Homework #4

Due Time (hand-written): 2022/01/04 14:20
Due Time (programming): 2022/01/11 14:20
Red Correction Date: 2021/12/18 19:20
Blue Correction Date: 2021/12/26 14:20
Contact TAs: `ada-ta@csie.ntu.edu.tw`

Instructions and Announcements

- There are **four programming problems** and **two hand-written problems**.
- Given the difficulty of the programming problems, the due time of the programming problems are extended by one week. Note that the due time of the hand-written problems is unchanged.
- **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 should upload your answer to **Gradescope** as demonstrated in class. 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 URL of the website you consulted or the people you discussed with) on the first page or comment in code of your solution to that problem. You may get zero points due to the lack of references.
- **Tips for programming problems.** Since the input files for some programming problems may be large, please add
  - `std::ios_base::sync_with_stdio(false);`
  - `std::cin.tie(nullptr);`

  to the beginning of the main function if you are using `std::cin`.

Problem 1 - ♈ Clique (Programming) (13 points)

Problem Description

♈ loves complicated things. Hence, in the graph theory, cliques are ♈'s favorite since they are the most complex structure within all simple graphs. One day, ♈ found a new interesting game called "click clique" on a gaming website. The rule of the game is described as follows.
When the game starts, you are given a simple undirected graph $G = (V, E)$. For each move, you can click a clique C of G, and it will remove those vertices and edges from the graph. That is, the graph will become the induced subgraph of $V \backslash C$ after the move. You can make arbitrary number of moves until the graph is empty. When the game ends (i.e. the graph becomes empty), you will receive stars according to the number of moves you make. Furthermore, you can receive full 3 stars only when you make the minimum number of moves. As a clique lover, ♈ can't bear to miss the full stars in the game. Thus, ♈ comes and asks you for help. Given a simple graph, please tell ♈ an optimal way to finish the game.

Input

The first line of the input contains an integer T indicating the number of testcases. For each testcase, the first line contains two integers N, M indicating the number of vertices and edges in the graph, respectively. The following M lines each contains two integers u, v describing the edges of the graph.

- $1 \leq T \leq 1000$
- $1 \leq N \leq 80$
- $0 \leq M \leq \frac{N(N-1)}{2}$
- $0 \leq u, v < N$
- $1 \leq C \leq 800$ where C is the number of maximal clique in the graph.
- It is guaranteed that the given graph is simple.
- Each test group will include its previous test groups.

Test Group 0 (0 %)

- Sample Input

Test Group 1 (10 %)

- $n \leq 7$
- $C \leq 800$

Test Group 2 (20 %)

- $n \leq 14$
- $1 \leq C \leq 100$

Test Group 3 (30 %)

- $T \leq 10$
- $n \leq 50$
- $C \leq 500$

Test Group 4 (40 %)

- $T \leq 3$

Output

For each testcase, the first line output an integer R, which is the minimum number of moves $\mathcal{P}$ needs to make.
Then each of the following R lines output the vertices that $\mathcal{P}$ should click in each move in the format of "k v_1 v_2 ... v_k", where k is the number of vertices in this move and $\{v_i\}$ are the indices of these vertices. You can output $\{v_i\}$ in any order.
Note that the summation of k in each testcase should be equal to N. If there are multiple optimal solutions, you can output any of them.

Sample Input 1

```
1
6 6
0 1
0 2
1 2
1 3
3 4
4 5
```

Sample Output 1

```
3
3 0 1 2
1 3
2 4 5
```

Sample Input 2

```
2
3 3
1 2
0 2
0 1
3 0
```

Sample Output 2

```
1
3 2 0 1
3
1 2
1 0
1 1
```

Sample Input 3

```
2
4 2
1 2
0 2
4 6
0 1
1 2
2 3
0 3
0 2
1 3
```

Sample Output 3

```
3
2 0 2
1 1
1 3
1
4 1 2 0 3
```

Notes

A clique is defined as a complete subgraph. That is, a set $C \subseteq V$ is a clique if and only if $\forall u, v \in C$ and $u \neq v$, $(u, v) \in E$.

A maximal clique is a clique that cannot be extended by adding any other vertex. That is, there is no proper superset of C which is also a clique. Formally speaking, a clique C is maximal if and only if $\nexists C \subset S \subseteq V$ such that S is a clique.

Hints

1. `std::bitset` is a good data structure and it might be useful in this problem.
2. You may want to use (integer) linear programming to solve this problem. Please see the following section to use the provided linear programming solver. (Of course, you are not forced to use it if you want to implement it by yourself.)

3. You are welcome to use any heuristic method to solve this problem. Happy hacking.

Linear Programming Solver

The GLPK (GNU Linear Programming Kit)[1] package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. You can include the GLPK library (`<glpk.h>`) and use it in this problem. See GLPK Manual[2] for more information about the usage.
There is also a packed version of the GLPK tools provided for you to use it easier. The full header file can be downloaded here[3]. You can include it by `#include "ypglpk.hpp"` in your solution.
There are three functions defined in the namespace "ypglpk". You can follow the instruction below to use them. All the std::vector below are 0-based. Note that if the parameters violate the restriction, the behavior is undefined.

```
std::pair<double,std::vector<double>> linear_programming(
const std::vector<std::vector<double>> &A,
const std::vector<double> &b, const std::vector<double> &c);
```

- The function `linear_programming(A,b,c)` is used to solve the standard linear programming problem.
- Parameters: $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$ where n is the number of structural variables and m is the number of constraints.
- Targets: Maximize $c \cdot x$ subject to $Ax \leq b, x \in \mathbb{R}^n$.
- Return value: pair of (optimal value $c \cdot x^*$, optimal vector x^*). If the solution is infeasible or unbounded, the return value is (negative infinity `-ypglpk::INF`, empty vector `vector<double>()`).

```
std::pair<double,std::vector<double>> mixed_integer_linear_programming(
const std::vector<std::vector<double>> &A, const std::vector<double> &b,
const std::vector<double> &c, const std::vector<bool> &isint);
```

- The function `mixed_integer_linear_programming(A,b,c,isint)` is used to solve the mixed integer linear programming problem, which can restrict some structural variables to be integers.
- Parameters: $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n, \text{isint} \in \{true, false\}^n$ where n is the number of structural variables and m is the number of constraints.
- Targets: Maximize $c \cdot x$ subject to $Ax \leq b, x \in \mathbb{R}^n$, and $\forall i$ with $\text{isint}_i = true, x_i \in \mathbb{Z}$.
- Return value: pair of (optimal value $c \cdot x^*$, optimal vector x^*). If the solution is infeasible or unbounded, the return value is (negative infinity `-ypglpk::INF`, empty vector `vector<double>()`).

```
void set_output(bool output);
```

[1] https://www.gnu.org/software/glpk/
[2] http://most.ccib.rutgers.edu/glpk.pdf
[3] http://www.csie.ntu.edu.tw/~giver/ADA/hw4/ypglpk.hpp

- The function `set_output(output)` is used to set whether GLPK to output verbose information about the (MI)LP solver.
- The default setting is `output=false`. You may enable it to debug your code with more information about the constraints and the solver. **However, please do not set it to `true` when you submit to the judge; otherwise, you will certainly get `Wrong Answer` verdict due to the unexpected output.**

Local Testing

Here[4] is a simple example code to demonstrate how to use the provided header file. It should be able to be compiled and run successfully on the judge (although you will receive `Wrong Answer`).
The installed GLPK version on the judge system is 5.0. (The version of the GLPK package does not affect the correctness. It only affects the performance in some cases since the later version might have better presolver and branching strategy.)
For CSIE students, you are highly recommended to run the program on the CSIE workstation, as there is already an installed GLPK 5.0 package for you. You can simply put your source code and the given header file together and compile them with the command below Compilation.
Otherwise, to compile and run the GLPK library successfully, please refer to the Appendix GLPK Installation to install it.

Compilation

After having an environment with installed GLPK package, you can compile the provided codes (or your source code) by adding the additional linker argument `-lglpk` to the compilation command. **Note that the order of the arguments is essential. The linker argument should be put after the source code.** For example (assume that the filename of the source code is "example.cpp"):

```
g++ -std=c++17 -O2 -oexample example.cpp -lglpk -Wall
```

Demonstration

You can try to compile and run the provided example program to ensure there is no problem with the package. Below is a script to download the provided files, compile them, and run the program. (Note that the compilation command and the execution command might differ if you built the package from source by yourself. See the Appendix GLPK Installation for more information.)

```
curl -o example.cpp 'https://www.csie.ntu.edu.tw/~giver/ADA/hw4/example.cpp'
curl -o ypglpk.hpp 'https://www.csie.ntu.edu.tw/~giver/ADA/hw4/ypglpk.hpp'
g++ -std=c++17 -O2 -oexample example.cpp -lglpk -Wall
./example
```

And the expected output of the program execution should be like figure 1.

[4] http://www.csie.ntu.edu.tw/~giver/ADA/hw4/example.cpp

```
giver@GiverArchLinux /t/tmp.7DFD7aBTUt> curl -o example.cpp 'https://www.csie.ntu.edu.tw/~giver/ADA/hw4/example.cpp'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   976  100   976    0     0   3728      0 --:--:-- --:--:-- --:--:--  3725
giver@GiverArchLinux /t/tmp.7DFD7aBTUt> curl -o ypglpk.hpp 'https://www.csie.ntu.edu.tw/~giver/ADA/hw4/ypglpk.hpp'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2898  100  2898    0     0  55511      0 --:--:-- --:--:-- --:--:-- 55730
giver@GiverArchLinux /t/tmp.7DFD7aBTUt> g++ -std=c++17 -O2 -oexample example.cpp -lglpk -Wall
giver@GiverArchLinux /t/tmp.7DFD7aBTUt> ./example
GLPK Simplex Optimizer 5.0
4 rows, 3 columns, 10 non-zeros
Preprocessing...
4 rows, 3 columns, 10 non-zeros
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  3.000e+00  ratio =  3.000e+00
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part is 4
*     0: obj =  -0.000000000e+00 inf =   0.000e+00 (3)
*     3: obj =   1.273823529e+01 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
LP: max=12.7382 with x=3.63529 y=0.941176 z=-2.88824
GLPK Integer Optimizer 5.0
4 rows, 3 columns, 10 non-zeros
2 integer variables, none of which are binary
Preprocessing...
4 rows, 3 columns, 10 non-zeros
2 integer variables, none of which are binary
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  3.000e+00  ratio =  3.000e+00
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part is 4
Solving LP relaxation...
GLPK Simplex Optimizer 5.0
4 rows, 3 columns, 10 non-zeros
*     0: obj =  -0.000000000e+00 inf =   0.000e+00 (3)
*     3: obj =   1.273823529e+01 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
Integer optimization begins...
Long-step dual simplex will be used
+     3: mip =     not found yet <=              +inf        (1; 0)
+     6: >>>>>   1.130000000e+01 <=   1.130000000e+01   0.0% (3; 0)
+     6: mip =   1.130000000e+01 <=     tree is empty   0.0% (0; 5)
INTEGER OPTIMAL SOLUTION FOUND
MILP: max=11.3 with x=1.3 y=0 z=-4
```

Sample output of the example.cpp program

Problem 2 - ♈ Game (Programming) (13 points)

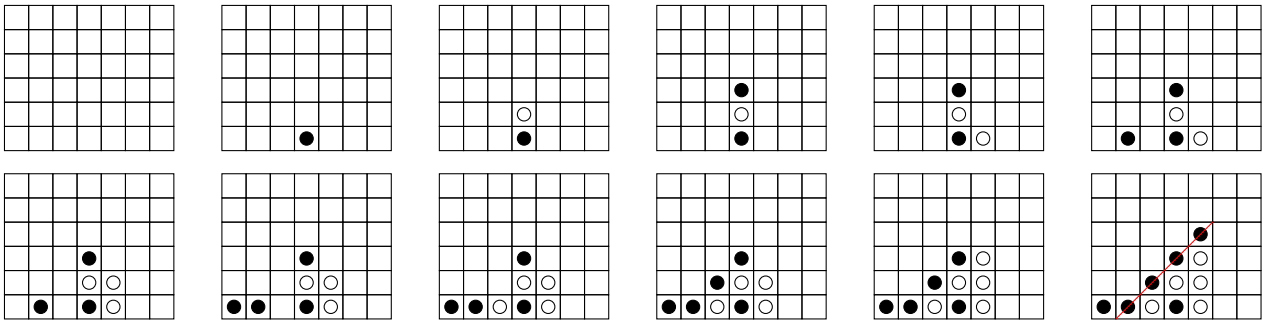
This task has unusual time limit and submission quota. Please read the notes carefully.

Problem Description

King ♈ loves to play board games. He is especially good at playing Connect4.

Connect Four (Connect4) is a two-player connection board game in which the players take turns dropping their pieces into a **seven-column**, **six-row** vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. The game's objective is to be the first to form a horizontal, vertical, or diagonal line of four with one's pieces. The game is determined to be a tie if a player can't make a move (i.e., the board is fully filled).

The following figure shows an example game:



You, as a local Connect4 champion, are invited by King ♈ to have a series of matches with him. Since King ♈ is a nice person, **he will always let you move first**. Additionally, since King ♈ is very busy, he can't pay full attention to the games. Sometimes, he will make a random move instead of using his strong mind. Given these advantages, can you defeat King ♈?

Scoring

There are multiple test groups. In each test group, there are two parameters, a probability p and a threshold *thres*. For each testcase within the test group, you will play **100 games** of Connect4 against King ♈. When it's King ♈'s turn, he has a probability p to make the decision using his strong mind. Otherwise, he will make a random move (each valid column has the same chance to be chosen).

Suppose that you got x wins and y ties out of all 100 games in a single testcase, your score *score* is defined as $2x + y$. That is, **each win gives you two points, while each tie gives you a single point**. You will get **Accepted** iff $score \geq thres$.

Test Group 0 (2 %)

- King ♈ always drops his piece in the leftmost valid column.
- $thres = 190$

Test Group 1 (8 %)

- $p = 0.0$
- $thres = 150$

Test Group 2 (10 %)

- $p = 0.5$
- $thres = 120$

Test Group 3 (10 %)

- $p = 0.5$
- $thres = 170$

Test Group 4 (15 %)

- $p = 0.7$
- $thres = 110$

Test Group 5 (15 %)

- $p = 0.85$
- $thres = 110$

Test Group 6 (15 %)

- $p = 0.85$
- $thres = 140$

Test Group 7 (25 %)

- $p = 0.85$
- $thres = 170$

Implementation details

First of all, **the columns are numbered from** 0 **to** 6. You need to implement the following procedure:

```
int decide(int yp_move)
```

In each game, `decide(-1)` will be called first. You should return the first move (index of the column) you want to make. Every subsequent call of the function denotes that King ♈ placed his piece in column `yp_move`. You should return the next move you want to make.

It is guaranteed that the game hasn't finished yet when `yp_move` $\neq$ `-1`. However, there are multiple games within a single testcase. **Please properly reset all your variables whenever `decide(-1)` is called.**

Local testing tool & Template

We've provided a local testing tool for you. The tool will connect to our sever (**so it needs network connectivity**) and play against the **exactly same** strategy on the judge (i.e., King ♈). So **please submit your solution to the judge only if it runs correctly at local.**

Please download `connect4_public.zip`[5], which contains the following two files:

`connect4.cpp`

This is the template file for you to work on. This is also the file you should submit for this problem. **You can pass Test Group 0 by submitting this file without any modifications.** You can (and should) include any header you need on your own. Feel free to use global variables.

[5] http://w.csie.org/~b08902107/ADA/connect4_public.zip

`connect4.h`

This is the local testing tool. We already included this file in your solution. You should directly compile `connect4.cpp`. It's recommended to add `-std=c++17` and `-O2` flag when compiling locally. **For Windows users, you need to append `-lws2_32` to the compiler option.**

It will ask for p and the number of games you want it to repeat. It is encouraged to take a look at the tool and change it as you like. Feel free to borrow any code from the tool.

Additionally, there are two "switches" located at the very beginning of this tool:

- `CONNECT4_DEBUG`: The tool will output the state of the game board if this flag is defined.
- `CONNECT4_INTERACTIVE`: The tool will let you (instead of King ♈) play with your own code if this flag is defined.

About the execution time

The local testing tool will help you measure the total time it takes to run your code on your computer (not including King ♈'s thinking time). King ♈ will spend approximately 3 seconds per testcase on the judge. However, beware that the execution time might change a lot on the judge. It's a good idea to run the same code locally and on the judge once and take the time difference as a reference. The time limit to this problem should be more than enough.

Notes

1. The time limit for this problem is **40 seconds per testcase** (100 games). However, **you can only submit 5 times per day**.
2. Do not write your own `main` function and do not try to interact with stdin and stdout.
3. Feel free to use any random number generators. However, it's encouraged to use a fixed seed so that your judge verdict can be deterministic.
4. The judge uses a fixed random seed, so you should get the same verdict (Accept or Wrong Answer) if you submit the same deterministic code twice. (Note that this does not hold for local testing tool, as it uses a different seed for every new invocation.)
5. Do not try to steal any data from the grader. Similar behavior will be regarded as cheating and will receive heavy penalties. If you find weird behavior from the grading or the local test tool, please contact TA.

Hints

1. It is important to examine why your code loses.
2. It might be easier to debug by playing with your own code (see the instruction above on how to do this).
3. This is an open question, we don't have a model solution. You can use your imagination to come up with some good strategies.
4. You can ask TAs for some additional hints if you don't have any idea. However, we can't really debug your ideas for you.
5. You can also come to play with TAs if you like.

Problem 3 - 𝔓 Record (Programming) (11 points)

Problem Description

As a competitive programming athlete, 𝔓 has participated in N contests, and the i^{th} contest has a unique ID i. However, 𝔓 knows that he got score s_i in the i^{th} contest and feels slightly dissatisfied with those results. Due to his self-esteem, 𝔓 wants to let his scores become **increasing** over time by concealing some of the contest records.

𝔏, who is the enemy of 𝔓, would like to interfere him. Since 𝔏 is a great time traveler, he will travel to the past and do one of the following operations many times:

1. Convince 𝔓 to participate in an extra contest.
2. Convince 𝔓 not to participate in a contest.

Notice that 𝔏 may perform the operations at any time. Any operation will not affect the scores of other contests, you don't need to worry about the Butterfly Effect.

𝔓 entrusts you, his assistant, to perform the concealment. Unfortunately, 𝔓 doesn't want to tell you the exact scores of each contest. You can only ask him many times about "whether the score of the a^{th} contest is less than the b^{th} contest", but not too many because 𝔓 is busy. You also need to keep updating the concealment after 𝔏 has done any operation.

Oh, 𝔓 does not care about minimizing the number of concealed contests. **𝔓 will accept any approximate solution that is at most twice as many as the minimum.**

Implementation details

Include `"ada-hw4-p3.h"` in the first line of your code, and then you should implement the following procedures:

```
std::vector<int> init(int N)
```

- N: 𝔓 has participated in N contests initially. The i^{th} contest's position is i initially.
- This procedure is called exactly once, and should return a single array D. The elements of the returned array should form a subset of 0 to $N-1$ in any order, describing the IDs of contests you want to conceal. An empty array is also valid when you don't want to conceal any contest.

```
std::vector<int> insert(int p, int id)
```

- p: position of the inserted contest. After the insertion, the id^{th} contest will have position p, and the original contests whose positions are greater than or equal to p will be increased by one.
- id: the ID of the inserted contest.
- This procedure should return a single array D. The elements of array should form a subset of the current contest **ID** list (after the insertion) in any order, describing the IDs of contests you want to conceal. An empty array is also valid when you don't want to conceal any contest.

```
std::vector<int> remove(int p)
```

- p: position of the removed contest. After the deletion, the contest having position p will be removed, and the original contests whose positions are greater than p will be decreased by one.
- This procedure should return a single array D. The elements of array should form a subset of the current contest **ID** list (after the removal) in any order, describing the IDs of contests you want to conceal. An empty array is also valid when you don't want to conceal any contest.

The above procedures can make calls to the following procedure:

```
bool compare(int a, int b)
```

- a: the first contest's ID you want to compare.
- b: the second contest's ID you want to compare.
- Both the the compared contests should exist (a removed contest is regarded as disappeared). And $a \neq b$.
- The procedure returns `true` if the a^{th} contest's score is less than the b^{th} contest's score. Otherwise, it will return `false`.
- The grading program will record the number of calls of this procedure, which is relate to the condition of getting `Accepted`.
- The time complexity of the procedure is $O(1)$.

Examples

Consider a scenario in which $\mathcal{V}$ has participated in 3 contests with scores $[1, 4, 2]$, in order. The procedure `init` is called in the following way:

```
init(3)
```

This procedure may call `compare(0, 1)` which (in this scenario) returns `true`. It may then call `compare(1, 2)`, which returns `false`.

At this point, there is sufficient information to conclude that we should conceal at least one contest. So, the procedure `init` can return [2] and get `Accepted`.

Notice that we can return an answer having twice of the minimum answer size. Hence, if the procedure `init` returns $[1, 2]$, it can also get `Accepted`.

After that, the procedure `insert` may be called in the following way:

```
insert(1, 3)
```

It means that $\mathcal{L}$ had convinced $\mathcal{V}$ to participate in the 3^{rd} contest at position 1. Assume that the score of it is 3. The contest ID list is $[0, 3, 1, 2]$ and the contest score list is $[1, 3, 4, 2]$ now.

This procedure may call `compare(3, 1)` which returns `true`. At this point, there is sufficient information to conclude that we can conceal only one contest. So, the procedure `insert` can return [2] and get `Accepted`.

Continue, the procedure `remove` may be called in the following way:

```
remove(3)
```

At this point, there is sufficient information to conclude that we don't need to conceal any contest. So, the procedure `remove` can return [] and get `Accepted`.

Notice that returning any non-empty array cannot get `Accepted` since the minimum answer size is 0.

Constraints

Let the number of calls of `insert` and `remove` be P and Q, respectively. Then:

- $1 \leq N \leq 2000$
- $0 \leq P, Q \leq 1000$

For each call to `insert`, let L be the current length of the array:

- $0 \leq p \leq L$
- $id = N + i - 1$ in the i^{th} call of `insert`

For each call to `remove`, let L be the current length of the array:

- $0 \leq p < L$

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (20 %)

- No additional constraint.

Test Group 2 (20 %)

- For the call of `init`, procedure `compare` can be called at most $N - 1$ times.
- $P = Q = 0$.

Test Group 3 (30 %)

- For the call of `init`, procedure `compare` can be called at most $N - 1$ times.
- For each call of `insert`, procedure `compare` can be called at most 30 times.
- $Q = 0$.

Test Group 4 (30 %)

- For the call of `init`, procedure `compare` can be called at most $N - 1$ times.
- For each call of `insert`, procedure `compare` can be called at most 2 times.
- For each call of `remove`, procedure `compare` can be called at most 2 times.

Sample grader

Download "ada-hw4-p3.zip"[6], extract and put them in the same directory of your code. Compile your code with the following command, assuming that your code is named `ada-hw4-p3.cpp`:

```
g++ -O2 -std=c++17 grader.cpp ada-hw4-p3.cpp -o ada-hw4-p3
```

We also have provided sample code (`ada-hw4-p3.cpp`), sample compile scripts (`compile_cpp.sh` for Linux and Mac OS, `compile_cpp.bat` for Windows) in the zip file.

The sample grader reads an array s of 32-bits unsigned integers indicating the scores of contests 0 to $N-1$ and Q operations. The sample grader reads input in the following format:

- line 1: N Q
- line 2: $s[0]$ $s[1]$ ... $s[n-1]$
- line $3+i (0 \leq i < Q)$: operation$[i]$: could be of the following two types:
  - insert p s: Insert a contest with score s at position p. The contest's ID will automatically become $N+j-1$ at the j^{th} call of `insert`.
  - remove p: Remove the contest at position p.

The output of sample grader is in the following format:

- line 1: $choice_{\text{init}}$ (t)
- line $2+i (0 \leq i < Q)$: $choice_i$ (t)

, where:

- $choice_{\text{init}}$: Choice returned by `init`.
- $choice_i$: Choice returned by operation$[i]$.
- t: the number of calls of the procedure `compare`.

All of the choices will be printed in the following format:

- k $c[0]$ $c[1]$ ... $c[k-1]$

Where k is the size of the choice and array c is the content of the choice.

Note:

- The sample grader will not help you judge the correctness of your answer except the format error.
- The sample grader will fail if it tries to compare two contests with the same score.

[6] http://w.csie.org/~b08902103/ADA/ada-hw4-p3.zip

Sample Input 1

```
3 2
1 4 2
insert 1 3
remove 3
```

Sample Output 1

```
2 1 2 (2)
1 2 (1)
0 (0)
```

Sample Input 2

```
3 3
3 2 1
insert 1 4
insert 2 5
remove 3
```

Sample Output 2

```
3 0 1 2 (2)
3 0 1 2 (2)
3 0 1 2 (2)
1 1 (2)
```

Sample Input 3

```
6 6
1 5 6 7 9 10
insert 1 8
insert 1 3
insert 6 4
remove 4
remove 7
remove 5
```

Sample Output 3

```
0 (5)
1 6 (1)
2 7 6 (0)
2 6 8 (2)
2 6 8 (2)
2 6 8 (2)
1 6 (0)
```

Hint

1. Try to reduce this problem to the **minimum vertex cover** problem, which has a 2-approximation algorithm as taught in class.
2. Do not write your own `main` function, and do not try to interact with stdin and stdout.
3. Feel free to use global variables.
4. Do not try to use complex data structures to maintain your array. Since the data size is small, you can spend linear time inserting or removing any element.
5. All you need to return are the IDs of the contests, not the positions.
6. For those who aren't familiar with `std::vector`, take a look at the document.
7. Do not try to steal any data from the grader. Similar behavior will be regarded as cheating and will receive heavy penalties. If you find weird behavior from the grading, please contact TA.

Problem 4 - $\mathfrak{P}$ Ring (Programming) (13 points)

Problem Description

Under the rule of King $\mathfrak{P}$, people live from hand to mouth in the kingdom of $\mathfrak{P}$. In the time of crisis, people rose to rebel and used a ring to seal King $\mathfrak{P}$.

To ensure everyone's safety, people decided to divide the ring into M segments and hide them at the end of the earth to prevent $\mathfrak{P}$ from resurrecting and slaughtering the people.

This ring is decorated by N symbols, and each can be represented by a value $a_1, a_2, \cdots, a_N$, in counterclockwise order. After dividing the ring into M segments (each consisting of some consecutive symbols), we can calculate each segment's risk value as follows.

Suppose the symbols within a segment can be represented as $b_1, b_2, \cdots, b_k$, the risk value of this segment will be

$$\sum_{i=1}^{k} b_i - \sum_{i=1}^{k-1} |b_{i+1} - b_i| + \sum_{i=1}^{k-2} b_{i+1} \times ((b_i \ \& \ b_{i+1}) \oplus (b_{i+1} \mid b_{i+2}) \oplus (b_i + b_{i+2}))$$

, where $x \ \& \ y$ means applying bitwise AND operation, $x \mid y$ means applying bitwise OR operation, and $x \oplus y$ means applying bitwise XOR operation to numbers x and y.

To reduce the risk, you want to minimize the sum of the risk values of the M segments. How should you divide the M segments?

Input

The first line contains two integers N, M, representing the number of symbols on the ring and the number of segments to be divided.

The following contains N space-separated integers $a_1, a_2, \cdots a_N$, indicating the symbols on the ring.

- $2 \leq M \leq N \leq 1000$
- $0 \leq a_i \leq 10^5$

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (20 %)

- $N \leq 25$.

Test Group 2 (30 %)

- $N \leq 100$

Test Group 3 (50 %)

- No additional constraint.

Output

Please print an integer representing the smallest sum of risk values of the M segments.

Sample Input 1

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

Sample Output 1

```
14
```

Sample Input 2

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

Sample Output 2

```
10
```

Sample Input 3

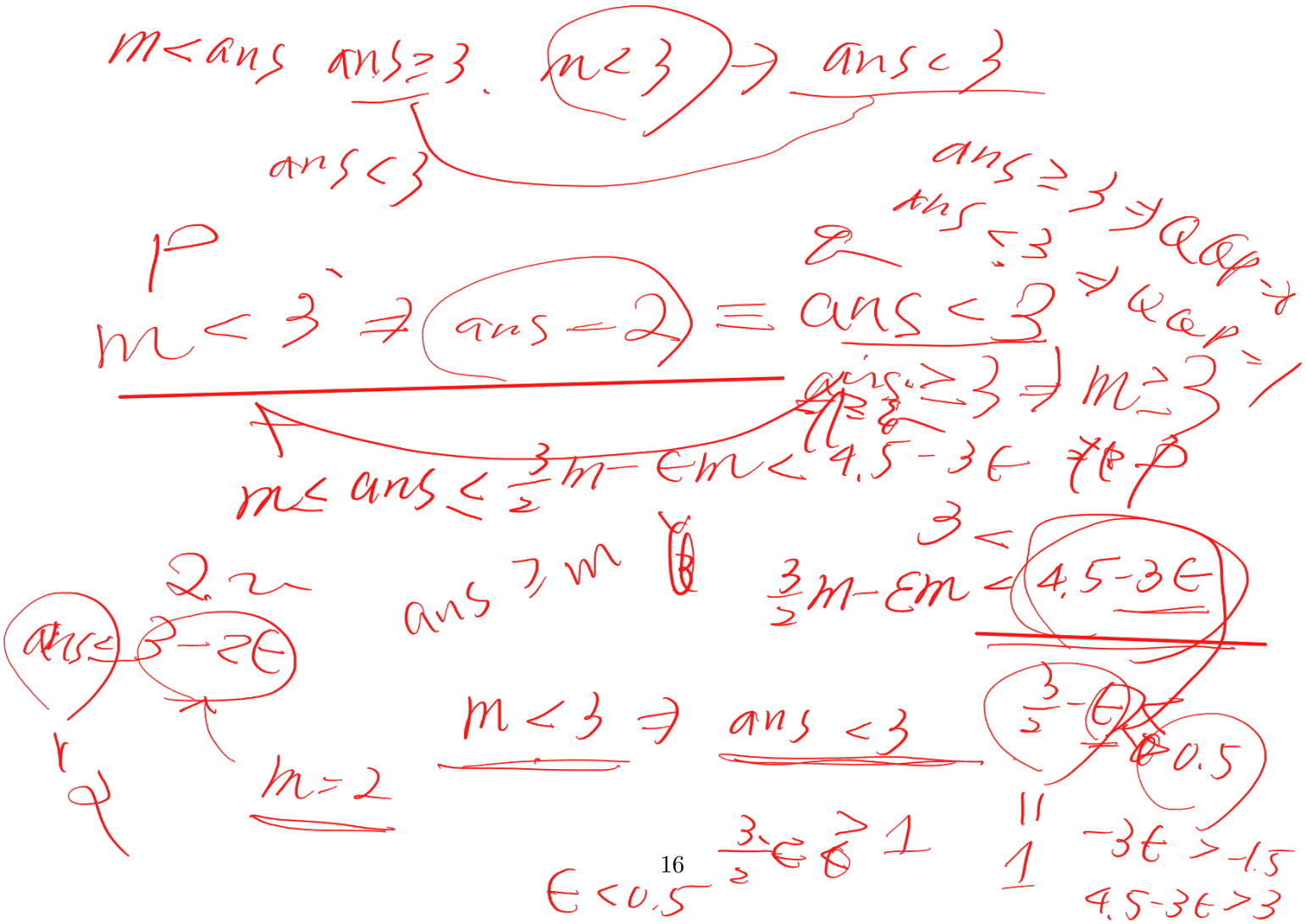
```
5 5
1 2 3 4 5
```

Sample Output 3

```
15
```

Hint

- The risk-value function has no specific meaning and you don't have to focus on it.
- You can check this sample code to see how to calculate the risk value.



Problem 5 - $\mathcal{P}$ NPC & Reduction (Hand-Written) (30 points)

First, here present some interesting problems. The definitions are given below:

JoJo's Bizarre Adventure Problem (JJBAP)

Given a sequence of integers $\mathcal{S} = (a_1, \cdots, a_n)$ with cardinality n and an integer T. The *JoJo's Bizarre Adventure Problem* seeks whether there is a subsequence of $\mathcal{S}$ whose sum equals T. This problem is known to be NP-complete.

Quintessential Quintuplets Problem (QQP)

Given a sequence of non-negative integers $\mathcal{S} = (a_1, \cdots, a_n)$ with cardinality n and $\sum_{i=1}^{n} a_i = U$. The *Quintessential Quintuplets Problem* seeks whether there is a subsequence of $\mathcal{S}$ whose sum equals $U/2$.

Dragon Ball Problem (DBP)

Given a collection of n balls $\mathcal{B}$ with weights $a_1, \cdots, a_n$ kilograms respectively, with constraint that $a_i \in [0, 1]$, i.e., a single ball's weight is at most 1 kilogram and at least 0 kilogram (yes, a_i may be 0, to make the following problem simpler). The *Dragon Ball Problem* seeks to partition balls into a minimum number of bins such that all the bins weigh at most 1 kilogram.

Recall that if problem Y can be reduced to problem X in polynomial time, we denote this by $Y \leq_p X$. It also means that X is at least as hard as Y because if you solve X, you solve Y. An immediate corollary is the transitivity of "$\leq_p$", i.e. if $Z \leq_p Y$ and $Y \leq_p X$ then we will have $Z \leq_p X$.

Reminder: It is good to practice how to write down proofs properly. **Do not use pseudocode to explain how your algorithm works.** Also, keep your proof and algorithm as readable as possible. If you want to make sure whether you can get full credit with your solution, you are encouraged to contact TAs via email or discuss with TAs during TA hours.

You may assume the transitivity of "$\leq_p$" and the NP-completeness of $JJBAP$ in the following problems.

For each "show that $Y \leq_p X$", please

- Show the correctness of your reduction. In particular, if Y and X are decision problems, f is a transformation between the instances of Y and those of X, L is an instance of Y and $f(L)$ is an instance of X, you need to show that the answer to L is "yes" if and only if the answer to $f(L)$ is "yes".
- Provide a polynomial-time reduction algorithm for f and briefly explain why your algorithm meets the time requirement. In particular, if f is your transformation, explain why it takes polynomial time to complete the transformation. This can be done within two sentences in the following problems. If your reduction algorithm is too complicated to explain in one or two sentences, it is probably wrong.

Here begins the problem section:

1. **(6 points)** Consider a variant of $JJBAP$: given a sequence of **non-negative** integers $\mathcal{S} = (a_1, \cdots, a_n)$ with cardinality n and an integer T, and determine whether there is a subsequence of $\mathcal{S}$ whose sum equals T. Let's denote this problem $JJBAP_+$.

In fact, $JJBAP$ and $JJBAP_+$ are equivalent with respect to polynomial-time reduction, i.e., $JJBAP_+ \leq_p JJBAP$ and $JJBAP \leq_p JJBAP_+$. An immediate consequence is that $JJBAP_+$ is also NP-complete. It is straightforward to see that $JJBAP_+ \leq_p JJBAP$. Please show **the other direction**.

Hint: you wish the transformed a_i are all non-negative, so you can solve $JJBAP$ by solving $JJBAP_+$ given the transformed problem instance. Let $L = (a_1, \cdots, a_n; T)$ be an instance of $JJBAP$, then consider a transformation f from instances of $JJBAP$ to those of $JJBAP_+$, in the sense that

$$f(a_1, a_2, \cdots, a_n; T) = (a_1 + K, a_2 + K, \cdots, a_n + K, \underbrace{K, \cdots, K}_{n\ K\text{'s}}; T + nK)$$

How do you choose K? And, when do you need to do the transformation? Don't forget to justify that the answer to the problem instance L is "yes" if and only if the answer to $f(L)$ is "yes".

More Hint: This problem is considered to be tricky. $-\min a_i$ is not a good enough choice for K. Absolute value and its triangle inequality may be useful for this problem.

2. **(0 points)** Show that $JJBAP \leq_p QQP$ using the result in the previous problem. Mimic the proof given in the lecture slides. This problem is just an extra exercise for you and will not be graded.

   Hint: add some integers to the original problem instance.

3. **(6 points)** ~~Show that $QQP \leq_p DBP$.~~ Write down the corresponding decision problem of DBP and let's call this problem $DDBP$. Show that $QQP \leq_p DDBP$, and deduce that $QQP \leq_p DBP$.

4. **(3 points)** Briefly explain why QQP and ~~DBP~~ $DDBP$ are both NP. Deduce that QQP and ~~DBP~~ $DDBP$ are NP-complete from the previous results.

5. **(6 points)** If $P \neq NP$, show that there is no polynomial time $(3/2-\epsilon)$-approximation for DBP, where $\epsilon > 0$ is any small positive number.

The following problems will guide you to give an example that "small additional restrictions may change a problem a lot".

We say a collection of balls $\mathcal{B}$ suffices (c, m)-*Kuan's first condition* if each ball in $\mathcal{B}$ is at least c kilogram and there are only m different types of weights in $\mathcal{B}$ for some $c > 0$ and $m \in \mathbb{N}$. Fix c and m, and we call the DBP problem with (c, m)-Kuan's first condition as (c, m)-Kuan's DBP. Here you will show that (c, m)-Kuan's DBP ~~is in the complexity class P~~ can be solved in polynomial time.

6. **(3 points)** For example, there are 5 balls which weigh $\pi/4$, $1/2$, $1/2$, $1/3$, $1/4$ respectively, then there are totally 9 choices of the balls to form a valid content of a single bin. The content of a single bin may be one of the following form:

   - $\{\}$, i.e., it's an empty bin.
   - $\{\pi/4\}$
   - $\{1/2\}$
   - $\{1/3\}$
   - $\{1/4\}$
   - $\{1/2, 1/2\}$
   - $\{1/2, 1/3\}$

- $\{1/2, 1/4\}$
- $\{1/3, 1/4\}$

Note that $\{\pi/4, 1/2, 1/4\}$, $\{1/2, 1/2, 1/3\}$ and $\{1/2, 1/3, 1/4\}$ are not valid contents of a single bin, since they all exceed the weight limit.

Find an upper bound of "the number of choices of the balls that form a valid content of a single bin", and represent it with m and c. Your answer should be an integer independent of n.

Hint: Consider the number of combinations. Note that there may be balls with identical weights, especially as $n \to \infty$.

7. **(3 points)** Denote the upper bound you found in the previous problem as M. Now given k indistinguishable bins, you can compute the number of ways to put all the balls inside these k bins, denote this number $\Gamma(k)$. Show that $\Gamma(k)$ is bounded-above by $C_M k^M$ for some C_M, which is a positive constant independent of choice of k and may depend on M. You can either "justify the existence of C_M" or "write down your favorite C_M and simply explain why your C_M meets the requirement". Again, the sum of weight of the content in any single bin should not exceed 1 kilogram.

   Remark: In fact, it is equivalent to show $\Gamma(k) = O(k^M)$ with respect to the variable k. If you use this fact, you also need to write down the reason why they are equivalent.

8. **(3 points)** Based on the previous results, provide an $O(n^{M+1})$ time algorithm for (c, m)-Kuan's DBP~~, where L is some non-negative integer independent of n~~. Analyze the time complexity of your algorithm and justify the asymptotic bound rigorously as you did in Homework #1. Deduce that ~~(c, m)-Kuan's DBP~~ (c, m)-Kuan's $DDBP$, the decision version of (c, m)-Kuan's DBP, is in the complexity class P.

(c, m)-Kuan's second condition: You should go to bed at 0037

Problem 6 - ♈ Merry Christmas (Hand-Written) (20 points)

*Please see the class slides for the definition of the **Traveling Salesman Problem (TSP)** and what it means to have a **TSP** whose edge costs satisfy the triangle inequality.*

In class, Ada has learned a 2-approximation algorithm using **Minimum Spanning Tree (MST) algorithms** to solve **TSP** on a graph **G** whose edge costs satisfy the triangle inequality. Now, Ada wants to design a better algorithm with a smaller approximation ratio. Can you help?

1. (2pt) Describe the sufficient and necessary condition for G to contain an Eulerian cycle.
2. (4pt) Given that T is a tree and $V' = \{\text{vertex } v \text{ with odd degree } |\forall v \in T\}$, prove that $|V'|$ is even.

In graph theory, a perfect matching in a graph is a matching that covers every vertex of the graph. More formally, given a graph $G = (V, E)$, a perfect matching in G is a subset M of E, such that every vertex in V is adjacent to exactly one edge in M.

3. (6pt) Let $V' \subseteq V$, such that $|V'|$ is even, and let M be a minimum cost perfect matching on V'. Prove that $cost(M) \leq OPT/2$.(this OPT is the TSP problem's OPT)
4. (8pt) Given a function **Oracle** which can find a minimum cost perfect matching of V in polynomial time, please design a 3/2-approximation algorithm to solve this problem and show that
   (a) It is 3/2-approximation.
   (b) The time complexity of your algorithm is polynomial.

Appendix

GLPK Installation

For some common Linux distributions or MacOS (you may need to install Homebrew[7] using the installation command in its official website) user, you may install the GLPK package from their official package repositories. For example:

```
# Arch Linux (GLPK 5.0):
sudo pacman -Syu glpk
# Ubuntu/Debian (GLPK 4.65):
sudo apt install libglpk-dev
# MacOS (GLPK 5.0):
brew install glpk
# Fedora/CentOS (GLPK 4.xx):
sudo yum install glpk
# or for newer version of Fedora/CentOS (GLPK 5.0):
sudo dnf install glpk
```

Note that for MacOS users that uses GNU g++ instead of clang++ (default g++), and install the GLPK package from Homebrew, since the GNU g++ does not use the system's library that includes the installed GLPK package, you may need to provide additional compilation arguments to specify the path to find the GLPK package. For example, you may try to use the below compilation command (assume your `g++` is GNU g++):

```
g++ -std=c++17 -O2 -oexample example.cpp -I /usr/local/include \
-L /usr/local/lib -lglpk -Wall
```

For Windows user, you may install the GLPK package from Cygwin[8]:

1. Download the Cygwin installer[9] and execute the installer. Below assume you install the cygwin at the default directory `C:\cygwin64`.
2. Follow the default installation settings to download from an arbitrary mirror.
3. At the package selection page, change the view to "Full" and install the following package:
   - Search for `g++` and select the package `gcc-g++` package with version 10.2.0-1.
   - Search for `glpk` and select both package `glpk` and `libglpk-devel` with version 5.0-1.
4. After the installation, copy the source code and the header file to the `C:\cygwin64\home` directory.
5. Run the `C:\cygwin64\Cygwin` batch file, and you should see a terminal (simple linux shell).
6. Type `ls` into the terminal and you should see the files you put in the previous step (step 4).
7. At last, you can simply run the compilation command described above to compile your program, and run it with `./example` command.

[7] https://brew.sh/index_zh-tw
[8] https://www.cygwin.com/
[9] https://www.cygwin.com/setup-x86_64.exe

For other Unix-based operating systems which are able to run `curl`, `tar`, `gcc`, `g++`, `make` commands, you can use the following script to build and install GLPK package from source:

```
curl -o glpk-5.0.tar.gz https://ftp.gnu.org/gnu/glpk/glpk-5.0.tar.gz
tar -xzvf glpk-5.0.tar.gz && cd glpk-5.0
./configure && make && sudo make install
# cd to the directory that contains your source codes and the given header
g++ -std=c++17 -O2 -oexample example.cpp -lglpk -Wall
LD_LIBRARY_PATH=/usr/local/lib ./example
# you can use ‘sudo make uninstall‘ command in the glpk-5.0 directory
# to uninstall the GLPK package
```

Moreover, if you do not have root access (or do not want to install the package globally, you can use the following script to install it in a rootless location:

```
# assume your current working directory is $HOME,
# the glpk package will be installed at $HOME/glpk-5.0/build
curl -o glpk-5.0.tar.gz https://ftp.gnu.org/gnu/glpk/glpk-5.0.tar.gz
tar -xzvf glpk-5.0.tar.gz && cd glpk-5.0
./configure --prefix=$PWD/build && make && make install
# cd to the directory that contains your source codes and the given header
g++ -std=c++17 -O2 -oexample example.cpp -I $HOME/glpk-5.0/build/include \
-L $HOME/glpk-5.0/build/lib -lglpk -Wall
LD_LIBRARY_PATH=$HOME/glpk-5.0/build/lib ./example
```

If you still have any problem with the GLPK package or failed to compile or run the programs locally, feel free to email TAs for help as soon as possible. Besides, to let us figure out the problem sooner, please provides the environment (operating system), the problem detail you encountered, and maybe some screenshot or error message you saw when you are asking for help.