# Homework #4

Due Time: 2020/01/02 (Thur.) 14:20
Contact TAs: `ada-ta@csie.ntu.edu.tw`

## Instructions and Announcements

- There are **four programming problems** and **two hand-written problems**.
- **Programming.** The judge system is located at `https://ada-judge.csie.ntu.edu.tw`. Please login and submit your code for the programming problems (i.e., those containing "Programming" in the problem title) by the deadline. NO LATE SUBMISSION IS ALLOWED.
- **Hand-written.** For other problems (also known as the "hand-written problems"), you **MUST** turn in **a printed/written version** of your answers to **the submission box at your classroom**. You can also upload your homework to the NTU COOL system as a backup; however, it will be marked **only when** you have turned in the printed/written answer but it is lost during the grading process.
  NO LATE SUBMISSION IS ALLOWED.
- **Collaboration policy.** Discussions with others are strongly encouraged. However, you should write down your solutions **in your own words**. In addition, for **each and every** problem you have to specify the references (e.g., the Internet URL you consulted with or the people you discussed with) on the first page or comment in code of your solution to that problem. You may get zero point due to the lack of references.

# Problem A - Loopy Tippy (Programming) (15 points)

## Problem Description

**Background** As you may have learned in class, there is no known algorithm that is able to solve NP-complete problems in polynomial time (yet). However, these problems still happen a lot in real life[1]. One method of solving these problems is to encode them into SAT, i.e., boolean satisfiability problems, and apply existing solvers[2]. There are several benefits to this approach, such as being able to take advantage of existing, highly optimized solvers, and having more flexibility than domain-specific solvers.

### Story

> I have coined a truly remarkable story about WillyPillow and Chino which this margin is too small to contain.

### Problem Formulation

You are given a Slitherlink[3] puzzle and *Cryptominisat5*, a CNF-SAT solver (for which the details are given below). Please solve the puzzle by translating it to CNF-SAT and applying the solver.

In essence, the rules of the puzzle are as follows: you are given a rectangular grid, with each square possibly containing a number. The objective is to color a subset of the edges so that the following conditions are satisfied:

1. If a square contains a number, the number of colored edges around it needs to be equal to the given number.
2. The set of colored edges forms exactly one simple loop.

You can try the puzzle out here[4] or (a harder version) here[5].

### Input Format

The first line contains two integers $r$ and $c$, denoting the numbers of rows and columns of the board, respectively. $r$ lines follow, where each line contains $c$ characters representing the board. A character, "`.`", corresponds to a square without a number. Note that the given puzzle is guaranteed to have a unique solution.

In addition, since it may be hard to predict the needed time for solving your SAT encoding, it is guaranteed that the puzzles in the subtasks 1 through 5 are sampled from this archive[6]. However, due to a large number of the released test cases, **the upload quota is limited to 5 times a day** to ease the burden on the judge server.

**EDIT:** It is guaranteed that there is at least one non-zero number on the board.

[1] E.g., *lock-chart solving* and *dependency management*.

[2] Usually based on conflict-driven clause learning (https://en.wikipedia.org/wiki/Conflict-driven_clause_learning), a fascinating algorithm worth reading about.

[3] `https://en.wikipedia.org/wiki/Slitherlink`

[4] `https://www.chiark.greenend.org.uk/~sgtatham/puzzles/js/loopy.html`

[5] `https://kwontomloop.com/`

[6] `https://www.csie.ntu.edu.tw/~b07902134/ada-19hw4p1-public.zip`

**Subtask 1, 6 (10% each)**

- $(r, c) = (3, 3)$

**Subtask 2, 7 (10% each)**

- $(r, c) = (7, 7)$

**Subtask 3, 8 (10% each)**

- $(r, c) = (10, 10)$

**Subtask 4, 9 (10% each)**

- $(r, c) = (15, 15)$

**Subtask 5, 10 (10% each)**

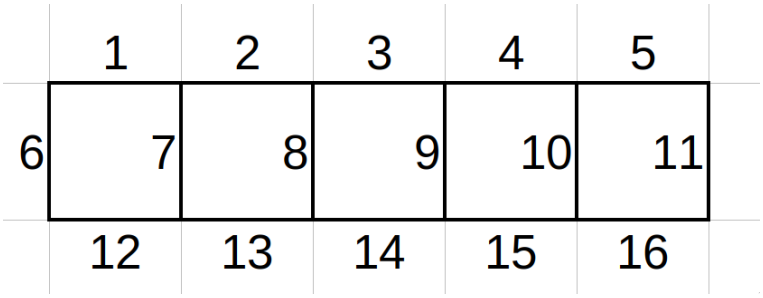
- $(r, c) = (20, 20)$

**Bonus (0%)**

- $r, c \leq 128$

### Output Format

Please print a binary string with $(2rc + r + c)$ characters, denoting the answer to the puzzle. The following order shows an example of $r = 1, c = 5$. The $i$-th character in your output indicates whether the edge at location $i$ is filled. Specifically, a character "`1`" denotes that the edge is filled and `0` otherwise.




### Sample Input

```
7 7
.....3.
22.3..2
32....2
.3..2.3
3.....2
.3132.2
3.....3
```

### Sample Output

```
11111011000011111111010000010
01111101110000100110101101100 11
10011100101100101101101000000
10111011010011001010111110011
```

(Line breaks added only for clarity; please output the string in one single line.)

The human-readable solution to this board is as follows:

```
+-+-+-+-+-+ +-+
|. . . . .|3|.|
+-+-+-+-+ +-+ +
 2 2 . 3|. . 2|
+-+-+-+-+ +-+-+
|3 2 . . .|. 2
+-+-+ +-+ +-+-+
 . 3|.|.|2 . 3|
+-+-+ + +-+ +-+
|3 . .|. .|.|2
+-+-+ +-+ + + +
 . 3|1 3|2|.|2
+-+-+ +-+ + +-+
|3 . .|. .|. 3|
+-+-+-+ + +-+-+
```

### Environment

*Cryptominisat5*[7] is installed on the judge server. Instructions about how to use it can be found on its website. It is recommended that you install the library locally for testing.

On the CSIE workstations, the library (and the `cryptominisat5` binary) should be already installed. **Note that you need to compile your program with the flag `-lcryptominisat5`.**

On systems with the proper dependencies (most notably, `make` and `cmake`) installed, you can build *Cryptominisat5* and compile your program with it by following this screencast[8].

However, if you have difficulty installing the library, we also provide the following alternative:

Please paste the provided header[9] at the beginning of your source file. The following functions are then provided:

- `void sat::Init(int n)`: Initialize the solver with `n` variables.
- `void sat::AddClause(std::vector<int> v)`: Add a CNF clause that consists of the variables in `v`. Note that negative numbers denote the negation of a variable (e.g., `{1, -2}` $\iff (x_1 \vee \neg x_2)$), and thus the variable numbering has to start from 1.
- `bool sat::Solve()`: Solve the given clauses. Returns true if a solution is found and false otherwise.
- `int sat::GetResult(int id)`: Get the result of the variable `id` after calling `Solve()`. Returns 1 if the variable is true, $-1$ if it is false, and 0 if it is indeterminate.

If the macro `DIMACS` is *not* defined, the library merely redirects your calls to *Cryptominisat*. However, if it is defined, then `Solve()` writes your clauses to `out.dimacs` in DIMACS format[10] and terminates the program. You can then feed the file to other solvers with standalone binaries, such as *Cryptominisat5*[11], *Microsat*[12], *Minisat*[13], *Glucose*[14], *Lingeling*[15], or even browser-based solvers like *Minisat.js*[16].[17]

## Hints

- Tseytin transformation[18]
- Truth table $\iff$ CNF
- Incremental SAT[19]
- Flood fill
- `https://codingnest.com/modern-sat-solvers-fast-neat-underused-part-1-of-n/`
- It is possible to solve this *without* using the solver, but I doubt that it is easier :)

[7] `https://www.msoos.org/cryptominisat5/`
[8] `https://asciinema.org/a/oRbZ9UjnWL36RfEZocJYmCyVZ`
[9] `https://gist.github.com/WillyPillow/a5b4f12faefd66c7ce42690668c002b5`
[10] `https://en.wikipedia.org/wiki/Boolean_satisfiability_problem#SAT_problem_format`
[11] Binaries for Windows and Linux can be found at `https://github.com/msoos/cryptominisat/releases`.
[12] `https://github.com/marijnheule/microsat`
[13] `http://minisat.se/`
[14] `https://www.labri.fr/perso/lsimon/glucose/`
[15] `http://fmv.jku.at/lingeling/`
[16] `http://jgalenson.github.io/research.js/demos/minisat.html`
[17] Other solvers can be found at `https://en.wikipedia.org/wiki/Boolean_satisfiability_problem#External_links`.
[18] `https://en.wikipedia.org/wiki/Tseytin_transformation`
[19] `https://baldur.iti.kit.edu/sat-competition-2016/index.php?cat=incremental`

# Problem B - Reachability Coefficient (Programming) (15 points)

## Problem Description

For two sets $X$ and $Y$, we define:

$$f(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

You are given a directed acyclic graph $G$, and for each vertex $v$, let $S(v)$ be the set of vertices that are reachable from $v$ (vertex $u$ is reachable from vertex $v$ if there exists a simple path from $v$ to $u$). $Q$ queries in the form of $u_i, v_i$ are specified, where $u_i$ and $v_i$ are vertices in $G$. You need to answer $f(S(u_i), S(v_i))$, for all given queries.

You are not asked to find the **exact** ratio. Instead, your answer is considered correct if the **absolute error** does not exceed 0.1.

## Input

The first line of the input contains three integers $N, M, Q$, representing the number of vertices, the number of edges, and the number of queries, respectively. $M$ lines follow, the $i$-th of which contains two integers $u_i, v_i$, representing a directed edge from the vertex $u_i$ to the vertex $v_i$. $Q$ lines follow, the $i$-th of which contains two integers $u_i, v_i$, representing a query on the vertex $u_i$ and the vertex $v_i$.

- $1 \leq N \leq 2 \times 10^5$
- $0 \leq M \leq 3 \times 10^5$
- $1 \leq Q \leq 2 \times 10^5$

## Output

Print the answer to each query. Your answer can be considered correct if the *absolute error of it to the actual answer does not exceed* 0.1.

### Test Group 0 (20 %)

- $1 \leq N, Q \leq 1000$
- $0 \leq M \leq 5000$

### Test Group 1 (80 %)

- No other constraints

### Sample Input 1

```
6 7 5
1 2
1 3
2 3
3 4
4 5
3 5
5 6
1 3
2 4
1 6
5 5
2 5
```

### Sample Output 1

```
0.666666666667
0.600000000000
0.166666666667
1.000000000000
0.400000000000
```

### Hints

- Read the problem statement carefully.
- Read the output section carefully.

# Problem C - Yet Another Permutation Problem (Programming) (15 points)

## Problem Description

Let $S_N$ be the set of all permutations of length $N$. Let $M_N$ be the set of all 0-1 matrices of size $N \times N$. We define

$$f : S_N \times M_N \to M_N$$

with $f(p, A)_{i,j} = A_{p_i,p_j}$. That is, the function $f$ interchanges the rows (as well as the columns) of matrix $A$ according to the permutation $p$.

Let $g(A)$ be the maximum size of the submatrix $B$ of $A \in M_N$ containing only 1's such that the diagonal of $B$ lies on the diagonal of $A$. Formally speaking, $g(A)$ is the maximum $r$ $(0 \le r \le N)$ such that there exists $1 \le i \le N - r + 1$ with the following property:

$$A_{xy} = 1, \text{ for all } i \le x, y \le i + r - 1$$

Given a $N \times N$ 0-1 matrix $A$, please find a permutation $p \in S_N$ such that $g(f(p, A))$ is maximized.

## Input

The first line of the input contains an integer $N$, indicating the size of the matrix $A$. $N$ lines follow, the $i$-th of which contains a string of length $N$, denoting the $i$-th row of the matrix $A$.

- $1 \le N \le 120$
- $A_{ij} \in \{0, 1\}$

**Test Group 0 (10 %)**

- $1 \le N \le 8$

**Test Group 1 (10 %)**

- $1 \le N \le 20$

**Test Group 2 (50 %)**

- $1 \le N \le 80$

**Test Group 3 (30 %)**

- No other constraints.

## Output

Print the optimal permutation $p$. If there are more than one solutions, any of which will be accepted.

## Sample Input 1

```
3
101
000
101
```

## Sample Output 1

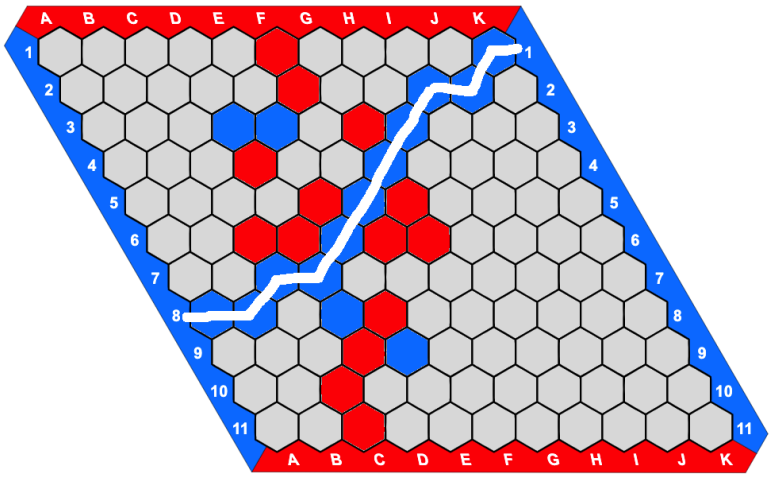
```
1 3 2
```

# Problem D - Hex (Programming) (10 points)

## Problem Description

In this problem, you need to play the game, hex, with the computer. Please read Wikipedia for the rules[20]. In this problem, the **pie rule** (or **swap rule**) is not used.

To try the game online, you can use this site[21]. (Make sure to uncheck **swap rule** first.)




a 11x11 Hex Board that blue wins

To solve this problem, you need to write a program to play with ours. Your program will be accepted if you win $Y$ rounds out of a total of $X$ rounds. You are always the first player.

In your program, you only need to implement two functions:

- `void init(int n)`: Start a new game with a `n`x`n` board
- `std::pair<int, int> decide(std::pair<int, int> p)`: Decide your move. `p` is the last move by the opponent. If it is the first move, `p` would be `{-1, -1}`. The function should return the position you plan to put you color.

The position on the board is represented by a pair(`std::pair<int, int> p`), where `p.first` is the index (starting from 0) of the row and `p.second` is the index (starting from 0) of the column.

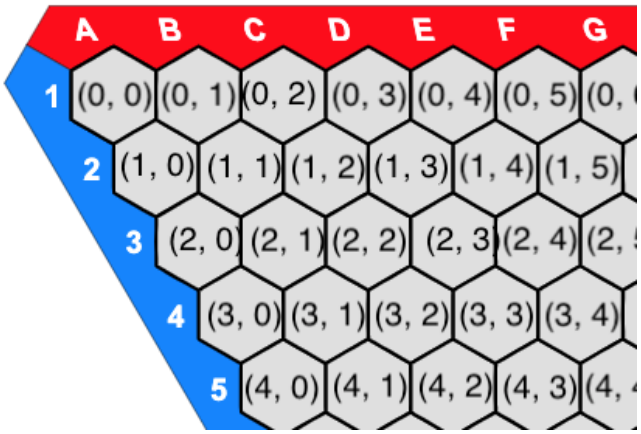
## Example Files

You can download a random sample solution and `hex.h` at: `https://cool.ntu.edu.tw/courses/368/files/folder/Homework/Homework4\%20Hex`

The `hex.h` file would connect to our server, which is the same AI as the judge. So please send your solution to the judge only if the program already runs correctly at local. **You have a very limited quota per day!**

To compile sample files on Windows, you need to add `-lws2_32` in the compiler options.

[20] `https://en.wikipedia.org/wiki/Hex_(board_game)#Game_play`
[21] `http://www.lutanho.net/play/hex.html`




the index of grids on a board

### Input Limits

- $1 \leq X \leq 20$
- $1 \leq Y \leq X$
- $3 \leq \texttt{n} \leq 11$
- $0 \leq \texttt{q.first}, \texttt{q.second} < \texttt{n}$

### Test Group 0 (0 %)

- $X = 1$
- $\frac{Y}{X} \leq 0.0$
- $\texttt{n} = 3$

### Test Group 1 (20 %)

- $\frac{Y}{X} \leq 0.9$
- $\texttt{n} = 3$

### Test Group 2 (20 %)

- $\frac{Y}{X} \leq 0.8$
- $\texttt{n} = 4$

### Test Group 3 (30 %)

- $\frac{Y}{X} \leq 0.4$
- $\texttt{n} = 8$

### Test Group 4 (15 %)

- $\frac{Y}{X} \leq 0.7$
- $\texttt{n} = 5$

**Test Group 5 (15 %)**

- $\frac{Y}{X} \leq 0.8$
- $\texttt{n} = 8$

## Problem E - Magic Wands Linking (Hand-Written) (25 points)

Have you seen *Harry Potter*? It is a fantasy story about wizards and magic. Every wizard has a magic wand, and there is a **fitness** value (which is always non-negative) between every pair of wands.

In the fantasy world, there are many phenomenal professors who teach at Hogwarts. Recently, one of the professors invented a new technology called **Magic Wands Linking**, which can greatly enhance the power of wizards. By linking two wands through the technology, both owners can increase their power by the fitness value between the two wands. For example, if a wand $w_1$ is linked to another wand $w_2$, and the fitness between $w_1$ and $w_2$ is 10, then both the power of $w_1$ and $w_2$ is enhanced by 10; the power of the Wizarding World is enhanced by 20 in total. Note that two wands can be linked even if their fitness is 0, but such a linking has no effect on their power. Also, a wand can be linked to more than one wand, and the power enhanced can be accumulated.

However, the technology is not yet mature. The links between the wands should follow the rules below:

1. **RULE1** There are three attributes of magic wands, **none**, **light**, and **dark**. Initially, all wands are unlinked and have the attribute **none**. If a wand $w_1$ is linked to another wand $w_2$, then the two wands must be assigned two opposite attributes, **light** and **dark**. (A wand can only be assigned an attribute once.)
2. **RULE2** To maintain the balance of the Wizarding World, the number of light wands and the number of dark wands should be the same.

Given $N$ magic wands ($\{w_1, w_2, ..., w_N\}$) and $M$ fitness values, can you help the wizards to find the maximum power they can enhance in total? For all of the following questions, you can assume that $N$ **is even,** $M = \binom{N}{2}$**, and the fitness values are always non-negative**.

(1) (4pts) Please solve two problems below.

(1-1) (2pts) Assume that:
N = 4 ($\{w_1, w_2, w_3, w_4\}$),
M = 6 ($\{(w_1, w_2, 5), (w_1, w_3, 1), (w_1, w_4, 8), (w_2, w_3, 1), (w_2, w_4, 7), (w_3, w_4, 4)\}$).
$(w_1, w_2, 5)$ means that the fitness between $w_1$ and $w_2$ is 5.
Under RULE1 and RULE2, what is the maximum power they can enhance in total?
Please provide the linking method as well.

(1-2) (2pts) Following the question (1-1), now assume that we can temporarily **ignore RULE2**. That is, $N$ is still an even number, but the numbers of light wands and dark wands can be different. What is the maximum power they can enhance in total? Please provide the linking method as well.

(2) (9pts) Please solve two problems below.

(2-1) (4pts) **Maximum Cut Problem**:
For a graph, a maximum cut is a cut whose size is at least the size of any other cut. The problem of finding a maximum cut in a graph is known as the Maximum Cut Problem.
Assuming that Maximum Cut Problem is NP-hard, prove the following problem is NP-hard:
**Ignore RULE2**, given $N$ magic wands and $M$ fitness values, find a linking method to enhance the maximum power in total.

(2-2) (5pts) **Ignore RULE2**, given $N$ magic wands and $M$ fitness values, now consider an approximation algorithm below:
Define $M_i = \{fitness\ of\ (w_k, w_i) \mid k < i\}$, $W_1$ and $W_2$ as two sets of wands. Initially, set $W_1 = \{w_1\}, W_2 = \phi$. Then for each $i$ from 2 to $N$, add $w_i$ to either $W_1$ or $W_2$, and link $w_i$ to all wands in the other set, such that the power enhanced in $M_i$ is maximized.

Prove that the above algorithm is a 2-approximation.

(3) (5pts) Now, **under both RULE1 and RULE2** above, assuming that Maximum Cut Problem and the question (2-1) are NP-hard, prove that the following problem is NP-hard:

Given $N$ magic wands and $M$ fitness values, find a linking method to enhance the maximum power in total.

(4) (4pts) The professor who invented Magic Wands Linking realizes that the total power enhanced by linking wands can not be too large, or the Wizarding World will crash due to the power storm! Thus, now he wants to know the minimum power that can be enhanced in total when every wand has at least one linking, so that he can evaluate the feasibility of Magic Wands Linking. However, the problem is harder than he thought.
Formally speaking, assuming that Maximum Cut Problem, the question (2-1), and the question (3) are NP-hard, please help him prove that the problem below is NP-hard:
Under both RULE1 and RULE2 above, given $N$ magic wands and $M$ fitness values, find a linking method such that **for each wand $w$, there exists at least one link to** $w$, but the total enhanced power is minimized.

(5) (3pts) One week after all magic wands are linked, the Dementors attack Hogwarts suddenly!

In order to send wizards to the battlefield, they use two teleportation spells. However, teleportation takes time, so it is better to distribute the wizards evenly to the two spells.
Thus, the problem is:
Assuming that there are $N$ wizards, given $N$ non-negative integers $(t_1, t_2, ..., t_N)$ indicating the time each wizard takes to teleport, please determine whether there is a subset whose sum equals to $\sum_{i=1}^{N} t_i/2$.

Please reduce the **Subset Sum Problem**[22] to the problem above in polynomial time.

[22] Given $n$ non-negative integers $(a_1, a_2, ...a_n)$ and a positive integer $W$, the Subset Sum Problem seeks whether there is a subset whose sum equals to $W$. The Subset Sum Problem is known to be NP-hard.

## Problem F - Band Dream (Hand-Written) (20 points)

Kasumi is the leader of the band Poppin' Party. Because it is the fifth year since Poppin' Party was created, she wants to create a new song containing continuous melodies of the songs that Poppin' Party has released before.

For example, if there are two melodies **Do, Re, Mi** and **Mi, Re, Do**. Then **Do, Re, Mi, Re, Do** contains two sub-melodies above but **Do, Re, Do, Mi, Re, Do** does not contain the sub-melody **Do, Re, Mi**.

Because she does not know how to minimize the length of the new song, she wants us to find a song that is *short enough* in polynomial time.

(1) To solve this problem, we introduce another problem called **Set Cover Problem** and one of its approximate algorithms.

Given a universe of $n$ elements $X$ and a collection of $m$ subsets $F = \{S_1, S_2, \ldots, S_n\}$, where $S_i \subset X$ for each $S_i \in F$. Each $S_i$ has a **positive** cost, denoted as $\text{cost}(i)$. The goal is to find a sub-collection $C \subset F$ of the minimum total cost that covers $X$, assuming one exists.

**Algorithm 1:** GreedySetCover(X, F)

```
C ⟵ ∅
U ⟵ X
while u ≠ ∅ do
    Find a set S ∈ F \ C that minimizes α = cost(S) / |S∩U|
    for x ∈ S ∩ U do
        price(x) ⟵ α
    C ⟵ C ∪ {S}
    U ⟵ U \ S
return C
```

(1-1) Order $X$ as $\{x_1, x_2, \ldots, x_n\}$ by the order in which they were covered by the algorithm. If there is more than one element covered in the same iteration, order them arbitrarily. Please show that for each $k \in \{1, 2, \ldots, n\}$, $\text{price}(x_k) \leq \frac{\text{OPT}}{n-k+1}$, where OPT is the cost of the optimal cover. (4 points)

(1-2) Please show that the above algorithm is in polynomial time, and it will obtain a ($\ln n + O(1)$)-approximate solution. You can use the result of (1-1) even if you are not able to prove it. (3 points)

(2) The problem that Kasumi wants to solve can be modelled as below: Given a finite set of alphabet $\Sigma$, and a set of $n$ strings $M = \{M_1, M_2, \ldots, M_n\}$, where each $M_i \in \Sigma^*$. WLOG, assume there is no $M_i$ is sub-string of $M_j$, for $i \neq j$. Find the shortest string $C$ that contains all sub-strings $M_i \in M$. To make things easier, if a string $s$ is a prefix of the string $y$, and a string $t$ is a suffix of the string $y$, and $\textbf{len}(s) + \textbf{len}(t) > \textbf{len}(y)$, we can call $y$ a possible **merge** of $s$ and $t$. For example, "`abbbc`" and "`abbbbc`" are possible **merge** of "`abbb`" and "`bbc`", while "`abbbbbc`", "`abbbbbbc`" are all not possible **merge**s.

(2-1) Please show that we can reduce Kasumi's problem to Set Cover Problem mentioned at the problem (1) in polynomial time, by picking **all strings** and **all possible merges of each pair of strings** in $M$ as the collection $F$ of subsets. Notice that some merges may contain more than two sub-strings. (6 points)

(2-2) Please show that the solution in (2-1) will obtain a $(2 \ln n + O(1))$-approximate solution. You can use the result of (1-2) even if you are not able to prove it. (7 points)