `G1`.

- `find3SAT.java`

  This program finds and prints a truth assignment, if exists, for each 3CNF in the input file. For example, on the Unix command line,

  ```
  java find3SAT cnfs2022.txt
  ```

  should print the results in the following format.

  ```
  * Solve 3CNF in cnfs2022.txt:  (reduced to K-Vertex Cover) *
  X means either T or F

  .............

  3CNF No.3:[n=4 k=3] (0 ms) Solution:[1:F 2:T 3:X 4:F]
  ( 2|-1|-1)∧(-3|-2|-4)∧( 4|-3|-1) ==>
  ( T| T| T)∧( X| F| T)∧( F| X| T)
  .............

  3CNF No.7:[n=3 k=6] (5 ms) No solution!  Random:[1:T 2:F 3:F]
  ( 1|-2| 3)∧(-2| 2| 1)∧( 2| 2| 1)∧(-3|-3|-1)∧(-2|-3|-3)∧(-1|-1| 3) ==>
  ( T| T| F)∧( T| F| T)∧( F| F| T)∧( T| T| F)∧( T| T| T)∧( F| F| F)

  3CNF No.8:[n=3 k=7] (0 ms) Solution:[1:T 2:T 3:F]
  ( 2| 1|-1)∧( 3| 3| 2)∧( 1| 2|-2)∧( 2|-3| 1)∧( 3| 1| 3)∧(-2| 1|-3)∧(-2|-1|-3) ==>
  ( T| T| F)∧( F| F| T)∧( T| T| F)∧( T| T| T)∧( F| T| F)∧( F| T| T)∧( F| F| T)

  .............
  ***
  ```

  The output indicates that 3CNF No. 3 in the file has 4 boolean variables and 4 clauses, and it is satisfiable witnessed by a solution `[1:F 2:T 3:X 4:F]`, which means $a_1 = false$, $a_2 = true$, $a_4 = false$, and a_3 can be either *true* or *false*. The next line presents the 3CNF with parenthesis to group clauses, where `|` means logical **or** and ∧ logical **and**, followed by `==>` and in the next line, its an evaluation of each literal using the assignment shown above. For another example, 3CNF No. 7, it is *unsatisfiable*, hence there is no solution to satisfy it. In this no solution case, a random assignment must be assigned, for example `[1:T 2:F 3:F]`, which means $a_1 = true, a_2 = a_3 = false$. However, this random assignment will not satisfy the 3CNF as witnessed by the last clause, which gives `( F| F| F)`.

- `findClique.java`

  This program finds and prints a maximum clique of each graph in the input file. For example, on the Unix command line,

  ```
  java findClique graphs2022.txt
  ```

  should print the results in the following format.

  ```
  * Max Cliques in graphs in graphs2022.txt (reduced to K-Vertex Cover) *
     (|V|,|E|) (size, ms used) Cliques
  G1 ( 5, 8) (size=4 ms=0) {0,1,3,4}
  G2 ( 5, 7) (size=3 ms=0) {0,1,4}
  G3 (10, 21) (size=5 ms=0) {1,4,6,7,9}
  ............
  G50 (125,3882) (size=9 ms=289) {0,6,18,24,68,85,90,91,110}
  ***
  ```

  For `G1`, there is a 4-clique, which is the maximum one in graph `G1`.

4. **Final Step on Unix Account.** After the team finishes the project, or has decided that what you have is the final version for me to grade, collect or transfer all needed program in the corresponding student's Unix account and select a secret name, say "`peekapoo`" as an example (you should chose your own, but no more than 8 characters and no space and special symbols.), and that will be the secret directory, and should not give to anyone except the team members and the grader (that's the instructor). Run the following bash script program from corresponding student's Unix account:

   ```
   bash /home/ad.ilstu.edu/cli2/Public/IT328/submit328.sh peekapoo
   ```

   Note that your programs have to be in the required directory ~`/IT328/npc/` as described in step 1 in order to let this script program correctly copy your programs into the secret directory, where your programs will be tested. If you modify your programs, you have to run `submit328.sh` again. Don't compile and run your programs from the directory where `submit328.sh` copies to. Also, unix is case sensitive. Name you directory and program files correctly.

5. **Report.** Each team has to write up a report and submit a copy of the report through corresponding student's ReggieNet. The report should include:

   - A cover page that contains assignment number, team memebers' names and indicate who is the corresponding student, ULID (not student ID) and **the name of the secret directory**.
   - A brief summary of the methods, algorithms and data structures, and the difficulties, if any, the project has faced and how to solve them.
   - The direct output of your programs on all required inputs. The program code is not required.

   The score is based on the correctness and documentation of your program, 50 points (45 on programs, 5 on report).