Secondo Progetto ASD 2025/2026

Secondo Progetto ASD 2025/2026

La Dunder Malloc (I)

In una delle filiali della grande ***Dunder Malloc***, nella profonda Povsylvania trentina, c'è aria di importanti cambiamenti.

Tra colleghi e colleghe, che non mancano di passione per i pettegolezzi, ha iniziato a diffondersi ormai da settimane la notizia di un **nuovo direttore**.

La Dunder Malloc (II)

In un mondo dominato dai migliori allocatori di memorie, l'azienda è da sempre sull'orlo del fallimento.

I dipendenti della ***Dunder Malloc***, presi dallo sconforto, lavorano in modo sempre più svogliato e inefficiente, sprecando tempo e capacità preziose.

Il nuovo direttore

Ma finalmente oggi è il grande giorno: il nuovo direttore **Albert Scott** fa il suo ingresso in azienda, pronto e convinto di poter cambiare le cose.

Per il dottor A. Scott la parola d'ordine è **ottimizzazione**.

"Si può fare meglio di così?" il suo famosissimo motto.

Chi lavora alla Dunder Malloc?

Il nuovo direttore, amante di qualsiasi chiacchiera possa tenere in pausa il suo "duro lavoro", chiama, uno alla volta, collaboratori e collaboratrici nel suo ufficio per un colloquio.

Si informa sulle loro **skills**, sulla loro **disponibilità oraria** e sulla **coesione** con ciascuno dei colleghi.

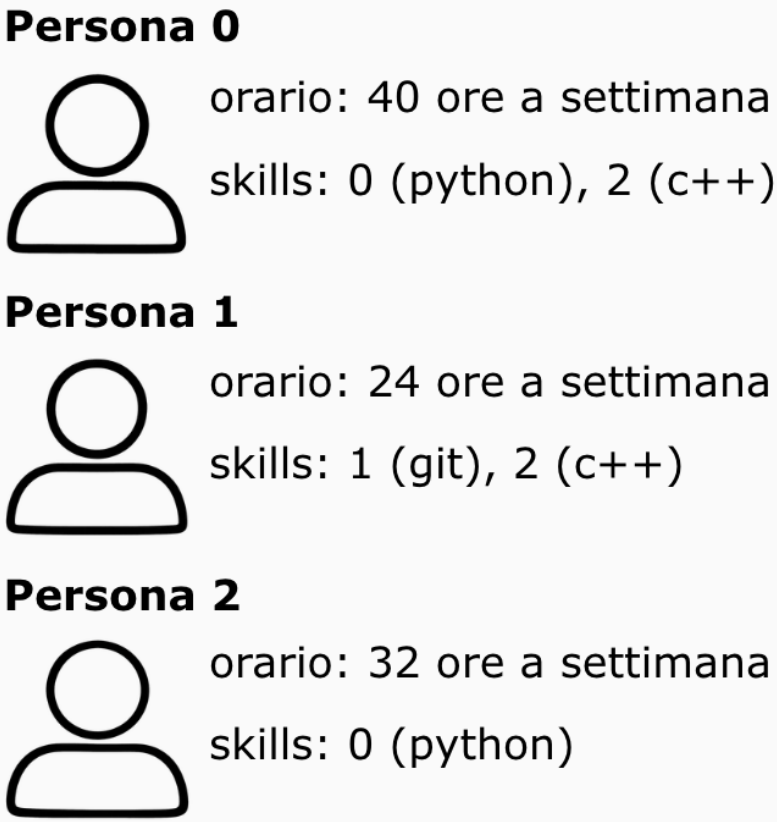


Figura: Disponibilità e skills (identificate da id numerici) di 3 persone.

La coesione tra colleghi e colleghe

Alla ***Dunder Malloc***, si sa, non tutti vanno d'accordo.

Per ogni coppia di persone p e q, il direttore Scott identifica 2 valori w_{pq} e w_{qp}: il primo rappresenta quanto p si trova bene a lavorare con q (*coesione di p con q*), il secondo il viceversa.

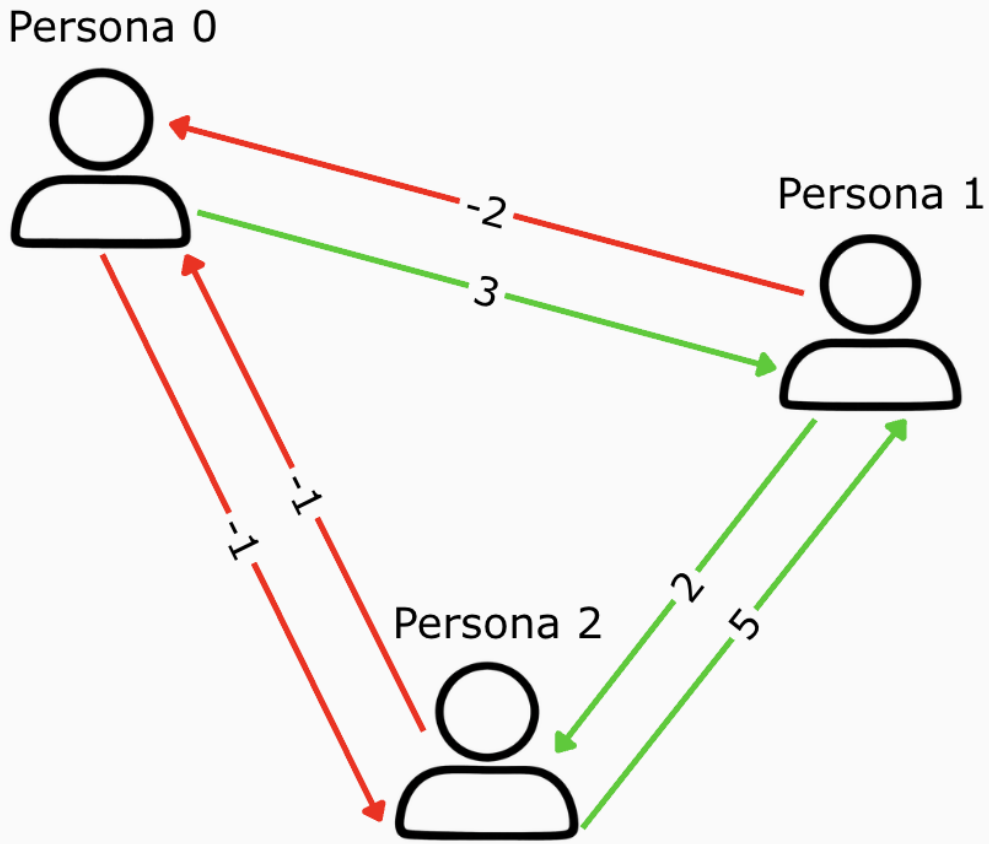


Figura: Le coesioni reciproche di 3 persone.

I PROGETTI

Il difficile ruolo di Albert Scott è quello di assegnare le persone giuste ai **progetti** che l'azienda vuole portare avanti.

Ogni progetto richiede un certo numero di ore per essere svolto e un insieme di skills.

Progetto 0

ore richieste: 42

skills richieste: 0, 1, 2

Progetto 1

ore richieste: 20

skills richieste: 1, 2

Progetto 2

ore richieste: 30

skills richieste: 0, 2

FIGURA: Progetti e rispettive richieste di tempo e skills.

La policy aziendale

Albert Scott ha grandi piani, ma i collaboratori sono ostili. Temendo di essere dispersi su troppi progetti senza potersi concentrare su nessuno, avanzano varie richieste e strappano una piccola concessione: si concorda che **nessuno potrà lavorare a più di K progetti in contemporanea**.

E ora? Il vostro compito (I)

Il dott. Scott, sospettando una *Non-Proprio* facile risoluzione, ammette di non avere idea di come risolvere il problema di assegnazione delle persone ai progetti: riuscireste ad aiutarlo?

E ora? Il vostro compito (II)

Il vostro compito è aiutare il direttore a costruire un'assegnazione valida delle persone ai progetti.
Gli obiettivi, in ordine di priorità, sono:

- **coprire le skill richieste dai progetti**: un progetto è considerato soddisfatto solo se, tra le persone assegnate ad esso, è presente almeno una persona per ciascuna skill richiesta;
- **minimizzare il tempo non coperto**: per i soli progetti con skill coperte, si misura quanta parte della domanda oraria resta non assegnata; se un progetto non copre tutte le skill richieste, tutta la sua domanda viene considerata non coperta;
- **massimizzare la coesione dei team**: tra soluzioni con simile copertura, sono preferibili team con alta coesione complessiva.

Input (I)

Un file con $1 + 3 \cdot P + 2 \cdot J$ righe.

- La prima riga riporta 4 numeri interi: P (`int`) il numero di persone, J (`int`) il numero di progetti, S (`int`) il numero di skills esistenti, K (`int`) il numero massimo di progetti a cui una persona può lavorare.
- Le successive $3 \cdot P$ righe (da 2 a $1 + 3 \cdot P$) descrivono le persone. Per ciascuna persona $p = 0, ...P - 1$:
  - la riga $(2 + 3 \cdot p)$-esima riporta 2 interi, cap_p ($\texttt{int} \in [1, 40]$), il numero di ore a settimana che può lavorare p, e σ_p (`int`) il suo numero di skills;
  - la riga $(2 + 3 \cdot p + 1)$-esima riporta σ_p interi, $\{s_i\}_{i=0}^{\sigma_p - 1}$ (`int`), gli identificativi delle skills che possiede p;
  - la riga $(2 + 3 \cdot p + 2)$-esima riporta P valori interi, $\{w_{pq}\}_{q=0}^{P-1}$ (`int`), dove il q-esimo valore w_{pq} è la coesione nel lavoro di p con q.

Input (II)

- Le successive $2 \cdot J$ righe (da $2 + 3 \cdot P$ a $1 + 3 \cdot P + 2 \cdot J$) descrivono i progetti. Per ogni progetto $j = 0, ...J - 1$:
  - la riga $(2 + 3 \cdot P + 2 \cdot j)$-esima riporta due numeri interi dem_j (`int`) e $\hat{\sigma}_j$ (`int`), rispettivamente il numero di ore e il numero di skills richieste per il progetto j;
  - la riga $(2 + 3 \cdot P + 2 \cdot j + 1)$-esima riporta $\hat{\sigma}_j$ valori interi, $\{\hat{s}_i\}_{i=0}^{\hat{\sigma}-1}$ (`int`), gli identificativi delle skills richieste dal progetto j.

Output

Un file con le vostre proposte di soluzione, ogni proposta è composta da $(2 + 3 \cdot J)$ righe. **Ogni proposta:**

- nella prima riga riporta il valore `score` (`int`) calcolato come descritto nella slide successiva;
- a seguire, per ogni progetto:
  - la prima riga riporta l'indice j del progetto, seguito dal numero di persone num_j assegnate al progetto;
  - la riga successiva riporta l'elenco delle persone assegnate al progetto;
  - la riga successiva riporta il rispettivo tempo di lavoro (in ore settimanali) di ciascuna persona sul progetto, **che deve essere sempre maggiore di 0**;
- la proposta deve terminare con un'ultima riga: `***`.

Lo score

Il valore che dovete calcolare e minimizzare è:

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

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

Tempo non coperto:

$$\texttt{total\_unmet} = \sum_{j=0}^{J-1} u_j \qquad u_j = \begin{cases} dem_j & \text{se le skill del progetto } j \text{ non sono tutte coperte} \\ \text{tempo non coperto nel progetto } j & \text{altrimenti} \end{cases}$$

Coesione totale:

$$\texttt{total\_cohes} = \left\lfloor \frac{1}{2P} \sum_{j=0}^{J-1} \sum_{p,q=0}^{P-1} y_{j,p} y_{j,q} w^*_{j,pq} \right\rfloor \qquad w^*_{j,pq} = \begin{cases} -|w_{pq}| & \text{se le skill di } j \text{ non sono tutte coperte} \\ w_{pq} & \text{altrimenti} \end{cases}$$

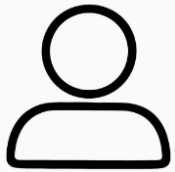
$y_{j,p}$ è 1 se p è assegnato al progetto j, 0 altrimenti. Vale sempre che $w_{pp} = 0$.

Esempio 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
```

P = 3, J = 3, S = 3, K = 2

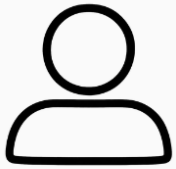
Persona 0



40 ore

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

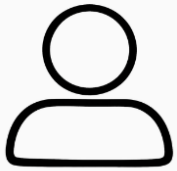
Persona 1



24 ore

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

Persona 2



32 ore

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

Progetto 0

42 ore

skills richieste:
0, 1, 2

Progetto 1

20 ore

skills richieste:
1, 2

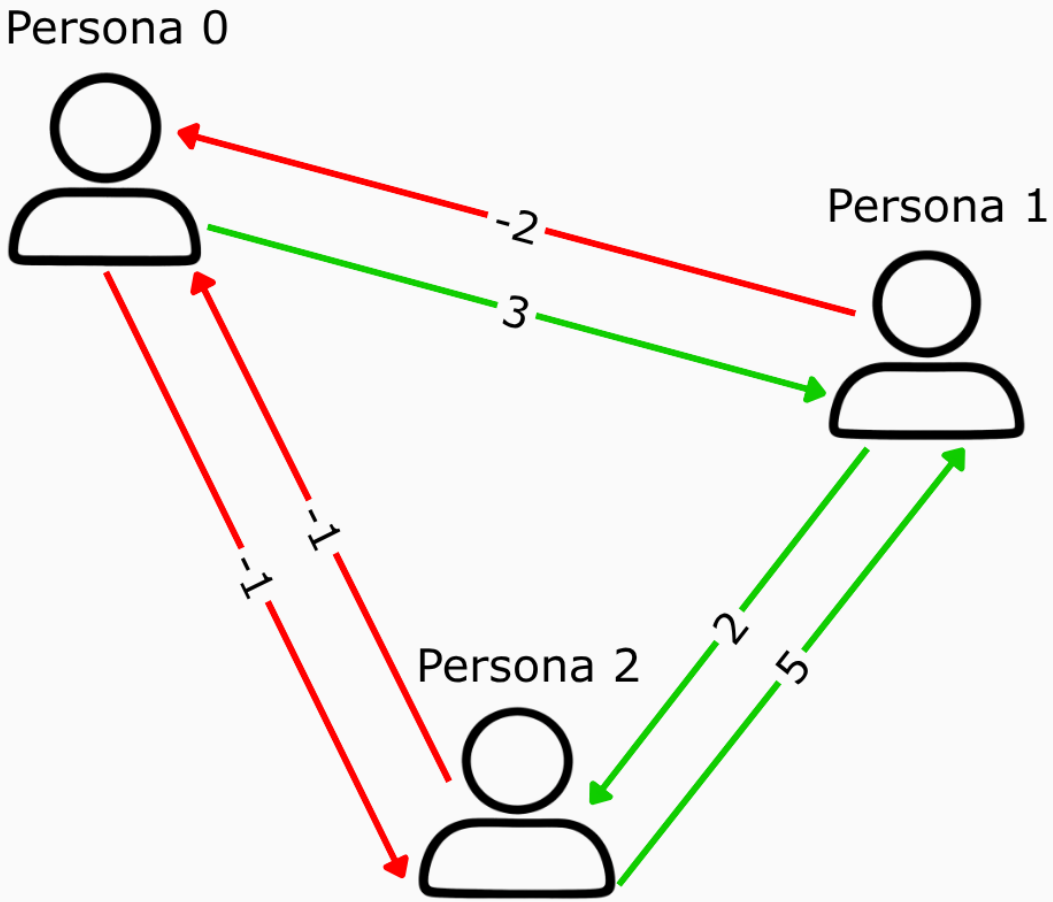
Progetto 2

30 ore

skills richieste:
0, 2

Esempio I - coesioni

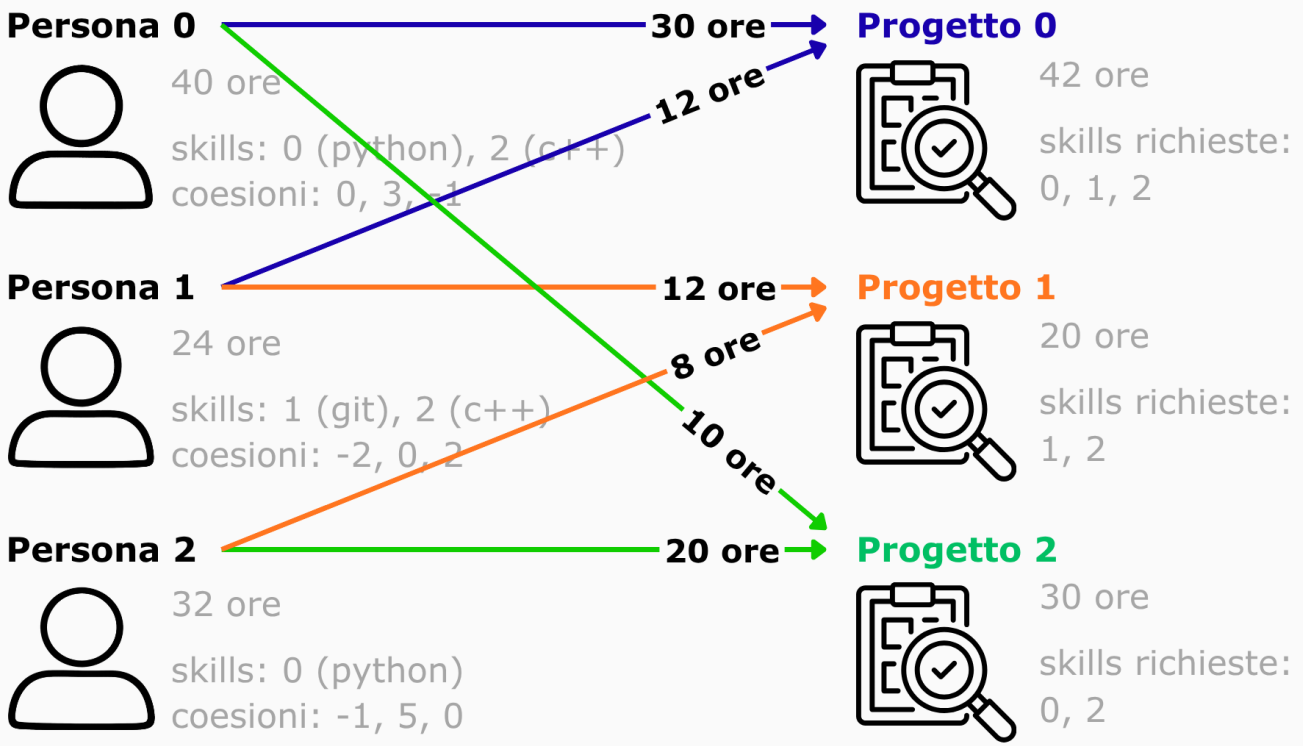
Le coesioni tra le persone dell'esempio precedente possono essere rappresentate con un grafo orientato:



Esempio I - output valido

Nell'esempio precedente, un output valido è il seguente:

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

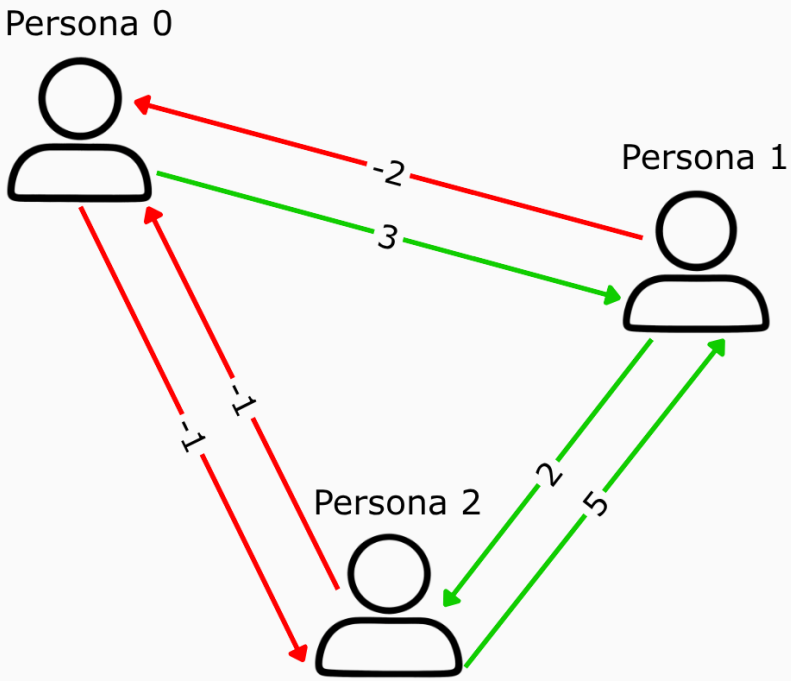


In tutti i progetti le skills sono coperte e la domanda oraria è soddisfatta.

Esempio I - calcolo score

In questo caso `total_unmet` $= 0$ e `total_cohes` $= 1$. Quindi `score` $= -3$.

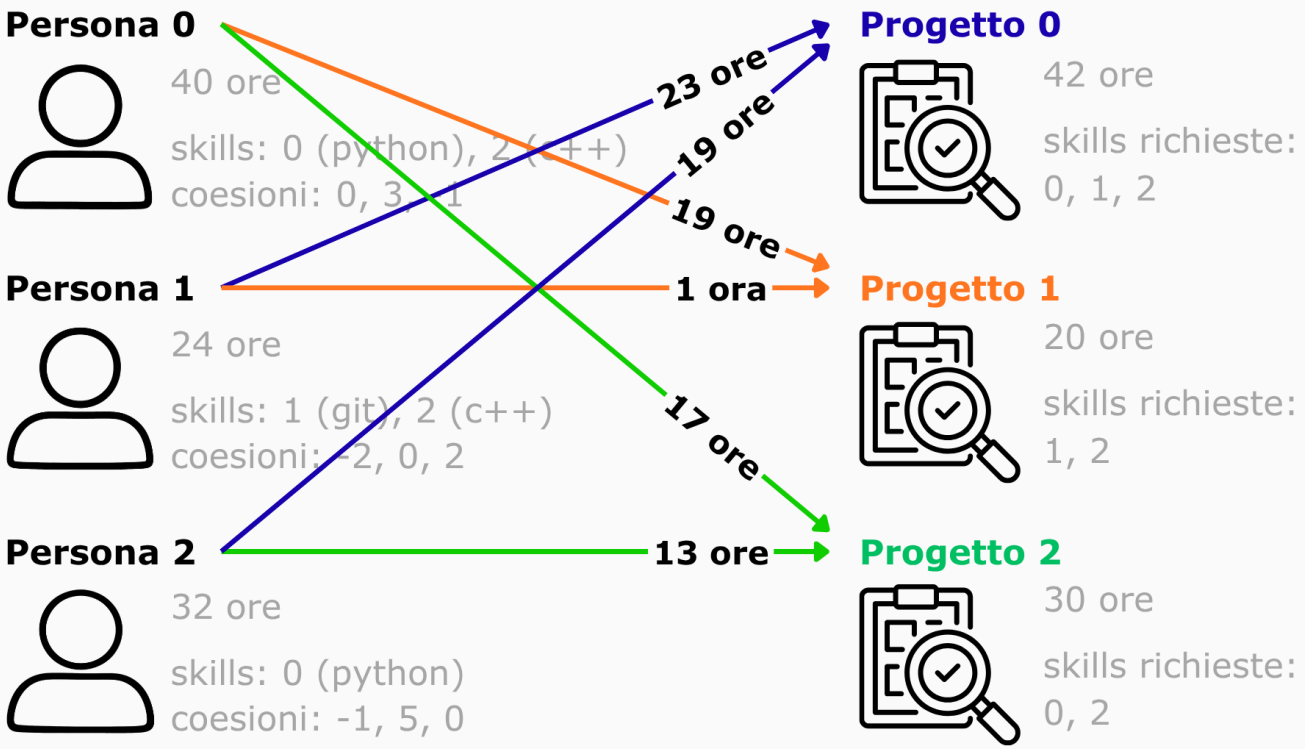
```
-3
0 2
0 1
30 12
1 2
1 2
12 8
2 2
0 2
10 20
***
```



Esempio I - altro output valido

Nell'esempio precedente, un altro output valido è il seguente:

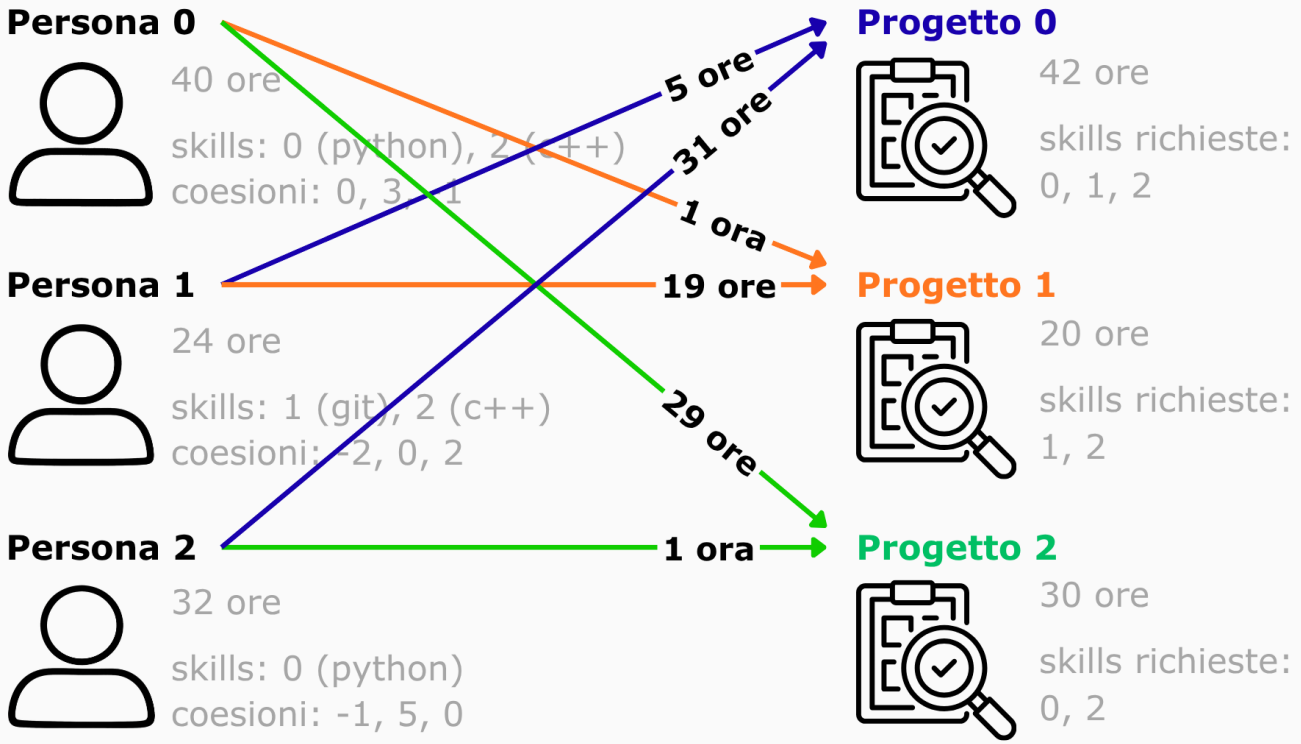
```
-3
0 2
1 2
23 19
1 2
0 1
19 1
2 2
0 2
17 13
***
```



Esempio I - output valido ma meno buono (I)

Nell'esempio precedente, un altro output valido ma con punteggio inferiore è il seguente (non tutte le ore richieste vengono coperte):

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

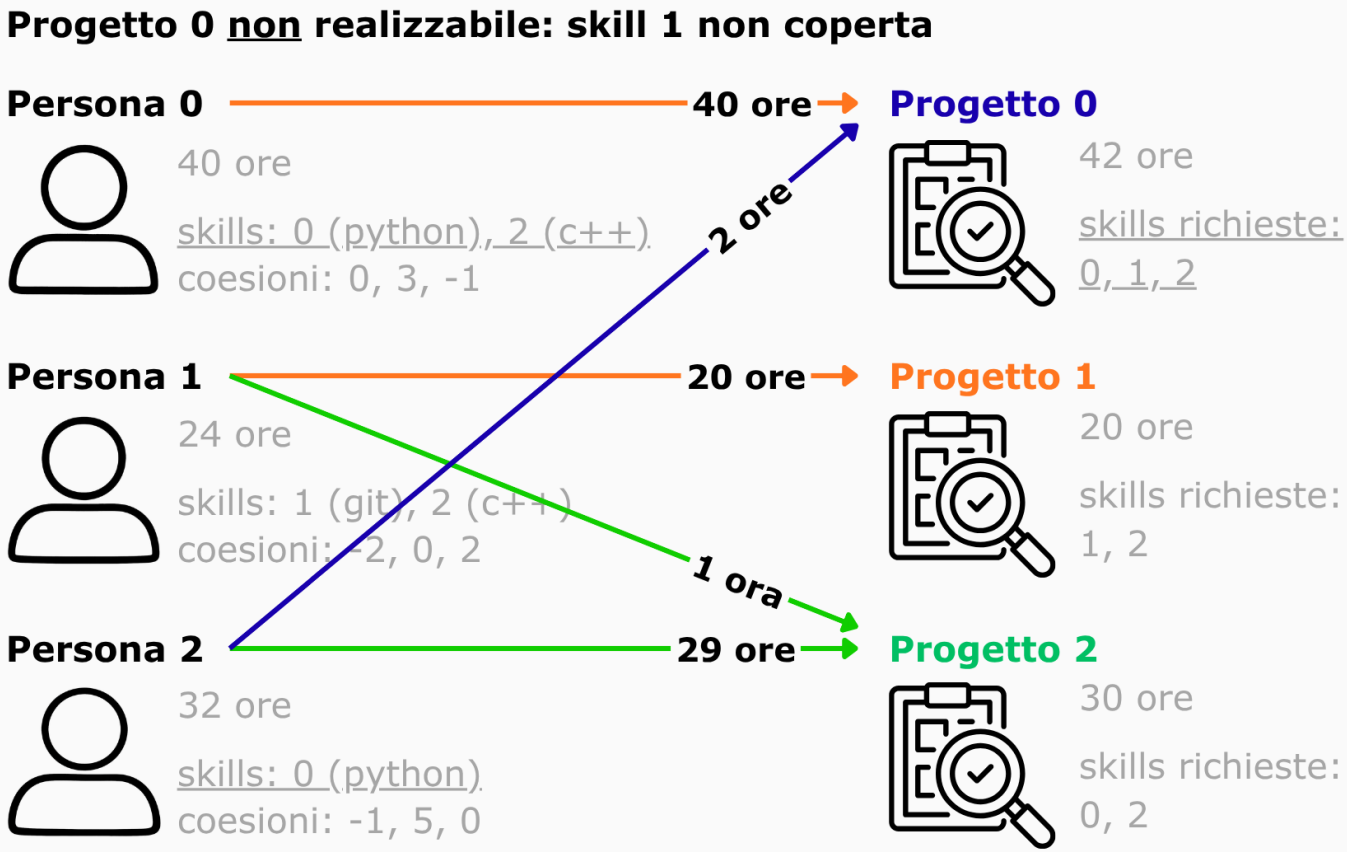


Qui `total_unmet` $= 6$, `total_cohes` $= 1$ e `score` $= 42 - 3 = 39$.

Esempio I - output valido ma meno buono (II)

Nell'esempio precedente, un altro output valido ma con punteggio inferiore è il seguente (non tutte le skills di un progetto vengono coperte):

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

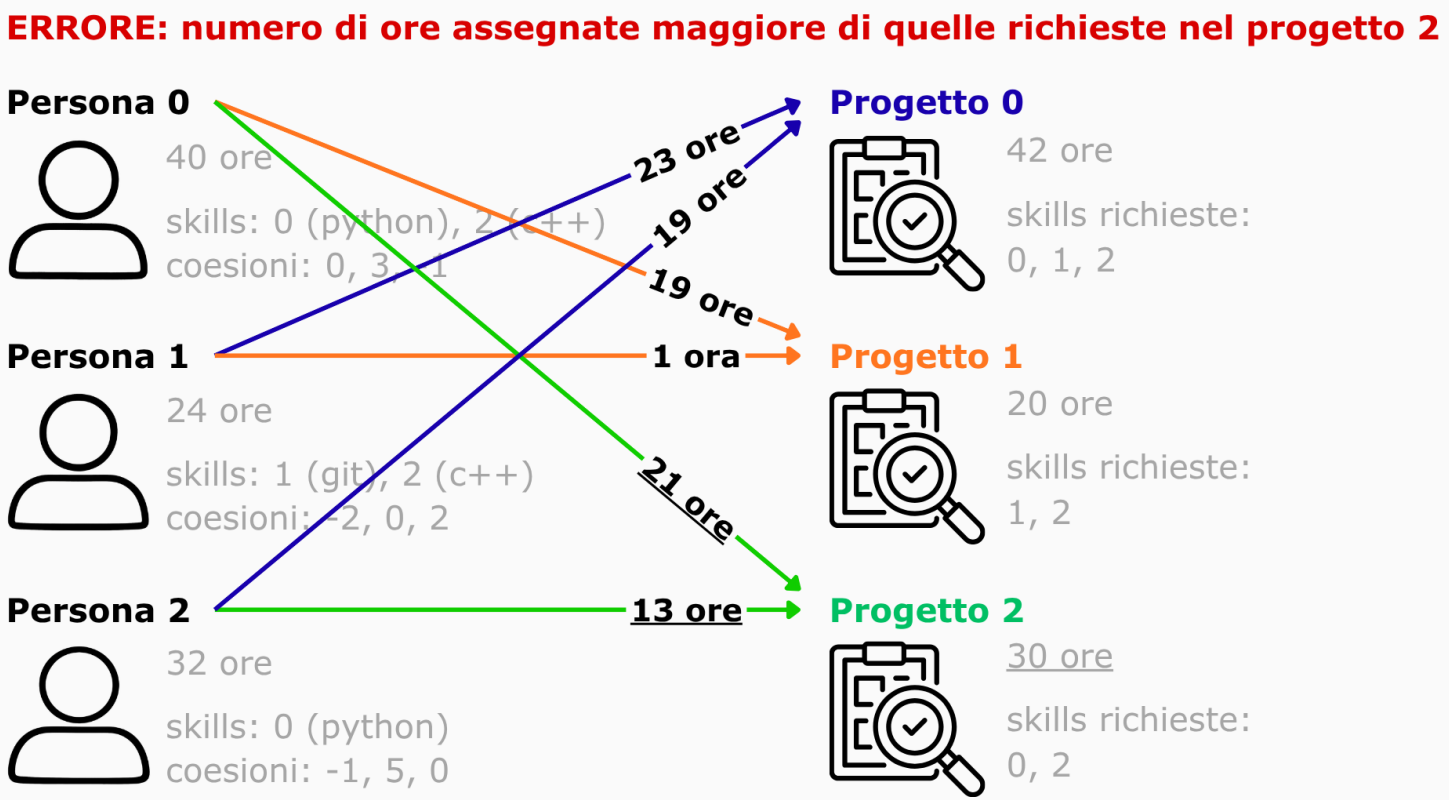


Qui `total_unmet` = 42, `total_cohes` = 0, `score` = 294.

Esempio I - output errato (I)

Nell'esempio precedente, un output errato può essere:

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

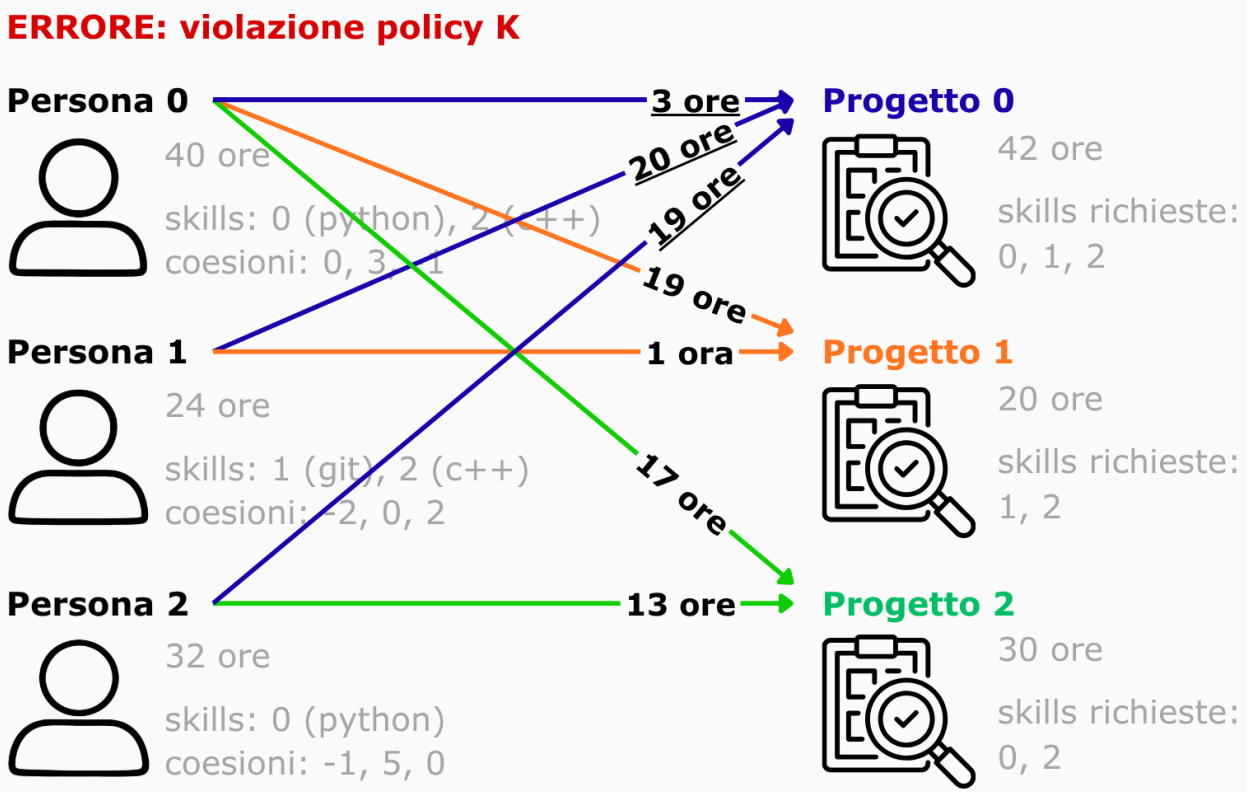


Progetto 2: tempo allocato (34) superiore a quello richiesto (30).

Esempio I - altro output errato (II)

Nell'esempio precedente, un altro output errato può essere:

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



Persona 0 assegnata a 3 progetti (supera limite massimo 2).

Note su Input e Output

Assunzioni generali input (I)

- $1 \leq P \leq 5\,000$
- $1 \leq J \leq 1\,000$
- $1 \leq S \leq 500$
- $1 \leq K \leq 40$
- $-10 \leq w_{pq} \leq 10$ per ogni coppia di persone p e q
- $w_{pp} = 0$ per ogni persona p
- $1 \leq cap_p \leq 40$ per ogni persona p
- $1 \leq dem_j \leq 3\,000$ per ogni progetto j

Note su Input

Assunzioni generali input (II)

- Le persone sono identificate dai valori $0...P-1$, i progetti dai valori $0...J-1$, le skills dai valori $0...S-1$.
- Ogni persona ha almeno una skills.
- Ogni progetto richiede almeno una skills.

Assunzioni generali output (I)

- Se ad un progetto viene assegnato più tempo di quello che richiede è considerato errore.
- Se una persona viene assegnata con 0 ore ad un progetto è considerato errore.

Casi di test

Ci sono 20 casi di test in totale:

- ★☆☆ In 6 casi su 20, $K = J$, $w_{pq} = 0$ per ogni p e q, tutte le persone possiedono la *stessa* skills e tutti i progetti richiedono *solo* quella skill;
- ★★☆ In 9 casi su 20, $w_{pq} = 0$ per ogni p e q;
- ★★★ Nel restante dei casi non ci sono particolari limitazioni.

Per la sufficienza, per esempio, potete concentravi sui 6 casi di difficoltà ★☆☆.

Limiti di tempo e memoria

I limiti di tempo e memoria sono:

- Tempo limite massimo: 3 secondi (*soft* timeout); 3.4 secondi (*hard* timeout).
- Memoria massima: 48 MB.

$\Rightarrow$ Limite di 50 sottoposizioni per gruppo.

$\Rightarrow$ Potete provare con un dataset equivalente sulla vostra macchina.

$\Rightarrow$ **Nota:** nei file di output nel dataset di esempio troverete due numeri: `worst_score` (`long long int`) e `best_score` (`long long int`). I valori che servono per calcolare il punteggio di ogni soluzione a partire da `score`.

Output

Potete stampare le soluzioni in maniera incrementale:

- Importate `the-office.h` (scaricabile da arena e dal sito).
- Man mano che migliorate la soluzione, scrivetela in output terminando la riga con `***`.
- La libreria arresterà il programma prima del timeout.

```
... include delle librerie di sistema ...
#include "the-office.h"
int main() {
  ...
}
```

Note:

- Il `main` va sempre dichiarato come `int main()` o `int main(void)`.
- Il correttore considererà l'**ultima soluzione** terminata da `***` quindi, anche se non stampate soluzioni multiple, terminate l'output con `***`.

Compilare in locale

Per testare le vostre soluzioni in locale (supponiamo che il vostro file si chiami `the-office.cpp`):

- Scaricate `grader.cpp`
- Il comando di compilazione è il seguente

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

I file `the-office.cpp`, `grader.cpp` e `the-office.h` devono essere nella stessa cartella.

Per sistemi Mac OS X e Windows vedere la nota nel testo.

Nota

Per questo esercizio è necessario usare `C++`, non è possibile usare `C`.

Punteggio (I)

Ogni caso di test vale 5 punti. Il punteggio massimo è di 100 punti.

I parametri usati nella valutazione sono i seguenti:

- `total_unmet`: il numero totale di ore non coperte nei progetti (calcolata sommando dem_j per i progetti in cui non sono coperte tutte le skills);
- `total_cohes`: la coesione totale (calcolata sommando $-|w_{pq}|$ per i progetti in cui non sono coperte tutte le skills);
- $\texttt{score} = \alpha \cdot \texttt{total\_unmet} - \beta \cdot \texttt{total\_cohes}$;
- `sum_dem`: il numero totale di ore richieste dai progetti;
- `lower_bound_cohes`: un lower bound stimato per il valore della coesione totale;
- `upper_bound_cohes`: un upper bound stimato per il valore della coesione totale.

Punteggio (II)

Punteggio finale

Per ogni caso di test per cui la vostra soluzione fornisce un output valido entro i limiti di tempo e memoria otterrete il seguente punteggio:

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

$\texttt{worst\_score} = \alpha * \texttt{sum\_dem} - \beta * \texttt{lower\_bound\_cohes}$ è un upper bound per il peggior score che può essere ottenuto.

$\texttt{best\_score} = -\beta * \texttt{upper\_bound\_cohes}$ è un lower bound per il miglior score che può essere ottenuto (ricordate che l'obiettivo è minimizzare lo $\texttt{score}$!).

Bounds

Il vostro obiettivo

Non è necessario che calcoliate il punteggio finale della vostra soluzione – vi servirebbero i bound sulle coesioni che non avete a disposizione. Ma non è un problema: quelli servono a noi per calcolare il punteggio finale.

Voi ricordate che il vostro obiettivo è **minimizzare il valore `score`**.

Esempio I - calcolo punteggio

Nel primo esempio un output aveva $\texttt{score} = -18$.

In quel caso $\texttt{sum\_dem} = 92$, $\texttt{lower\_bound\_cohes} = -8$ e $\texttt{upper\_bound\_cohes} = 20$. Calcolando le stime sui punteggi si ottengono $\texttt{worst\_score} = 668$, $\texttt{best\_score} = -60$.

E quindi

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

NB: Come vedete il punteggio è molto buono. Notate anche che, dovendo utilizzare delle stime per i punteggi migliore e peggiore, è possibile che alcuni casi non raggiungano il punteggio massimo anche trovando la soluzione ottima.

Punteggio

⇒ La **sufficienza è posta a 30 punti**.

✗ se una qualsiasi informazione data in output non è corretta (per esempio, viene violata la policy K), si ottengono 0 punti.

Punti bonus per l'esame

L'assegnazione punti avviene in maniera competitiva:

- **3 punti** ai gruppi nel primo terzile della classica (primo terzo della classifica);
- **2 punti** ai gruppi nel secondo terzile della classifica (secondo terzo della classifica);
- **1 punto** ai gruppi nel terzo terzile della classifica (ultimo terzo della classifica).

Vengono considerati nella classifica per l'assegnazione dei punti solamente i **gruppi che raggiungono la sufficienza** (punteggio maggiore o uguale a 30).

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

Consegna

Consegna: venerdì 22 maggio 2026 ore 18:00

Per caricare il vostro codice, recatevi su
`https://asdlab.disi.unitn.it/arena/`

Ricordiamo che nel calcolo del punteggio verrà considerata **l'ultima** soluzione consegnata.

Suggerimenti

Cominciate subito a lavorare al progetto per presentarvi al prossimo ricevimento (martedì 19 maggio) con tutte le domande che vorrete fare.
In ogni caso, sappiate che:

- potete venire a ricevimento
- rispondiamo su Telegram
- risponderemo alle vostre mail

Do's

È permesso:

- Discutere all'interno del gruppo
- Chiedere chiarimenti sul testo
- Chiedere opinioni su soluzioni
- Sfruttare codice fornito nei laboratori
- Utilizzare pseudocodice da libri o Wikipedia
- Richiedere aiuto (anche pesante) per la soluzione "minima"
- Venire a ricevimento
- Mandare meme (algoritmici) sul nostro canale Telegram: it's your time to shine!

Don'ts

È vietato:

- Discutere con altri gruppi
- Mettere il proprio codice su repository pubblici
- Utilizzare codice scritto da altri
- Condividere codice (abbiamo potenti mezzi!)
- Chiedere suggerimenti online (es: stackoverflow)
- Fare riferimenti a soluzioni del progetto nei meme

Ricevimento

Date e orari

- martedì 19 maggio 2026 dalle 13:30 alle 15:30 (B109);
- giovedì 21 maggio 2026 dalle 11:30 alle 13:30 (B109);
- ci saranno ulteriori ore di ricevimento online nei giorni in cui non c'è laboratorio in presenza, vi faremo sapere.

⇒ Negli orari di ricevimento saremo a disposizione sempre online, quando avrete bisogno di un aiuto scrivetelo sul gruppo telegram e risponderemo in chat privata o tramite una chiamata zoom.

⇒ Per qualsiasi domanda mandateci una mail a: `asd.disi@unitn.it` oppure contattateci su Telegram.