Algoritmi e Strutture Dati

The office (`the-office`)

Testo del problema

Slides originali su: asdlab.disi.unitn.it/asd25/prog2.pdf

Il Problema

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. 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.
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. 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.

La coesione nel lavoro. 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.

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.
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**.

Il vostro compito. 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?
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

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 (`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 la persona 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.

- 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 sezione 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:

$$\texttt{score} = \alpha \cdot \texttt{total\_unmet} - \beta \cdot \texttt{total\_cohes}. \tag{1}$$

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

Tempo non coperto:

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

Coesione totale:

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

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

Esempi

Gli esempi fanno riferimento a un unico caso di input (Figura 1), e propongono diverse soluzioni (valide o sbagliate).

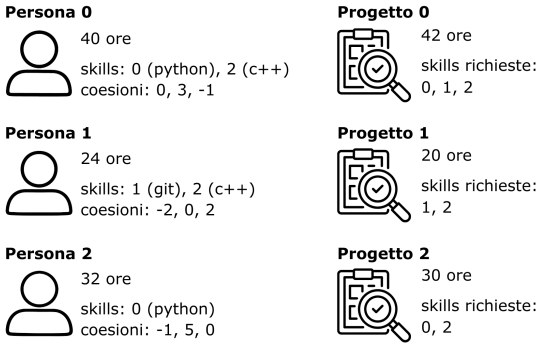

Figura 1: Esempio con $P = 3, J = 3, S = 3, K = 2$.

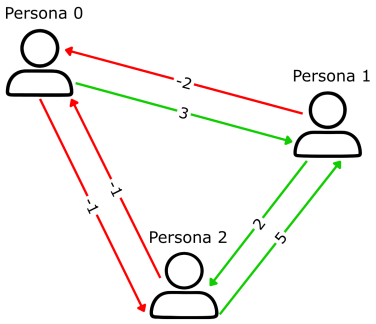

Figura 2: Coesioni tra le persone di Figura 1 visualizzate con un grafo orientato

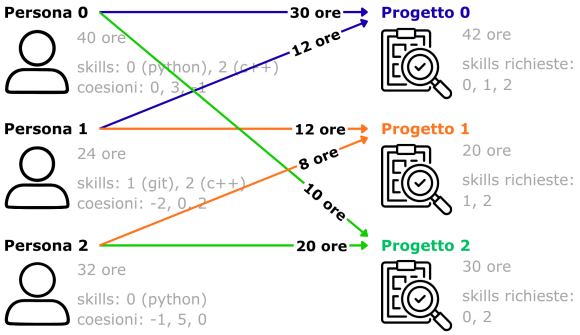

Figura 3: Soluzione valida per l'input di Figura 1 (`score` = -3)

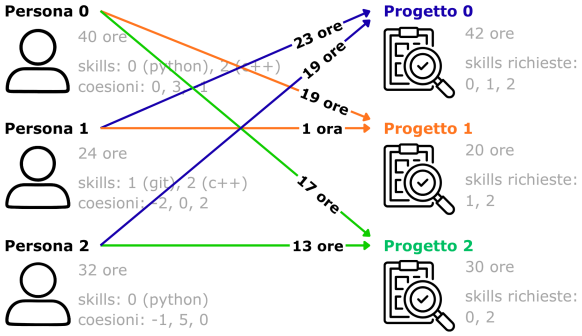

Figura 4: Soluzione valida per l'input di Figura 1 (`score` = -3)

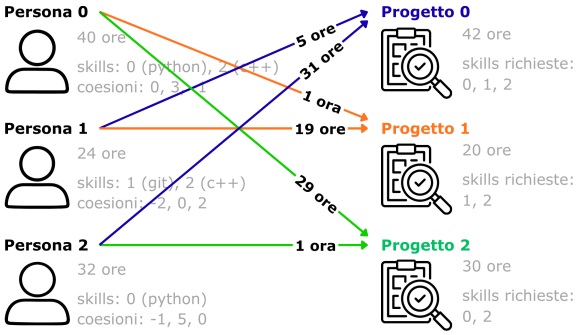

Figura 5: Soluzione valida, ma meno buona delle precedenti (non tutte le ore richieste vengono coperte), per l'input di Figura 1 (`score` = 39)

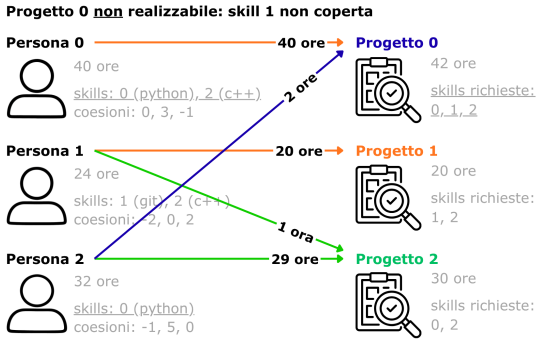

Figura 6: Soluzione valida, ma meno buona delle precedenti (non tutte le skills di un progetto vengono coperte), per l'input di Figura 1 (`score` = 294)

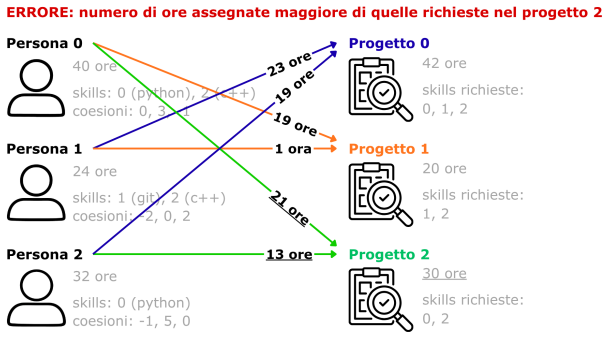

Figura 7: Proposta soluzione errata per l'input di Figura 1. Progetto 2: tempo allocato (34) superiore a quello richiesto (30).

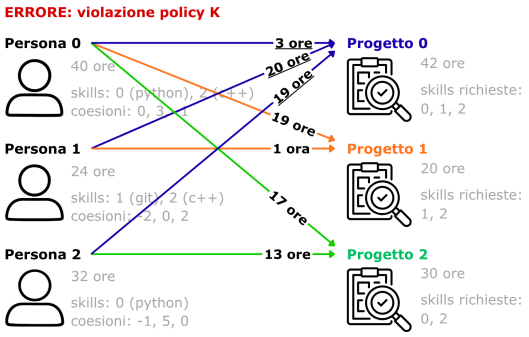

Figura 8: Proposta soluzione errata per l'input di Figura 1, violazione policy K. Persona 0 assegnata a 3 progetti (supera limite massimo 2).

Assunzioni e casi di test

Assunzioni sull'input

Parametri:

- $1 \leq P \leq 5\,000$
- $1 \leq J \leq 1\,000$
- $1 \leq S \leq 500$
- $1 \leq K \leq 40$
- $-10 \leq w_{pq} \leq 10$ 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

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 sull'output

- 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.

Limiti di tempo e memoria

- Tempo limite massimo: 3 seconds (*soft* timeout): 3.4 seconds (*hard* timeout).
- Memoria massima: 48 MB.
- Limite di 50 sottoposizioni per gruppo.

Valutazione

La sufficienza è posta a **30 punti**. La classifica è disponibile su https://asdlab.disi.unitn.it/arena/ranking/
Potete provare con un dataset equivalente sulla vostra macchina. **Nota:** nei file di output nel dataset di esempio troverete due numeri: `worst_score` (`long long int`) e `best_score` (`long long int`). II valori che servono per calcolare il punteggio di ogni soluzione a partire da `score`.

Istruzioni di compilazione

Di seguito riportiamo le istruzioni per testare i vostri programmi su vari sistemi. Si suppone che il sorgente con il vostro codice si chiami file `the-office.cpp`. I file `the-office.cpp`, `grader.cpp` e `the-office.h` devono stare nella stessa cartella.

Sistemi GNU/Linux

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

Sistemi Mac OS X

Su sistemi Mac OS X usate il seguente comando di compilazione:

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

Se ottente un errore del tipo: `use of undeclared identifier quick_exit`, sostituite in `grader.cpp` l'istruzione `quick_exit(EXIT_SUCCESS);` con `exit(EXIT_SUCCESS);`.

Sistemi Windows

Per il sistema Windows 10 potete installare il "Windows Subsystem for Linux"[1]. Successivamente potete installare i tool necessari per usare Visual Studio Code[2] o Visual Studio 2017[3] seguendo le relative guide riportate nelle note. Usando questo sistema fate attenzione a dove salvate i file e a quale nome gli date in quanto potreste avere delle difficoltà con percorsi che contengano spazi e caratteri speciali.
In alternativa, o per sistemi precedenti a Windows 10 potete installare *Cygwin*[4], un ambiente completamente POSIX-compatibile per Windows. Anche in questo caso esistono guide per configurare i comuni editor disponibili su Windows di modo che utilizzino l'ambiente Cygwin, come per esempio Visual Studio[5].
Una volta installato Cygwin è possibile simulare quanto avviane su arena compilando il proprio sorgente senza includere l'header `the-office.h` e il grader `grader.cpp`:

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

e lanciare il comando come:

```
timeout.exe 3 ./the-office
```

`timeout.exe` arresterà il programma dopo 3 secondi.

[1] `https://docs.microsoft.com/en-us/windows/wsl/install-win10`
[2] `https://code.visualstudio.com/docs/cpp/config-wsl`
[3] `https://devblogs.microsoft.com/cppblog/targeting-windows-subsystem-for-linux-from-visual-studio/`
[4] `https://www.cygwin.com/`
[5] `https://devblogs.microsoft.com/cppblog/using-mingw-and-cygwin-with-visual-cpp-and-open-folder/`

Esempi di input/output

File `input.txt`	File `output.txt`

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

File `input.txt`	File `output.txt`

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

File `input.txt`	File `output.txt`

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

File `input.txt`	File `output.txt`

`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`
	`-6`
	`0 3`
	`0 1 2`
	`10 23 9`
	`1 2`
	`1 2`
	`1 19`
	`2 1`
	`0`
	`30`
	`***`