Second Project ADS 2025/2026

Second Project ADS 2025/2026

Dunder Malloc (I)

In one of the branches of the great ***Dunder Malloc***, deep in the Povsylvania of Trentino, there is an air of significant change.

Among colleagues, who never lack a passion for gossip, news of a **new director** has been spreading for weeks.

Dunder Malloc (II)

In a world dominated by the best memory allocators, the company has always been on the brink of failure.

The employees of ***Dunder Malloc***, overcome by discouragement, work in an increasingly listless and inefficient manner, wasting precious time and skills.

The New Director

But finally today is the big day: the new director **Albert Scott** enters the company, ready and convinced that he can change things.

For Dr. A. Scott, the watchword is **optimization**.

"Can we do better than this?" is his famous motto.

Who works at Dunder Malloc?

The new director, a lover of any gossip that can keep his "hard work" on hold, calls his collaborators into his office one by one for an interview.

He asks about their **skills**, their **hourly availability**, and the **cohesion** with each of their colleagues.

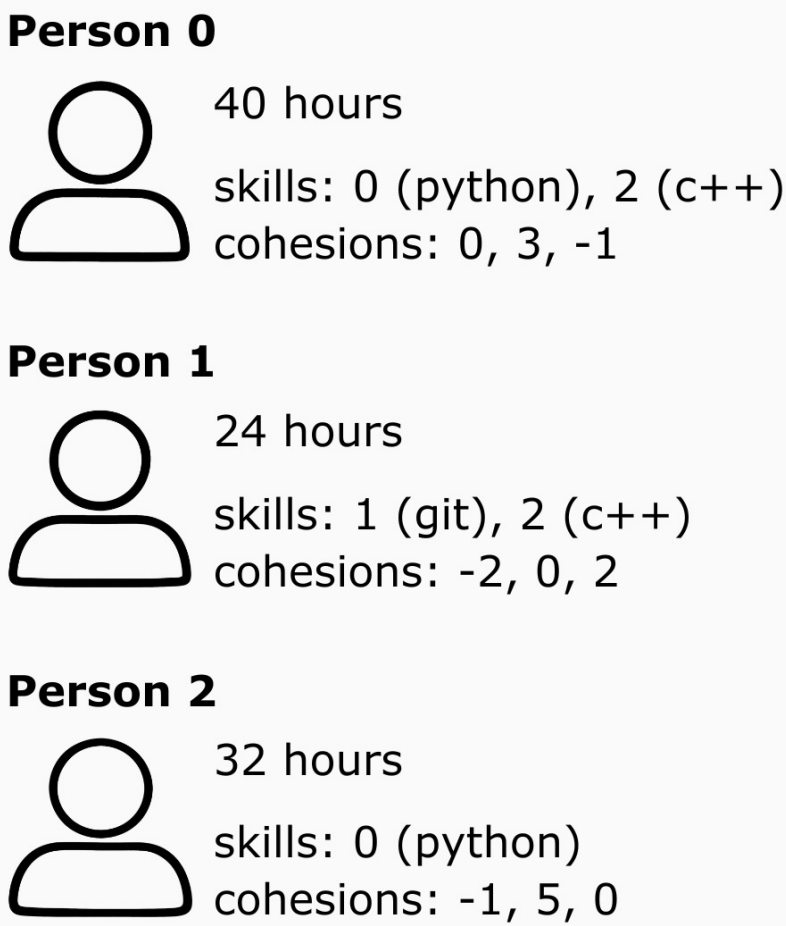


Figura: Availability and skills (identified by numerical IDs) of 3 people.

Cohesion among colleagues

At ***Dunder Malloc***, it's well known that not everyone gets along.

For each pair of people p and q, Director Scott identifies 2 values w_{pq} and w_{qp}: the first represents how well p feels working with q (*p's cohesion with q*), the second represents the vice versa.

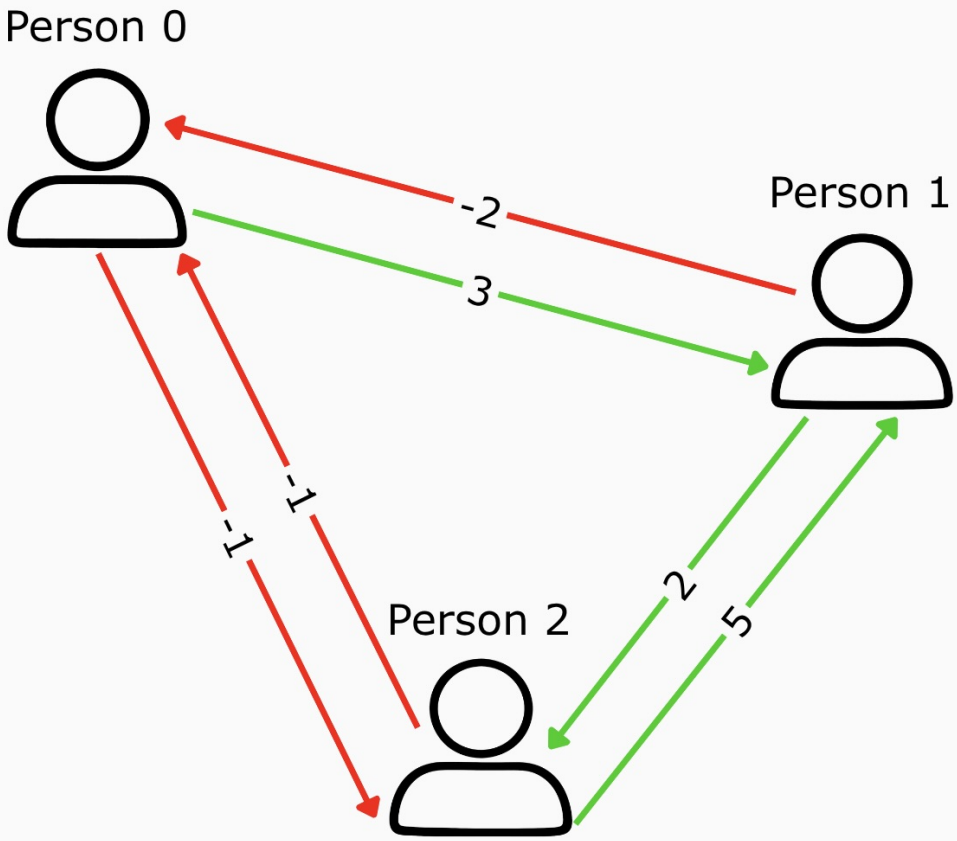


Figura: The mutual cohesion of 3 people.

The Projects

Albert Scott's difficult role is to assign the right people to the **projects** that the company wants to carry out.

Each project requires a certain number of hours to be completed and a set of skills.

Project 0

42 hours

Required skills:
0, 1, 2

Project 1

20 hours

Required skills:
1, 2

Project 2

30 hours

Required skills:
0, 2

Figura: Projects and their respective time and skill requirements.

Company Policy

Albert Scott has big plans, but the collaborators are hostile. Fearing they will be scattered across too many projects without being able to focus on any, they make several requests and win a small concession: it is agreed that **no one can work on more than** K **projects at the same time**.

And now? Your task (I)

Dr. Scott, suspecting a *Not-Particularly* easy resolution, admits he has no idea how to solve the problem of assigning people to projects: could you help him?

And now? Your task (II)

Your task is to help the director build a valid assignment of people to projects.
The objectives, in order of priority, are:

- **cover the skills required by the projects**: a project is considered satisfied only if, among the people assigned to it, there is at least one person for each required skill;
- **minimize uncovered time**: only for projects with covered skills, we measure how much of the hourly demand remains unassigned; if a project does not cover all required skills, its entire demand is considered uncovered;
- **maximize team cohesion**: among solutions with similar coverage, teams with high overall cohesion are preferable.

Input (I)

A file with $1 + 3 \cdot P + 2 \cdot J$ lines.

- The first line contains 4 integers: P (`int`) the number of people, J (`int`) the number of projects, S (`int`) the number of existing skills, and K (`int`) the maximum number of projects a person can work on.
- The following $3 \cdot P$ lines (from 2 to $1 + 3 \cdot P$) describe the people. For $p = 0, \ldots P - 1$:
  - each line $2 + 3 \cdot p$ contains 2 integers, cap_p ($\texttt{int} \in [1, 40]$), the number of hours per week that person p can work, and σ_p (`int`) their number of skills;
  - each line $2 + 3 \cdot p + 1$ contains σ_p integers (`int`), $\{s_i\}_{i=0}^{\sigma_p - 1}$, the identifiers of the skills possessed by person p;
  - each line $2 + 3 \cdot p + 2$ contains P integer values (`int`), $\{w_{pq}\}_{q=0}^{P-1}$, where the q-th value w_{pq} represents person p's work cohesion with person q.

Input (II)

- The following $2 \cdot J$ lines (from $2 + 3 \cdot P$ to $1 + 3 \cdot P + 2 \cdot J$) describe the projects. For $j = 0, ...J - 1$:
  - each line $2 + 3 \cdot P + 2 \cdot j$ contains an integer dem_j (`int`), the number of hours needed to complete project j and $\hat{\sigma}_j$ (`int`) the number of skills required for project j;
  - each line $2 + 3 \cdot P + 2 \cdot j + 1$ contains $\hat{\sigma}_j$ integer values (`int`), the identifiers of the skills required by project j.

Output

A file containing your proposed solutions, where each proposal consists of $(2 + 3 \cdot J)$ lines. **Each proposal:**

- the first line contains the `score` value (`int`) calculated as described in the next slide;
- following this, for each project:
  - the first line contains the project index j, followed by the number of people num_j assigned to the project;
  - the next line contains the list of people assigned to the project;
  - the next line contains the respective working time (in weekly hours) of each person on the project, **which must always be greater than 0**;
- the proposal must end with a final line: `***`.

The score

The value you have to compute and minimize is:

$$\texttt{score} = \alpha \cdot \texttt{total\_unmet} - \beta \cdot \texttt{total\_cohes}.$$

Weights: $\alpha = 7$, $\beta = 3$.

Uncovered time:

$$\texttt{total\_unmet} = \sum_{j=0}^{J-1} u_j, \qquad u_j = \begin{cases} dem_j & \text{if the skills of project } j \\ \text{uncovered time} & \text{are not all covered} \\ \text{in project } j & \text{otherwise} \end{cases}$$

Total cohesion:

$$\texttt{total\_cohes} = \left\lfloor \frac{1}{2P} \sum_{j=0}^{J-1} \sum_{p,q=0}^{P-1} y_{j,p} y_{j,q} w^*_{j,pq} \right\rfloor, \quad w^*_{j,pq} = \begin{cases} -|w_{pq}| & \text{if the skills of project } j \\ & \text{are not all covered} \\ w_{pq} & \text{otherwise} \end{cases}$$

$y_{j,p}$ is 1 if p is assigned to project j, 0 otherwise. It is always true that $w_{pp} = 0$.

Example I - input

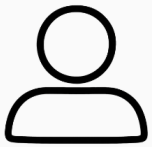
```
3 3 3 2
40 2
0 2
0 3 -1
24 2
1 2
-2 0 2
32 1
0
-1 5 0
42 3
0 1 2
20 2
1 2
30 2
0 2
```

Person 0

40 hours

skills: 0 (python), 2 (c++)
cohesions: 0, 3, -1

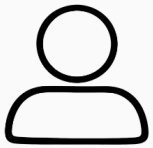
Person 1



24 hours

skills: 1 (git), 2 (c++)
cohesions: -2, 0, 2

Person 2



32 hours

skills: 0 (python)
cohesions: -1, 5, 0

Project 0

42 hours

Required skills:
0, 1, 2

Project 1

20 hours

Required skills:
1, 2

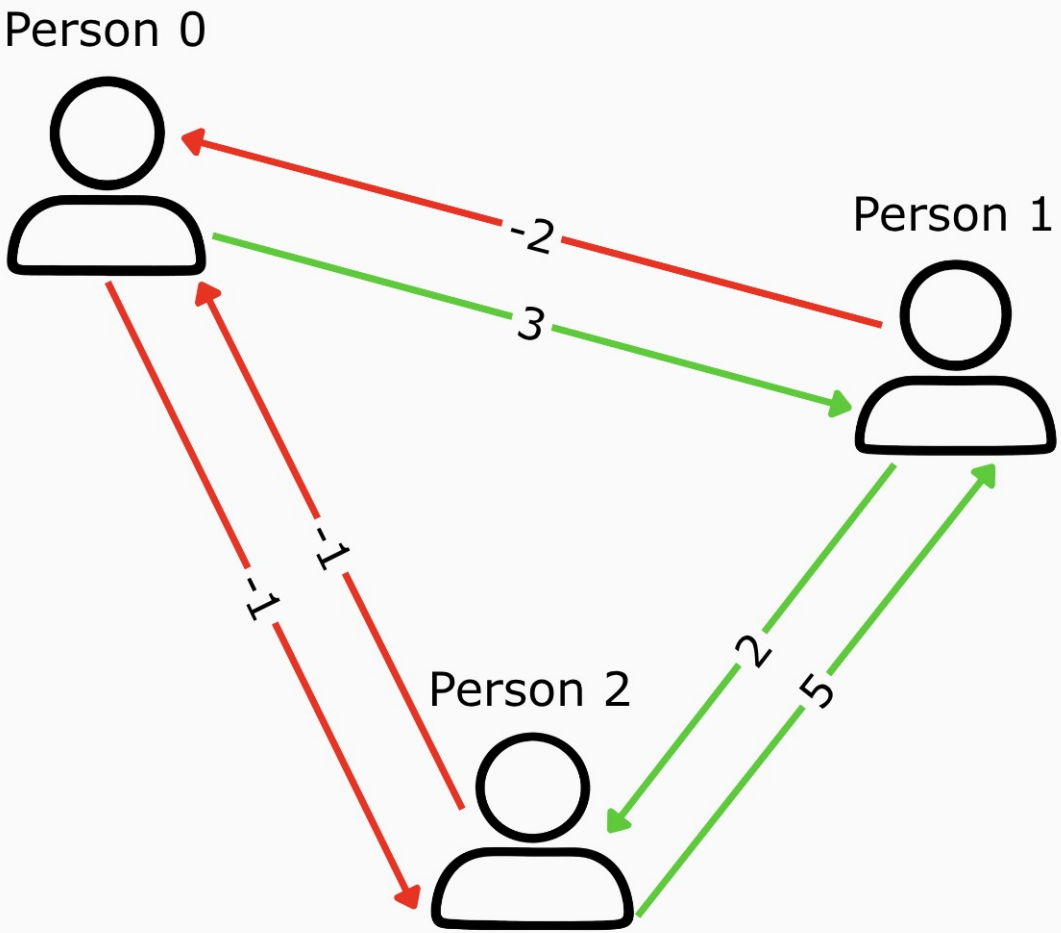
Project 2

30 hours

Required skills:
0, 2

Example I - cohesions

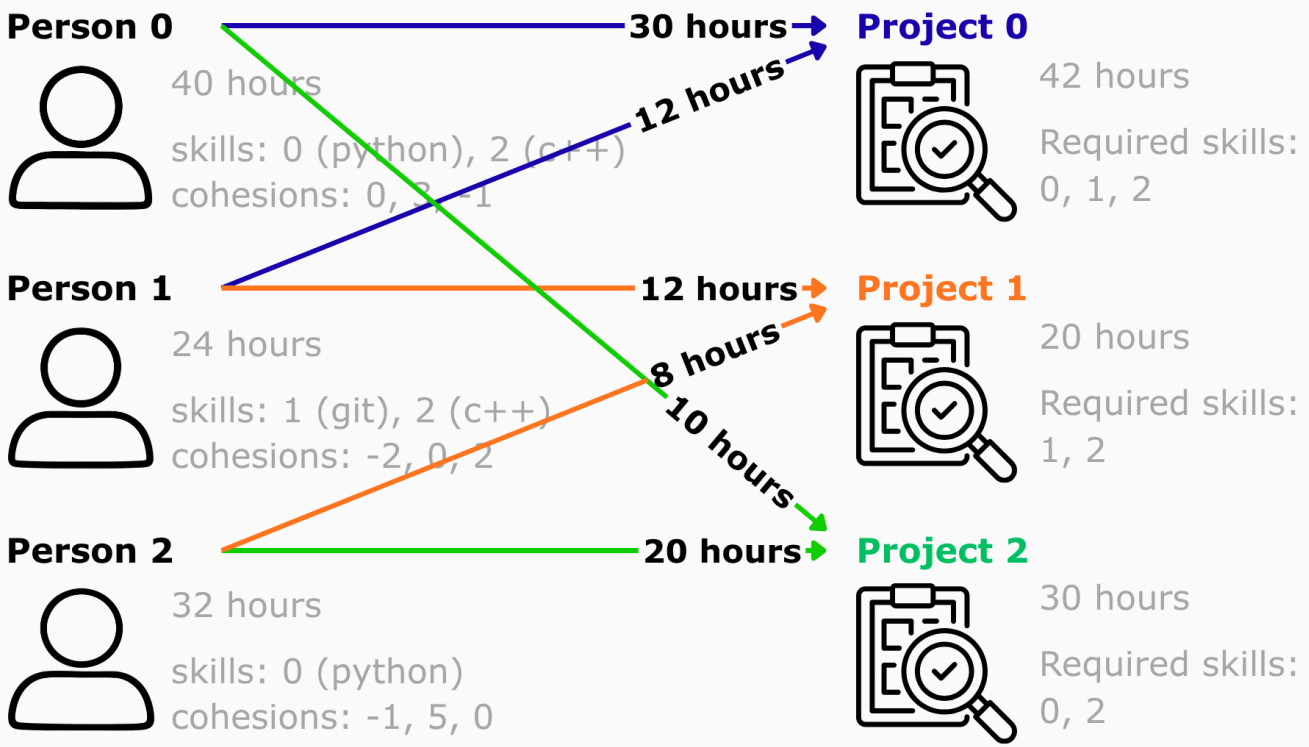
The cohesions between the people in the previous example can be represented with a directed graph:



Example I - valid output

In the previous example, a valid output is the following:

```
-3
0 2
0 1
30 12
1 2
1 2
12 8
2 2
0 2
10 20
***
```

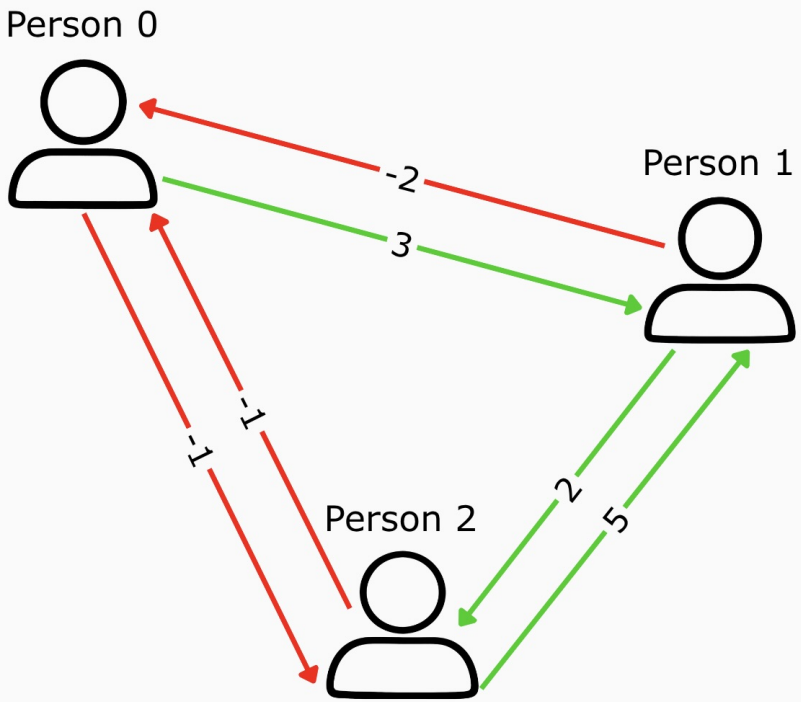


In all projects, the skills are covered and the hourly demand is satisfied.

Example I - score calculation

In this case `total_unmet` $= 0$ and `total_cohes` $= 6$. Therefore `score` $= -18$.

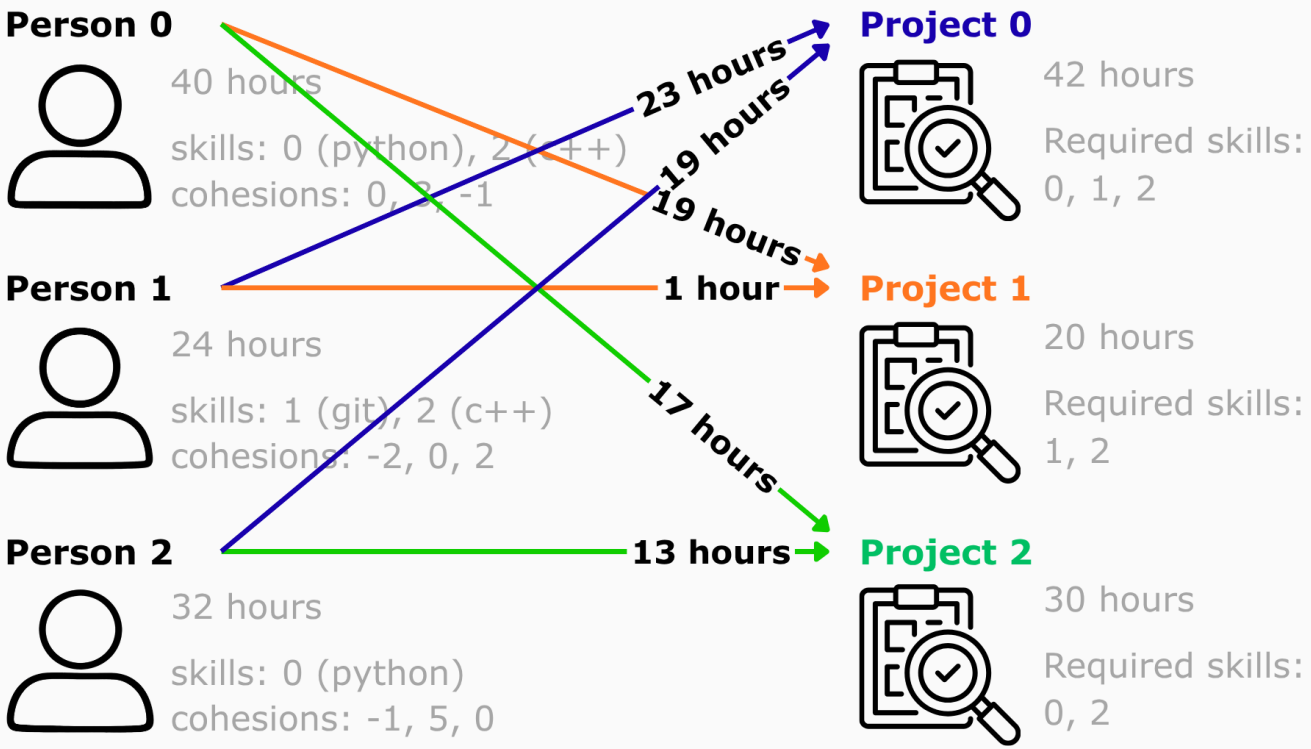
```
-3
0 2
0 1
30 12
1 2
1 2
12 8
2 2
0 2
10 20
***
```



Example I - another valid output

In the previous example, another valid output is the following:

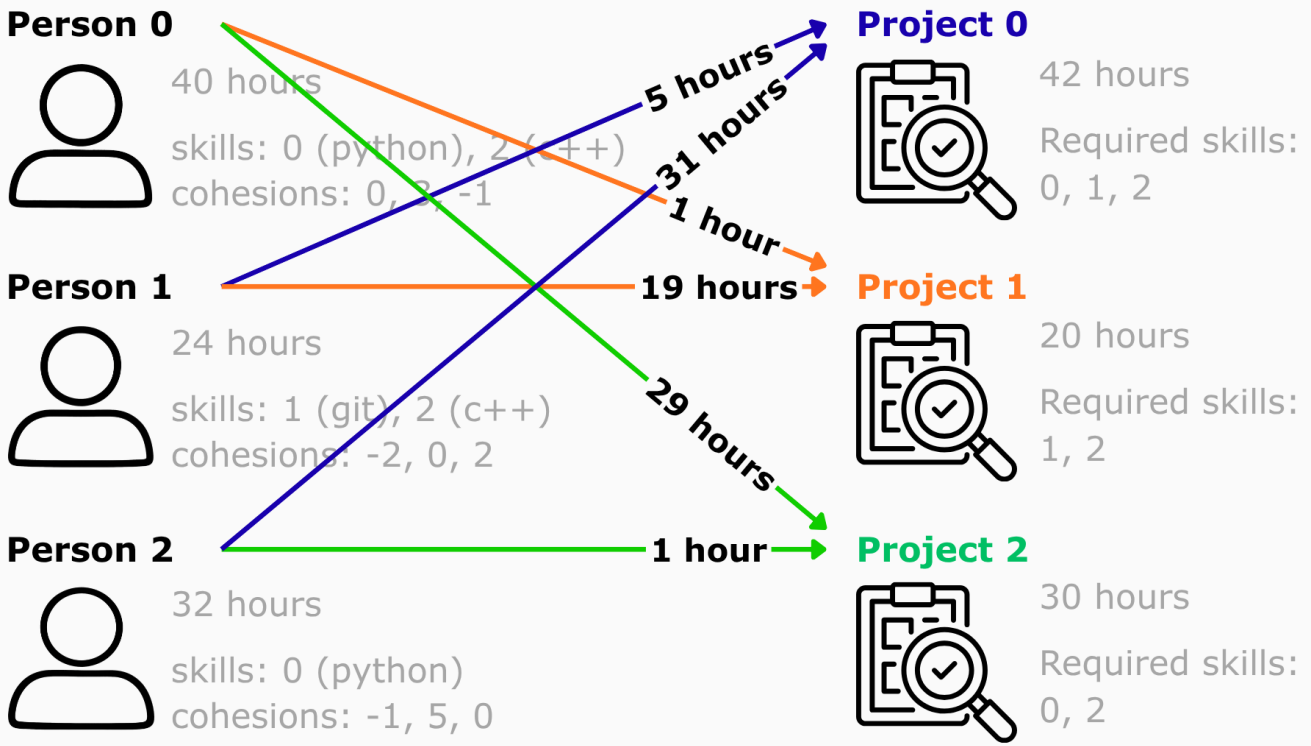
```
-3
0 2
1 2
23 19
1 2
0 1
19 1
2 2
0 2
17 13
***
```



Example I - valid but lower-scoring output (I)

In the previous example, another valid output but with a lower score is the following (not all requested hours are covered):

```
39
0 2
1 2
5 31
1 2
0 1
1 19
2 2
0 2
29 1
***
```

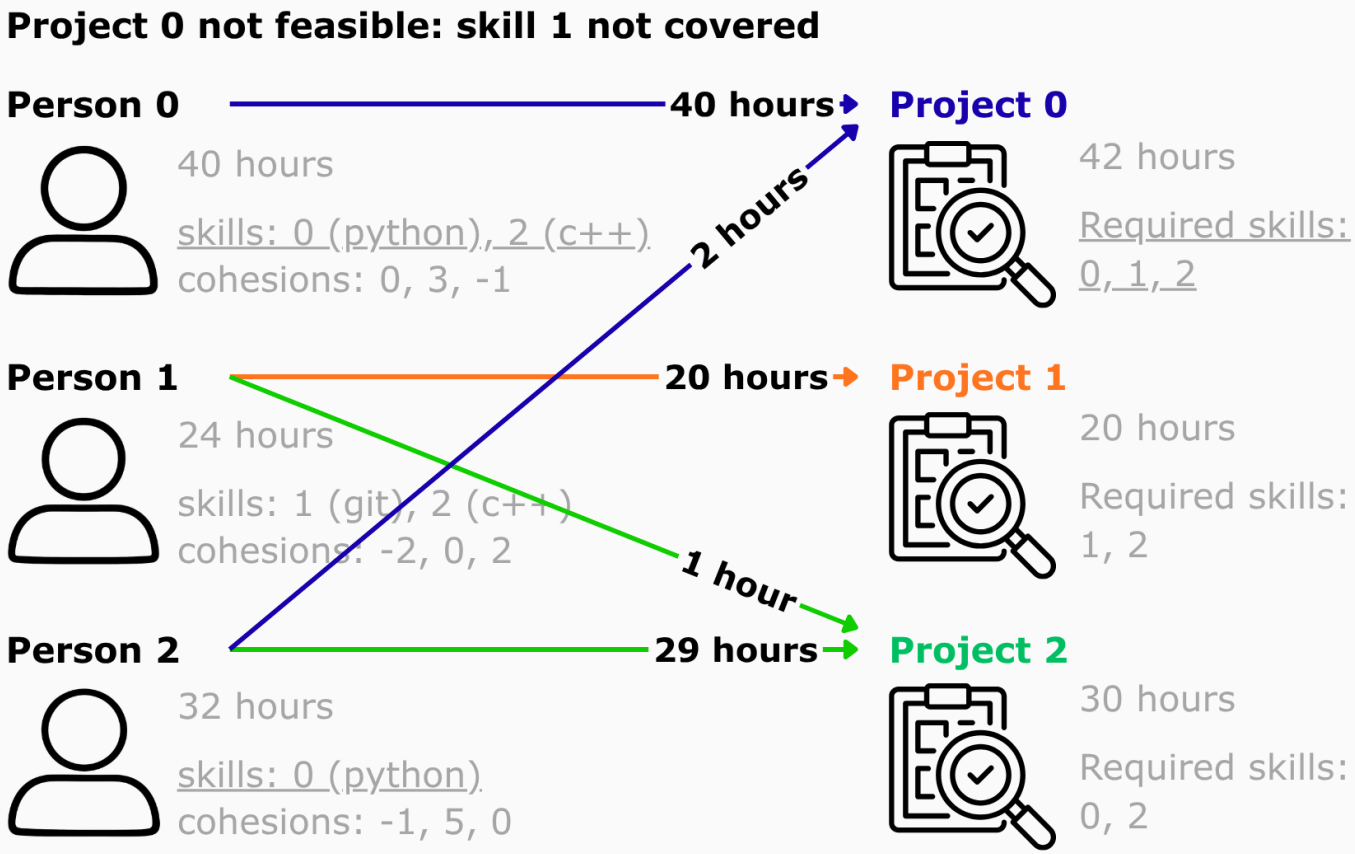


Here `total_unmet` $= 6$, `total_cohes` $= 1$ and `score` $= 42 - 3 = 24$.

Example I - valid but lower-scoring output (II)

In the previous example, another valid output but with a lower score is the following (not all skills of a project are covered):

```
294
0 2
0 2
40 2
1 1
1
20
2 2
1 2
1 29
***
```

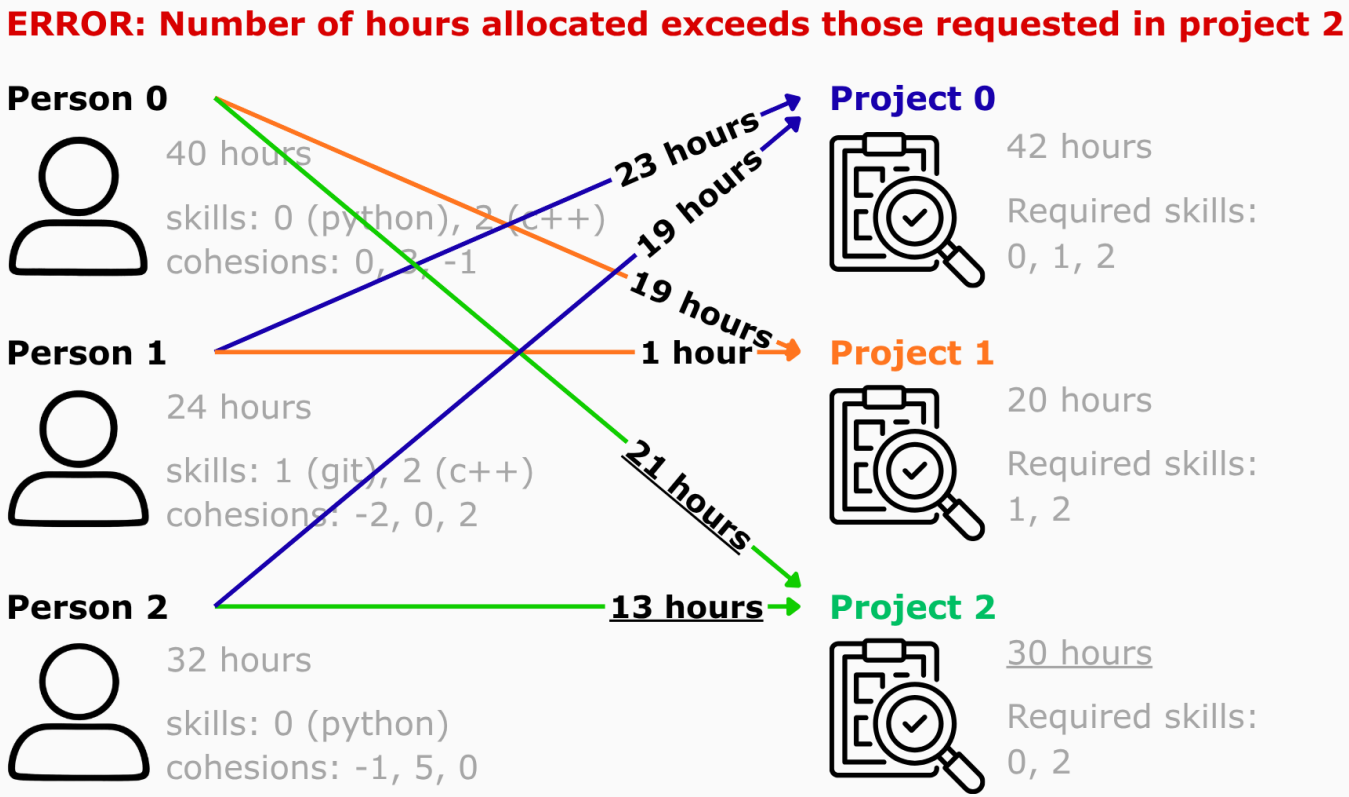


Here `total_unmet` = 42, `total_cohes` = 0, `score` = 294.

Example I - incorrect output (I)

In the previous example, an incorrect output could be:

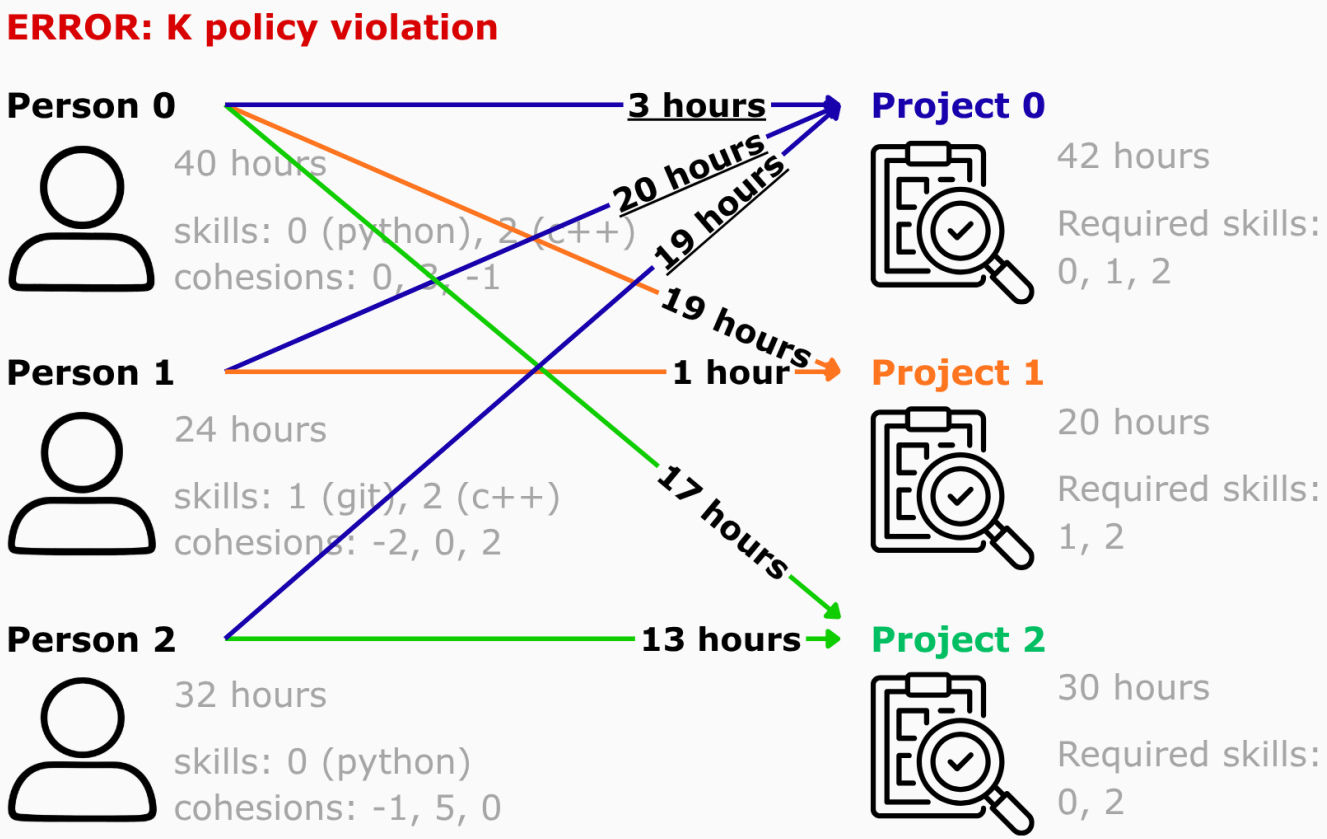
```
-25
0 2
1 2
23 19
1 2
0 1
19 1
2 2
0 2
21 13
***
```



Example I - another incorrect output (II)

In the previous example, another incorrect output could be:

```
-15
0 3
0 1 2
3 20 19
1 2
0 1
19 1
2 2
0 2
17 13
***
```



Notes on Input and Output

General input assumptions (I)

- $1 \leq P \leq 5\,000$
- $1 \leq J \leq 5\,000$
- $1 \leq S \leq 500$
- $1 \leq K \leq 40$
- $-10 \leq w_{pq} \leq 10$ for each pair of people p and q
- $w_{pp} = 0$ for each person p
- $1 \leq cap_p \leq 40$ for each person p
- $1 \leq dem_j \leq 5\,000$ for each project j

Notes on Input

General input assumptions (II)

- People are identified by values $0...P-1$, projects by values $0...J-1$, skills by values $0...S-1$.
- Each person has at least one skill.
- Each project requires at least one skill.

General output assumptions (I)

- If a project is assigned more time than it requires, it is considered an error.
- If a person is assigned with 0 hours to a project, it is considered an error.

Test cases

There are 20 test cases in total:

- ★☆☆ In 6 out of 20 cases, $K = J$, $w_{pq} = 0$ for every p and q, all people possess the *same* skill and all projects require *only* that skill;
- ★★☆ In 9 out of 20 cases, $w_{pq} = 0$ for every p and q;
- ★★★ In the remaining cases there are no particular limitations.

For a passing grade, for example, you can focus on the 6 cases with difficulty ★☆☆.

Time and memory limits

The time and memory limits are:

- Timeout: 2 seconds (*soft*); 2.4 seconds (*hard*).
- Maximum memory: 48 MB.

$\Rightarrow$ Limit of 50 submissions per group.

$\Rightarrow$ You can test with an equivalent sample dataset on your own machine.

$\Rightarrow$ **Note:** in the output files in the sample dataset you will find two numbers: `worst_score` (`long long int`) and `best_score` (`long long int`). These are the values needed to calculate the score of each solution starting from `score`.

Output

You can print solutions incrementally:

- Import `the-office.h` (downloadable from arena and the website).
- As you improve the solution, write it to the output, ending the line with `***`.
- The library will stop the program before the timeout.

```
... system library includes ...
#include "the-office.h"
int main() {
  ...
}
```

Notes:

- The `main` must always be declared as `int main()` or `int main(void)`.
- The judge will consider the **last solution** terminated by `***`; therefore, even if you do not print multiple solutions, terminate the output with `***`.

Compiling locally

To test your solutions locally (assuming your file is named `the-office.cpp`):

- Download `grader.cpp`
- The compilation command is as follows

```
/usr/bin/g++ -DEVAL -std=gnu++11 -O2 -pipe -static -s -
    o the-office grader.cpp the-office.cpp
```

The files `the-office.cpp`, `grader.cpp`, and `the-office.h` must be in the same folder.

For Mac OS X and Windows systems, see the note in the text.

Note

For this exercise, it is necessary to use `C++`; it is not possible to use `C`.

Scoring (I)

Each test case is worth 5 points. The maximum score is 100 points.

The parameters used in the evaluation are the following:

- `total_unmet`: the total number of uncovered hours in projects (calculated by summing dem_j for projects where not all skills are covered);
- `total_cohes`: the total cohesion (calculated by summing $-|w_{pq}|$ for projects where not all skills are covered);
- `score` $= \alpha \cdot$ `total_unmet` $- \beta \cdot$ `total_cohes`;
- `sum_dem`: the total number of hours required by the projects;
- `lower_bound_cohes`: an estimated lower bound for the total cohesion value;
- `upper_bound_cohes`: an estimated upper bound for the total cohesion value.

Scoring (II)

Final score

For each test case for which your solution provides a valid output within the time and memory limits, you will obtain the following score:

$$\mathcal{P} = 5.0 \times \left(1 - \frac{\texttt{score} - \texttt{best\_score}}{\texttt{worst\_score} - \texttt{best\_score}}\right)$$

$\texttt{worst\_score} = \alpha * \texttt{sum\_dem} - \beta * \texttt{lower\_bound\_cohes}$ is an upper bound for the worst score that can be obtained.

$\texttt{best\_score} = -\beta * \texttt{upper\_bound\_cohes}$ is a lower bound for the best score that can be obtained (remember that the goal is to minimize the $\texttt{score}$!).

Bounds

Your goal

It is not necessary for you to calculate the final score of your solution – you would need the cohesion bounds which you do not have available. But it's not a problem: those are used by us to calculate the final score.

Just remember that your goal is to **minimize the `score` value**.

Example I - score calculation

In the first example, an output had $\texttt{score} = -18$.

In that case $\texttt{sum\_dem} = 92$, $\texttt{lower\_bound\_cohes} = -8$ and $\texttt{upper\_bound\_cohes} = 20$. Calculating the score estimates, we get $\texttt{worst\_score} = 668$, $\texttt{best\_score} = -60$.

Therefore

$$\mathcal{P} = 5.0 \times \left(1 - \frac{-18 - (-60)}{668 - (-60)}\right) = 4.71$$

NB: As you can see, the score is very good. Also note that, as estimates for the best and worst scores must be used, it is possible that some cases do not reach the maximum score even when finding the optimal solution.

Scoring

⇒ The **passing threshold is set at 30 points**.

✗ if any information provided in the output is incorrect (for example, policy K is violated), you obtain 0 points.

Bonus points for the exam

Points are assigned competitively:

- **3 points** to groups in the first tertile of the ranking (top third of the ranking);
- **2 points** to groups in the second tertile of the ranking (middle third of the ranking);
- **1 point** to groups in the third tertile of the ranking (bottom third of the ranking).

Only the **groups that reach the passing threshold** (score greater than or equal to 30) are considered in the ranking for point assignment.

$\Rightarrow$ Ranking: https://asdlab.disi.unitn.it/arena/ranking/

Submission

Deadline: Friday, May 22, 2026, 6:00 PM

To upload your code, go to
`https://asdlab.disi.unitn.it/arena/`

We remind you that the **last** solution submitted will be considered for the score calculation.

Suggestions

Start working on the project immediately so you can come to the next office hours (Tuesday, May 19) with all the questions you want to ask. In any case, know that:

- you can come to office hours
- we answer on Telegram
- we will respond to your emails

Do's

Permitted:

- Discussing within the group
- Asking for clarifications on the text
- Asking for opinions on solutions
- Using code provided in the labs
- Using pseudocode from books or Wikipedia
- Requesting help (even significant) for the "minimum" solution
- Coming to office hours
- Sending (algorithmic) memes on our Telegram channel: it's your time to shine!

Don'ts

Forbidden:

- Discussing with other groups
- Putting your code on public repositories
- Using code written by others
- Sharing code (we have powerful means!)
- Asking for suggestions online (e.g., StackOverflow)
- Making references to project solutions in memes

Office Hours

Dates and times

- Tuesday, May 19, 2026, from 1:30 PM to 3:30 PM (B109);
- Thursday, May 21, 2026, from 11:30 AM to 1:30 PM (B109);
- there will be additional online office hours on days when there is no in-person lab; we will let you know.

$\Rightarrow$ During office hours, we will always be available online; when you need help, write it on the Telegram group and we will respond in private chat or via a Zoom call.

$\Rightarrow$ For any questions, send us an email at: `asd.disi@unitn.it` or contact us on Telegram.