Homework #4 (Programming)

Due Time for Problem 3, 4 and 5: 2022/12/30 23:59
Due Time for Problem 6: 2023/1/7 23:59
Contact TAs: `ada-ta@csie.ntu.edu.tw`

Instructions and Announcements

- There are **four programming 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.
- **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 of your solution to that problem. You may get zero point 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 3 - LongLong (Programming) (10 points)

Problem Description

"You should never `#define int long long` in ADA course", said the Great $\tau m \tau 514$.

In this problem, we want you to find the longest simple path from S to T. You are given a **unidirectional** graph with N vertices and M edges with non-negative integer weights. Please output the maximum weight among all simple paths from S to T, where S, T are starting and ending vertices, respectively. You can provide any path with that maximum weight.

- The weight of a simple path is defined as the sum of all weights of edges on the path.
- A path is called simple if it has no repeated vertices.

Input

The first line of the input contains two space-separated positive integers N, M. The i-th of next M lines contains three space-separated integers u_i, v_i, w_i, indicating an edge from u_i to v_i with a weight w_i.

- $S = 1, T = N$
- $2 \leq N \leq 80$
- $1 \leq M \leq 90$
- $1 \leq u_i, v_i \leq N$ where $u_i \neq v_i$.
- $0 \leq w_i \leq 2^{20}$

Output

If you cannot reach T from S, please just output -1.

Otherwise, the first line of your output should be a non-negative integer representing the maximum weight. In the second line you should output a binary string X with length M. $X_i \in \{0, 1\}$ indicates whether you chose the i-th edge in your optimal simple path.

If there is more than one path with that maximum weight, you can output any path with that maximum weight. Note that the path needs to be simple, starting from S and ending at T.

Sample Input 1

```
5 10
1 3 906394
3 2 548103
2 5 745834
1 2 1
3 1 0
1 4 0
3 1 1
1 5 1
2 1 1
3 2 1
```

Sample Output 1

```
2200331
1110000000
```

Sample Input 2

```
6 7
1 2 1048576
2 6 1048576
1 3 1
3 4 114
4 5 514
5 3 228922
5 6 1
```

Sample Output 2

```
2097152
1100000
```

Sample Input 3

```
10 20
4 8 127
8 7 307
7 1 765
1 4 331
10 9 1009
9 6 715
6 10 665
5 2 107
3 8 908
2 3 402
3 5 467
5 2 177
2 4 527
8 6 295
7 2 1010
7 6 723
1 8 378
3 9 605
7 9 450
6 8 561
```

Sample Output 3

```
4162
11010110010000100100
```

Test Group 1 (20 %)

- The graph is an acyclic graph. That is, there are no cycles in the graph.

Test Group 2 (80 %)

- No additional constraint

Hints

1. 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.)
2. You are welcome to use any heuristic method to solve this problem. Happy hacking.
3. There will be some graph visualizing Python scripts here [1].
4. In sample 2, you should choose the edges $[(1,2),(2,6)]$ rather than $[(1,2),(2,6),(3,4),(4,5),(5,3)]$. The latter contains an isolated cycle. To prevent this, you probably would like to search for "Miller, Tucker, and Zemlin" technique.

Linear Programming Solver

The GLPK (GNU Linear Programming Kit)[2] 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[3] for usage.
It is easier for you to use a packed version of the GLPK tools. The full header file can be downloaded here[4]. You can include it via `#include "ypglpk.hpp"`.
There are three functions defined in the namespace "ypglpk". You can follow the instruction below for usage. All the std::vector below are 0-based. Note that if 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 $Ax \le b, x \in \mathbb{R}^n$.
- 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>()`).

[1] https://www.csie.ntu.edu.tw/~b10902082/ada22hw4material/
[2] https://www.gnu.org/software/glpk/
[3] http://most.ccib.rutgers.edu/glpk.pdf
[4] https://www.csie.ntu.edu.tw/~b10902082/ada22hw4material/ypglpk.hpp

```
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<int> &vartype);
```

- The function `mixed_integer_linear_programming(A,b,c,vartype)` is used to solve the mixed integer linear programming problem, which can restrict some structural variables to be integers.
- Parameters: $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n, \text{vartype} \in \{\texttt{GLP\_BV}, \texttt{GLP\_IV}, \texttt{GLP\_CV}\}^n$ 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 $\text{vartype}_i = \texttt{GLP\_IV}, x_i \in \mathbb{Z}$, $\forall j$ with $\text{vartype}_j = \texttt{GLP\_BV}, x_j \in \{0, 1\}$. $\texttt{GLP\_BV}, \texttt{GLP\_IV}$ and `GLP_CV` stands for binary variable, integer variable and continuous variable, respectively.
- 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>()`).
- In mixed integer linear programming, the time complexity primarily depends on the number of integral type variables.

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

- 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[5] 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.

[5] https://www.csie.ntu.edu.tw/~b10902082/ada22hw4material/example.cpp

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"):

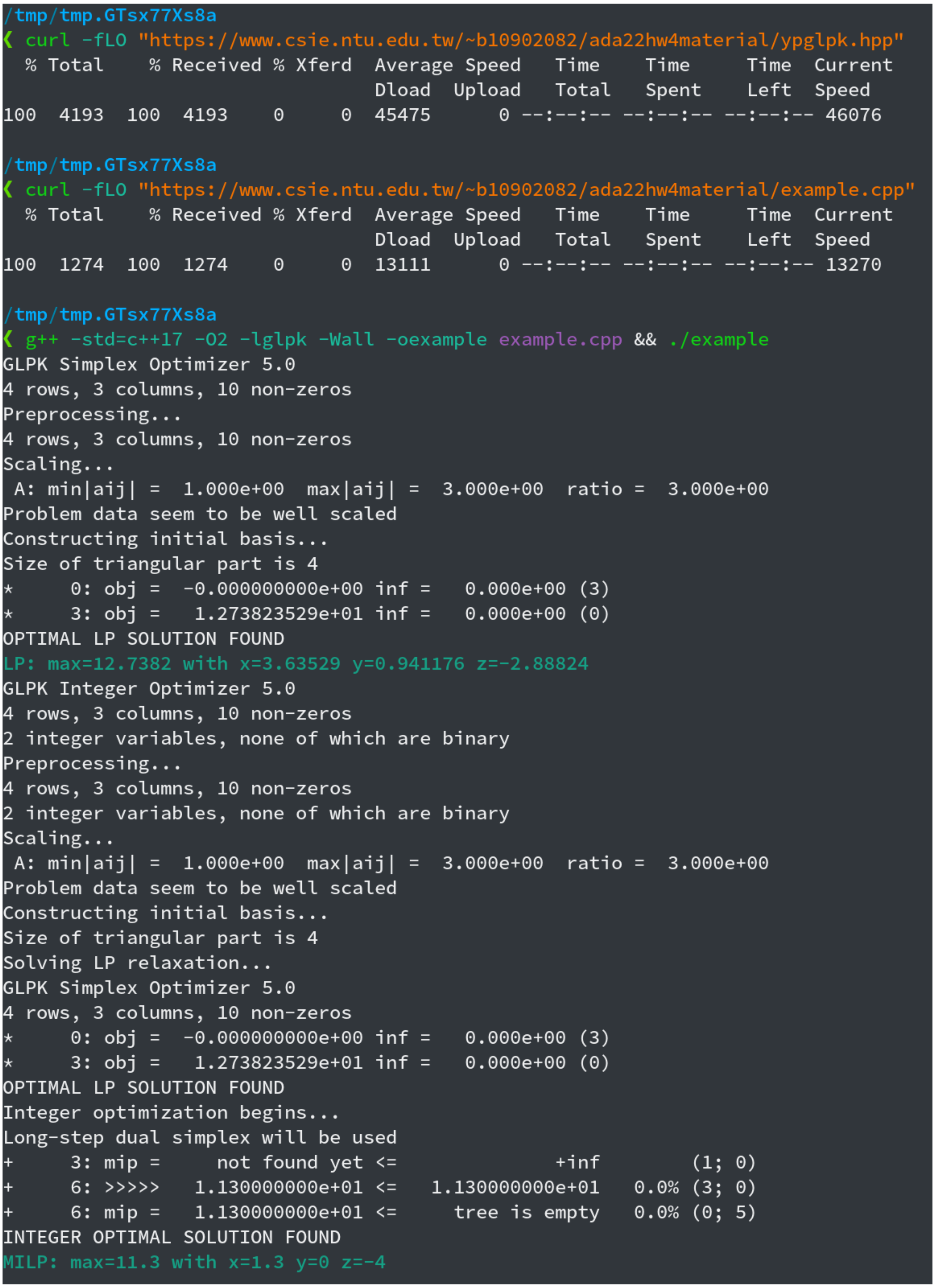
```
g++ -std=c++17 -O2 -oexample example.cpp -lglpk -Wall
```

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 -fLO "https://www.csie.ntu.edu.tw/~b10902082/ada22hw4material/example.cpp"
curl -fLO "https://www.csie.ntu.edu.tw/~b10902082/ada22hw4material/ypglpk.hpp"
g++ -std=c++17 -O2 -oexample example.cpp -lglpk -Wall
./example
```

The expected output is shown in Figure 1.



```
/tmp/tmp.GTsx77Xs8a
❮ curl -fLO "https://www.csie.ntu.edu.tw/~b10902082/ada22hw4material/ypglpk.hpp"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  4193  100  4193    0     0  45475      0 --:--:-- --:--:-- --:--:-- 46076

/tmp/tmp.GTsx77Xs8a
❮ curl -fLO "https://www.csie.ntu.edu.tw/~b10902082/ada22hw4material/example.cpp"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1274  100  1274    0     0  13111      0 --:--:-- --:--:-- --:--:-- 13270

/tmp/tmp.GTsx77Xs8a
❮ g++ -std=c++17 -O2 -lglpk -Wall -oexample example.cpp && ./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
```

Figure 1: Sample output of the example.cpp program

Problem 4 - Choose (Programming) (10 points)

Problem Description

Three-mouthed sheep needs to do N tasks with zir two identical machines. To solve a task, ze assigns it to one of the machines. Since the machines are identical, assigning any specific task τ to machine 1 or machine 2 takes the same time. However, before having a task done, ze is not able to know the exact time required to finish it. Instead, ze can only

- know which of two specific tasks τ_1, τ_2 takes longer time, and
- assign a task τ to a machine m and the machine will show its earliest available time.

To prevent any of zir machines from overworking, three-mouthed sheep would like to partition the tasks as even as possible. That is, ze tends to make $T = \max(t_1, t_2)$, where t_i denotes the total working duration of the i-th machine, as small as possible.

If three-mouthed sheep knew the time required for each task beforehand and partitioned the tasks optimally, $\max(t_1, t_2)$ equals to T_{opt}. Merely with the two operations mentioned above, can you come up with a ρ-approximation algorithm such that $T \leq \rho \cdot T_{\text{opt}}$?

Implementation details

Include `"choose.h"` in the first line of your code, and then you should implement the following procedure:

```
void schedule(int N)
```

- N: the number of tasks Three-mouthed sheep needs to do.
- This procedure is called exactly once by the grader. All of the N tasks should be scheduled after the procedure completes.

The above procedure can make calls to the following procedures:

```
int compare_task(int x, int y)
```

- x: the first task ID you want to compare with.
- y: the second task ID you want to compare with.
- Both of the compared task ID should satisfy $1 \leq x, y \leq N$.
- The procedure will return one of the following values:
  - -1 if $t_x < t_y$.
  - 0 if $t_x = t_y$.
  - 1 if $t_x > t_y$.
- The grading program will record the number of calls for this procedure, which is related to the condition of getting `Accepted`.

- The time complexity of the procedure is $O(1)$.

```
int assign_task(int m, int x)
```

- m: The machine ID you want to assign to.
- x: The task ID you want to assign with.
- The machine ID should satisfy $1 \leq m \leq 2$.
- The task ID should satisfy $1 \leq x \leq N$, and task x should not have been scheduled before.
- The procedure will schedule task x to machine m. After the assignment, it will return the earliest available time of machine m.
- The time complexity of the procedure is $O(1)$.

Examples

Consider a scenario in which three-mouthed sheep has 2 tasks. The procedure `schedule` is called in the following way:

```
schedule(2)
```

If you want to assign the task having a smaller process time to the first machine, and a bigger one to the second machine. In the `schedule` procedure, you can first call `compare_task(1, 2)` to know which of the two tasks takes longer time.

Let's call the return value of `compare_task(1, 2)` by x. If $x \leq 0$, you can call `assign_task(1, 1)`, `assign_task(2, 2)` to achieve you goal; otherwise, you can call `assign_task(1, 2)`, `assign_task(2, 1)` to achieve you goal.

Moreover, if you take the return values of the calls of `assign_task`, you can know the process time of task 1 and task 2 since `assign_task` will return the assigned machine's earliest available time.

You can see more details in `choose.cpp` provided in the attached file `"choose.zip"`[6].

Constraints

- $1 \leq N \leq 500$
- $1 \leq t_i \leq 500$
- You can only call `compare_task` at most N^2 times.

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (30 %)

- $\rho = \frac{3}{2}$

Test Group 2 (70 %)

- $\rho = \frac{7}{6}$

[6] http://w.csie.org/~b10902084/choose.zip

Sample grader

Download `"choose.zip"`[7], extract and put them in the same directory of your code. Compile your code with the following command, assuming that your code is named `choose.cpp`:

```
g++ -O2 -std=c++17 grader.cpp choose.cpp -o choose
```

We also have provided sample code (`choose.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 input in the following format:

$$\begin{array}{l} a \;\; b \;\; N \\ t_1 \;\; t_2 \; \cdots \; t_N \end{array}$$

Here,

- $\rho = \dfrac{a}{b}$ is the approximation ratio.
- N is the number of tasks.
- $t_1, t_2, \ldots, t_N$ are 32-bit unsigned integers, where t_i indicates the processing time of task i.

The output of the sample grader is in the following format:

$$S$$

S is the output message, and could be `Accepted` or `Wrong Answer` optionally with detailed information.

Sample Input 1

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

Sample Output 1

```
Accepted with approximation ratio 1.2000.
```

Sample Input 2

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

Sample Output 2

```
Accepted with approximation ratio 1.0000.
```

Hint

1. Do not write your own `main` function, and do not try to interact with `stdin` and `stdout`.
2. Feel free to use global variables.
3. Do not try to steal any data from the grader. Similar behavior may be regarded as cheating and may receive heavy penalties. If you find weird behavior from the grading, please contact the TAs.

[7] http://w.csie.org/~b10902084/choose.zip

Problem 5 - Three-More-Quotient Ring (Programming) (15 points)

Problem Description

There is a magical ring in President Cheng's hometown called Three-More-Quotient Ring. President Cheng wants to split Three-More-Quotient Ring into k parts to maximize their total magic value.

Three-More-Quotient Ring is circularly welded of n rare medals. That is, the i-th medal is welded with the $(i+1)$-th medal and the $(i-1)$-th medal, where the 0-th and the $(n+1)$-th medals are considered to be the n-th and the 1st medals, respectively.

When splitting the Three-More-Quotient Ring into k parts, each rare medals should be in the same part, and each part should be a non-empty continuous fragment of the Three-More-Quotient Ring. President Cheng will tell you the magic value of every possible continuous fragment of the Ring. Write a program to tell him the maximum of the total magic values of all parts.

Note that splitting the Three-More-Quotient Ring into one part means cutting the Ring at the point between the i-th and the $(i+1)$-th medals for some i. As a remark, there are n ways to split Three-More-Quotient Ring into one part, and different ways may lead to different results.

Input

The first line of the input consists an integer n – the number of rare medals of Three-More-Quotient Ring.

Each of the following n lines has n integers – the j-th integer of the i-th line, denoted by v_{ij}, is the magic value of the continuous fragment of Three-More-Quotient Ring that starts from the i-th medal and ends at the $(j-1)$-th medal.

Note that the fragment that starts from the i-th medal and ends at the $(j-1)$-th medal is the fragment that consists of the $\begin{cases} i, i+1, i+2, \ldots, j-1 & \text{, if } i < j \\ i, i+1, i+2, \ldots, n, 1, 2, \ldots, j-1 & \text{, if } i \geq j \end{cases}$-th medals.

Constraints

- $1 \leq n \leq 400$
- $1 \leq v_{ij} \leq 10^9$

Test Group 0 (0 %)

- Sample input

Test Group 1 (40 %)

- $n \leq 100$

Test Group 2 (60 %)

- No additional constraint

Output

Output n integers, where the i-th integer is the maximum of the total magic value of all parts when splitting Three-More-Quotient Ring to i parts.

Sample Input 1

```
4
13 1 5 9
10 14 2 6
7 11 15 3
4 8 12 16
```

Sample Output 1

```
16 15 13 10
```

Notes

In sample input 1, we can split Three-More-Quotient Ring to k parts like the following:

- $k = 1 : [1234], [2341], [3412], [4123]$, and the magic values are 13, 14, 15, **16**, respectively.
- $k = 2 : [1, 234], [2, 341], [3, 412], [4, 123], [12, 34], [23, 41]$ and the magic value are 11, 13, **15**, 13, 12, 14, respectively.
- $k = 3 : [12, 3, 4], [23, 4, 1], [34, 1, 2], [41, 2, 3]$, and the magic values are 12, 11, 10, **13** respectively.
- $k = 4 : [1, 2, 3, 4]$, and the magic value is **10**.

Problem 6 - SPlAnDAr (Programming) (15 points)

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

Problem Description

After the midterm exam, all students hold a party to celebrate. -10^{11}, who gets the 1st place in the exam, loves to play board games. He is especially good at playing *Splender*.

During the party, -10^{11} outperforms all students in *Splender*. However, he is not satisfied with the result, so he posts his strategy as an ADA programming homework.

Following are the rules of *Splender*.

Rules

Notice that the rules may be different from the real-world *Splender*. Please follow the rules defined in the following if there is any conflict. If you have any question about rules, please email TAs.

Components

- 20 gem tokens
  - 4 Emerald tokens (green)
  - 4 Diamond tokens (white)
  - 4 Sapphire tokens (blue)
  - 4 Onyx tokens (black)
  - 4 Ruby tokens (red)
- 4 Gold Joker tokens (yellow)
- 90 Development cards
  - 40 Level 1 cards
  - 30 Level 2 cards
  - 20 Level 3 cards

Setup

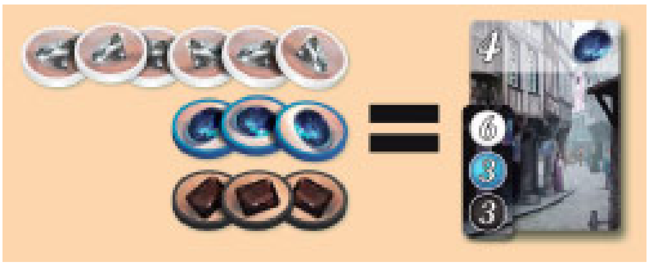
Separate development cards into three decks according to their levels. Shuffle each development card deck separately, and then place them in a column in the middle of the table in an increasing order of level from top to bottom.

Then reveal 4 cards from each level.

The Development Cards

To win prestige points, the players must purchase development cards. These cards are visible in the middle of the table and can be purchased by all players during the game.

The developments in hand are the cards the players reserve throughout the game, which may only be purchased by the players holding them.



Example: The player who purchases this card wins 4 prestige points.

Owning this card allows the player to benefit from a blue bonus. To purchase this card, the player must spend 6 white tokens, 3 blue tokens, and 3 black tokens.

Notice that a player can not have 6 white tokens in the same time because there are only 4 white tokens in the game. As a result, this card can only be obtained by bonus.

Game Play

This game is for two people—you and -10^{11}. you always go first, and -10^{11} always goes second.

For each turn, a player must choose to perform only one of the following three actions:

- Take 3 gem tokens of different colors or 2 gem tokens of the same color.
- Reserve 1 development card and take 1 gold token (joker).
- Purchase 1 face-up development card from the middle of the table or a previously reserved one.

If a player wants to make an illegal move, he loses immediately.

Selecting Tokens

If a player have more than 10 tokens (including jokers), he loses immediately.

Note: a player can only choose to take 3 gem tokens of different colors or 2 gem tokens of the same color, meaning that he can never choose to take 2 gem tokens of different colors or 1 gem token only.

Reserve a development card

To reserve a card, a player simply takes a face-up development from the middle of the table.

The reserved cards are kept in hand and cannot be discarded. Players may not have more than 3 reserved cards in hand, and the only way to get rid of a card is to buy it.

As mentioned above, if a player has more than 3 reserved cards, he loses immediately.

Reserving a card is also the only way to get a gold token (joker).

However, if a player has 2 or fewer reserved cards and 10 tokens, and there are some gold tokens left, reserving a card would lead to more than 10 tokens and make the player lose, because taking tokens is a must once he reserves cards. Besides, if there is no gold token left, reserving a card also makes him lose the game because he is unable to get a gold token.

Buying a development card

To purchase a card, a player must spend tokens indicated on the card. A joker token can replace any color. The spent tokens (including any jokers) are returned to the middle of the table.

A player may purchase one of the face-up development cards in the middle of the table or a card in his hand reserved from the previous turn.

When a player pays for the card, he can consume gem tokens only if the bonuses are not enough, and he can consume joker tokens only if the bonuses and the gem tokens are not enough.

Note: when a development card from the middle of the table is acquired or reserved, it must immediately be replaced by a card of the same level.

During the whole game, there must be 4 face-up cards of each level (unless the deck in question is empty, in which case the empty spaces also remain empty).

Bonuses

The development cards acquired from previous turns provide bonuses for the player who buys them, which give discounts on purchasing new cards. The bonus of a given color is equal to a token of that color.

Thus, if a player has 2 blue bonuses and wants to purchase a card that costs 2 blue tokens and 1 green token, the player only needs to spend 1 green token.

If a player has enough development cards (and therefore bonuses), they can even purchase a card without spending any tokens.

End of the Game

When a player reaches 15 prestige points, he wins immediately.

Scoring

There are multiple test groups. In each test group, -10^{11} makes decisions in different strategies. **Winning one game grants you 2 pts. There are 110 pts in total, but the final score will be capped at 100.**

Group 0 (2 pts)

-10^{11} always chooses an illegal move.

Group 1-6 (12 pts)

-10^{11} chooses a legal move randomly.

Group 7-12 (12 pts)

-10^{11} purchases development card only if he cannot take gem tokens. Besides, he does not reserve card.

Group 13-18 (12 pts)

-10^{11} intends to purchase low-level development card. Besides, he does not reserve cards.

Group 19-24 (12 pts)

-10^{11} intends to purchase high-level development card. Besides, he does not reserve cards.

Group 25-34 (20 pts)

-10^{11} make the decision using his strong mind. However, he pretend to be weak. Therefore, in the first **10** moves, there is a **50%** chance for him to make a random move.

Group 35-44 (20 pts)

-10^{11} make the decision using his strong mind. However, he is not very careful. Therefore, in the first **10** moves, there is a **20%** chance for him to make a random move.

Group 45-54 (20 pts)

-10^{11} make the decision using his strong mind.

Implementation details

Each development card is represented as the following structure:

```
struct card{
    int id;
    int cost[5];
    int gem;
    int score;
};
```

- `id` indicates the card, which can be used by some functions or struct below.
- `cost` indicates the gem for purchasing it.

- `gem` indicates the bonus.
- `score` indicates the prestige points.

Each move (both yours and -10^{11}'s) is represented as the following structure:

```
struct move{
    int type;
    int card_id;
    int gem[5];
};
```

- `type` indicates the action type. `type` $= 0$ indicates no previous round (you can check the below function about the situation). `type` $= 1$ indicates taking gems. `type` $= 2$ indicates purchasing a development card. `type` $= 3$ indicates reserving a development card.
- `card_id` indicates the card id the player purchases or reserves. It is used only when `type` $= 2$ or `type` $= 3$.
- `gem[5]` indicates the number of tokens the player takes. It is used only when `type` $= 1$.

You need to implement the following procedure:

```
void init (std::vector<struct card> stack_1,
           std::vector<struct card> stack_2,
           std::vector<struct card> stack_3)
```

In the beginning, the judge would call this function to provide you the content of stack. Index 0 of the `vector` is on the top. During the game, you would never get the board status, because it is determined by the initial state and both players' actions. There are multiple games within a single test case. Please properly reset all your variables whenever `init()` is called.

```
struct move player_move (struct move server_move)
```

In each game, `player_move` will be called first with `server_move.type` $= 0$. You should return the first move you want to make. Every subsequent call of the function denotes the action 10^{-11} takes. You should return your next move.

It is guaranteed that the game hasn't finished yet when calling this function.

Note: If a player can't take any legal action in his round, then any return value of the function would be considered as Wrong Answer, but the server would not remind you.

Local testing tool & Template

We've provided a local testing tool for you. The tool will connect to our server (so it needs network connectivity) and allows you to play against the same strategy on the judge. Hence, please submit your solution to the judge after it runs correctly at local.

Please download splender.zip[8], which contains the following two files:

[8] http://w.csie.org/~b09902096/ADA/splender.zip

```
splender.cpp
```

This is the template file for you to work on, which is also the file to submit.

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.

```
splender.h
```

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

It will ask for the test group and the number of games you want. 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:

- `SPLENDER_DEBUG`: The tool will output the state of the game board if this flag is defined.
- `SPLENDER_INTERACTIVE`: The tool will let you (instead of -10^{11}) 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 -10^{11}'s thinking time). 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.

Notes

1. The time limit for this problem is 50 seconds per testcase (1 games). However, you can only submit 1 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 behaviors 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.

Appendix

GLPK Installation

For some common Linux distributions or MacOS (you may need to install Homebrew[9] 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[10]:

1. Download the Cygwin installer[11] 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.

[9] https://brew.sh/index_zh-tw
[10] https://www.cygwin.com/
[11] 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.