Algoritmi e Strutture Dati

Il grande conclave algoritmico (`conclave`)

Testo del problema

Slides originali su: asdlab.disi.unitn.it/slides/asd24/prog2.pdf

Il Grande Conclave

Nel profondo delle austere mura del Sacrum Algorithmicum Collegium (SAC), una congregazione di scienziati che influenzano il destino computazionale del mondo, un antico rituale si sta per compiere: il Grande Conclave Accademico.
Ogni 1000_2 anni, quando i cicli delle conferenze si allineano e il calendario degli esami si fa insostenibile, le menti più brillanti (ma anche più litigiose) dell'algoritmica si riuniscono per eleggere una figura leggendaria: il nuovo *P.A.P.A. (Pater Algorithmorum Polinomiorum Approximantis)*, il supremo rappresentante del SAC, incaricato di guidare la comunità verso nuove frontiere dell'ottimizzazione computazionale.

Il sogno del Prof. A.

Il Prof. A, stimato docente di Algoritmi, ha deciso che è giunto il momento di raggiungere il suo massimo traguardo. Non il premio Turing, non la cattedra onoraria a Princeton… ma qualcosa di molto più ambizioso. Vuole diventare P.A.P.A.

Le riunioni delle fazioni

Tuttavia, non è un'impresa semplice. Il Conclave non prevede votazioni trasparenti, né campagne ufficiali. Al contrario, tutto si gioca nelle riunioni riservate delle fazioni algoritmiche, piccoli e potenti gruppi che si ritrovano a porte chiuse, dibattendo tra una dimostrazione e l'altra su chi meriti di salire al vertice.
Le riunioni sono microcosmi accademici, e uniscono Votanti con diverse idiosincrasie:

- i Greedy Monks, convinti che ogni decisione vada presa localmente;
- i NP-Hardliners, che ritengono che tutto sia troppo complesso per essere deciso;
- gli Heuristic Clerics, non credono nella teoria, ma le loro soluzioni funzionano.

Ogni Votante può partecipare a multiple riunioni di diversa composizione durante il Conclave.

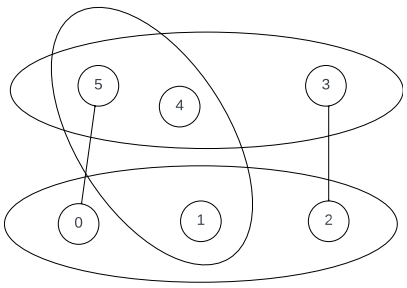


Figura 1: Conclave composto da 6 Votanti e 5 riunioni. Votanti nello stesso insieme o legati da un link formano una riunione.

Come influenzare il Conclave?

Il Prof. A. sa bene che non potrà presenziare a tutte queste riunioni, e lavorando da solo sarà difficile avere abbastanza influenza.
Ma sa anche un'altra cosa: **basta una sola voce amica in ogni riunione per orientare la discussione**. Per questo, punterà a conquistare alcuni individui chiave, tra membri fidati, ex studenti, compagni di dipartimento, che saranno scelti in maniera strategica data la composizione delle varie riunioni.

E ora? Il vostro compito

Le risorse del Prof. A sono limitate: **vuole a tutti i costi minimizzare il numero di Votanti da assoldare**.
Il vostro compito è molto semplice: a partire dalla lista e dalla composizione delle riunioni, selezionate il **minor numero possibile di persone da influenzare**, in modo che **ogni riunione abbia almeno un Votante fedele** al Prof. A.

Esempi

Esempio I - solo riunioni tra due votanti

- 7 votanti, 12 riunioni
- **Tutte le riunioni coinvolgono esattamente** 2 **votanti.**
- Rappresentiamo la situazione come un grafo in cui i vertici sono i votanti e gli archi uniscono votanti coinvolti insieme in una riunione.

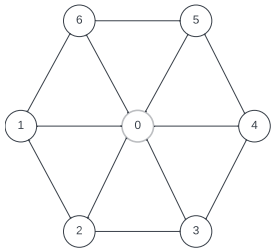
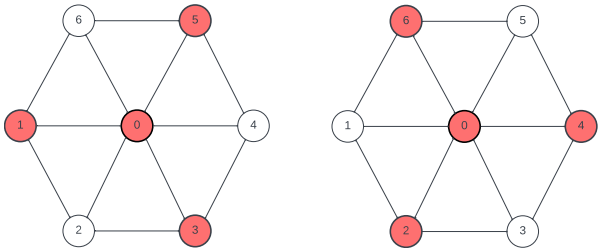


Figura 2: Esempio I - Tutte le riunioni coinvolgono 2 votanti.

Nell'esempio precedente, l'input che trovereste sarebbe il seguente:

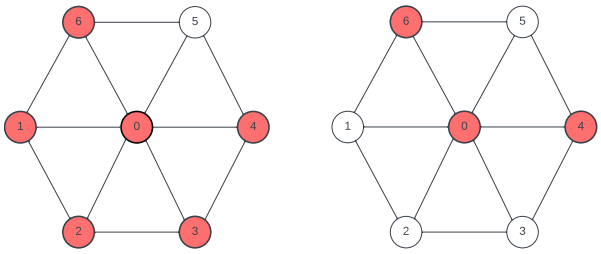
```
7 12
2 0 1
2 0 2
2 0 3
2 0 4
2 0 5
2 0 6
2 1 2
2 1 6
2 2 3
2 3 4
2 4 5
2 5 6
```



(a) Una soluzione ottima

(b) Altro esempio di soluzione ottima

Figura 3: Esempio I - Soluzioni ottime



(a) Soluzione subottima

(b) Soluzione errata, le riunione tra 1 e 2 e tra 2 e 3 non sono influenzate

Figura 4: Esempio I - Soluzione subottima e soluzione errata

Nell'esempio precedente, un output valido (selezioniamo almeno un vertice in ogni arco, quindi almeno un votante assoldato per ogni riunione) sarebbe il seguente:

```
4 0 1 3 5 #
```

Che è anche ottimo, perché togliendo qualsiasi dei vertici rossi, non copriremmo più almeno una riunione.
Possiamo osservare che la precedente non è l'unica soluzione ottima, ne esistono altre:

```
4 0 2 4 6 #
```

A parità di vertici selezionati, ci basta una soluzione qualsiasi.
Un altro output valido potrebbe essere il seguente:

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

Che è peggiore del precedente (in quanto ho bisogno di assoldare più votanti), ma comunque valido.
Nell'esempio precedente, un **output errato** sarebbe il seguente:

```
3 0 4 6 #
```

Infatti, esistono riunioni (archi) che non sono coperti da nessun vertice in rosso, come ad esempio l'arco 1-2 o 2-3.

Esempio II - riunioni di dimensione arbitraria

- 6 votanti, 5 riunioni.
- Visualizziamo i votanti come vertici e le riunioni come insiemi generici (o archi, se riunioni tra 2 votanti).
- Per esempio, 0 e 5 hanno fatto una riunione insieme.
- Ma anche 3, 4 e 5 hanno fatto una riunione insieme.

L'input che trovereste sarebbe il seguente:

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

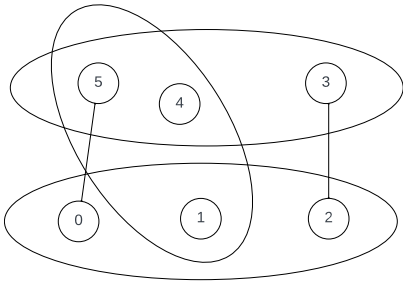
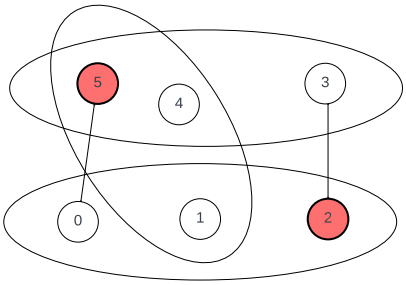
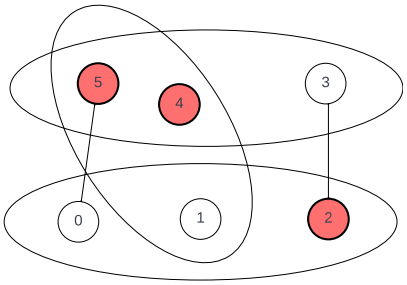


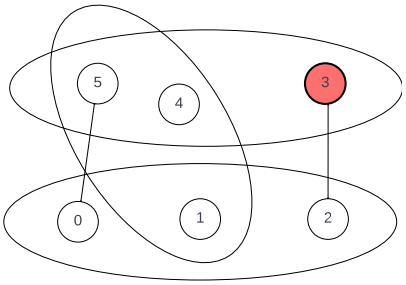
Figura 5: Esempio II - Istanza con riunioni di dimensione arbitraria.



(a) Soluzione ottima



(b) Soluzione subottima



(c) Soluzione errata, la riunione di $1, 4, 5$ non è influenzata

Figura 6: Esempio II

Un output valido e ottimo sarebbe il seguente:

```
2 2 5 #
```

Infatti assoldando solo questi 2 votanti siamo in grado di avere sempre un votante in ogni riunione. Non si può far meglio di così. Per esempio, lasciando fuori il 2 non copriremmo tra le altre la riunione tra 2 e 3.
Un output valido ma subottimo sarebbe il seguente:

```
3 2 4 5 #
```

Questo output è valido perché ogni riunione è coperta, tuttavia aver assoldato anche il votante 4 è **ridondante** in quanto tutte le riunioni a cui partecipa erano già coperte dal votante 5.
Un **output errato** invece sarebbe il seguente:

```
1 3 #
```

Questo output non è valido in quanto esistono riunioni non coperte da nessun vertice rosso, per esempio l'insieme 1, 4, 5.

Esempio III - riunioni di dimensione arbitraria

- 9 votanti, 5 riunioni.
- Visualizziamo i votanti come vertici e le riunioni come insiemi generici.
- Per esempio 3, 6, 7 e 8 hanno fatto una riunione insieme.

L'input che trovereste sarebbe il seguente:

```
9 5
3 0 1 2
3 1 2 3
3 4 5 6
3 4 7 8
4 3 6 7 8
```

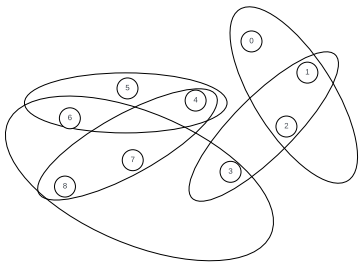


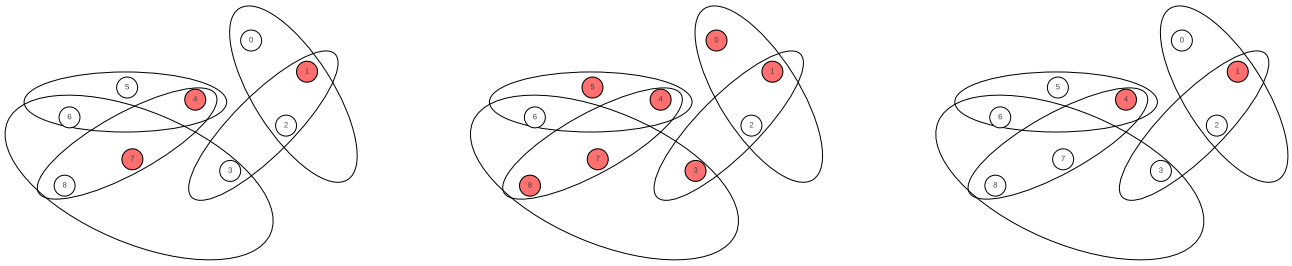
Figura 7: Esempio III - Ancora riunioni di dimensione arbitraria

Un output valido e ottimo sarebbe il seguente:

```
3 1 4 7 #
```

Infatti assoldando solo questi 3 votanti siamo in grado di avere sempre un votante in ogni riunione. Non si può far meglio di così.
Un output valido ma subottimo sarebbe il seguente:

```
7 0 1 3 4 5 7 8 #
```



(a) Soluzione ottima

(b) Soluzione subottima

(c) Soluzione errata, la riunione $3, 6, 7, 8$ non è influenzata

Figura 8: Esempio III

Ovviamente c'è una certa ridondanza qui.
Un **output errato** sarebbe il seguente:

```
2 1 4 #
```

Infatti la riunione con 4 elementi non viene coperta da nessuno.

Input/Output

Input: Un file con $1 + M$ righe.

- La prima riga riporta 2 numeri interi: N (`int`) e M (`int`), rispettivamente il numero di votanti e il numero di diverse riunioni tra i votanti.
- Le successive M righe descrivono le riunioni. Ogni riga i riporta un intero R_i (`int`), rappresentante il numero di votanti coinvolti nella riunione i-esima, e R_i valori interi (`int`) rappresentanti gli identificativi dei votanti coinvolti nella riunione i-esima.

Output: Un file con le vostre proposte di soluzione, ogni proposta è composta da 1 riga:

- La riga deve riportare il valore di S (`int`) individuato, ovvero il numero di votanti assoldati;
- sulla stessa riga deve poi contenere altri S valori (`int`), ovvero gli identificativi dei votanti assoldati in modo da coprire ogni riunione;
- la riga deve terminare con #

Limiti e assunzioni

- $1 < N \leq 15\,000$.
- $1 < M \leq 20\,000$.
- $1 \leq R_i \leq 3\,000$.
- I votanti sono identificati rispettivamente da interi da 0 a $N - 1$.
- Le riunioni sono costituite sempre da almeno due Votanti.
- Non esistono riunioni doppie, ovvero due riunioni differiscono sempre per almeno un Votante.
- Potenzialmente, alcune persone potrebbero non partecipare a nessuna riunione.

Casi di test

Ci sono 20 casi di test in totale:

- In 6 casi su 20, $N \leq 20$;
- In 4 casi su 20, $R_i = 2$ per ogni riunione i;
- Nel restante dei casi non ci sono particolari limitazioni.

Limiti di tempo e memoria

- Tempo limite massimo: 3 secondi (timeout: 3.1 secondi).
- Memoria massima: 128 MB.
- Limite di 40 sottoposizioni per gruppo.
- Potete provare con un dataset equivalente sulla vostra macchina
- **Nota:** nei file di output troverete due numeri: il lower bound minS (`int`) e la soluzione ottima optS (`int`). La soluzione ottima è disponibile solo per i casi di test con $N \leq 24$, negli altri casi troverete -1.

Punteggio

Ogni caso di test vale 5 punti. Il punteggio massimo è di 100 punti. I parametri di valutazione sono i seguenti:

- S: il numero di Votanti assoldati nel vostro output;
- minS: lower bound al numero di Votanti tali che ogni ad riunione partecipa almeno un votante. Quando è disponibile la soluzione ottima optS, questa viene usata come lower bound: minS = optS.
- maxS = $\min(N, M)$: upper bound al numero di Votanti tali che ogni ad riunione partecipa almeno un votante.

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:

$$\mathcal{P} = 5.0 \times \begin{cases} 0 & \text{se } S \geq \max S \text{ o } S < \min S \\ 1 & \text{se } S = \min S \\ 1 - \dfrac{S - \min S}{\max S - \min S} & \text{se } \min S < S < \max S \end{cases}$$

Bounds

Lower bound. Non è necessario che calcoliate il punteggio della vostra soluzione – vi servirebbe il bound min$S$ che non avete. Il vostro obiettivo è in ogni caso di **minimizzare il numero di Votanti assoldati**.
Upper bound Il numero di Votanti assoldati è dominato sicuramente dal numero N di Votanti totali. Tuttavia, la vostra soluzione S è dominata anche dal numero di riunioni M, in quanto potrei assoldare (stando larghi) un diverso Votante per ogni riunione. Per ogni dataset, abbiamo centrato in 0 il punteggio tenendo conto di questi due valori e prendendo il minimo.

Esempio di calcolo del punteggio

In questo esempio avevamo come output:

```
2 2 5#
```

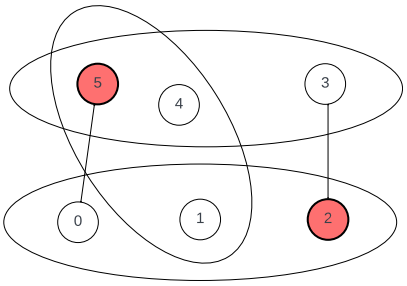
Che come abbiamo visto è valido e ottimo.
In questo caso \$S = \$ $\min S = 1$, quindi $\mathcal{P} = 5 \times 1 = 5$
In questo output dallo stesso esempio precedente:

```
3 2 4 5 #
```

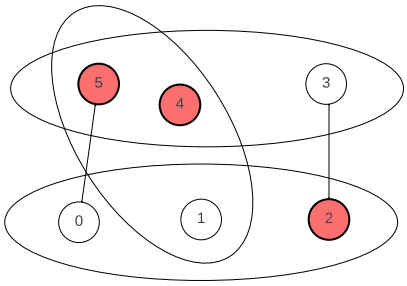
abbiamo una soluzione subottima, infatti ne esisterebbe una di cardinalità 2. Siamo nella situazione $\min S < S < \max S$ dove $\min S = 2$, $S = 3$ e $\max S = \min(6, 5)$, quindi $\mathcal{P} = 1 - \frac{3-2}{\min(6,5)-2} = 1 - \frac{1}{3} = 0.67$. % e porta punti in classifica.
In questo caso, diamo in output una soluzione composta da solo un Votante assoldato:

```
1 3 #
```

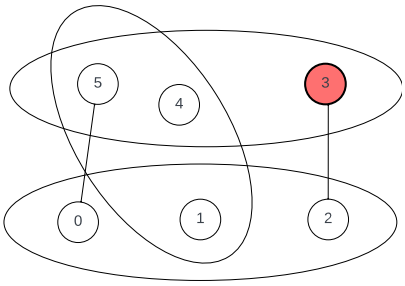
Abbiamo che esiste almeno una riunione non coperta (e.g., $0 - 5$) ed infatti $S < \min S$, quindi il punteggio è $\mathcal{P} = 0$.



(a) Esempio di calcolo del punteggio di una soluzione ottima.



(b) Esempio di calcolo del punteggio di una soluzione subottima.



(c) Esempio di calcolo del punteggio di una soluzione che non copre tutte le riunioni.

Valutazione

La sufficienza è posta a **30 punti**.
Classifica: https://asdlab.disi.unitn.it/arena/ranking/

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 `conclave.cpp`. I file `conclave.cpp`, `grader.cpp` e `conclave.h` devono stare nella stessa cartella.

Sistemi GNU/Linux

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o conclave conclave.cpp grader.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 conclave conclave.cpp grader.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 `conclave.h` e il grader `grader.cpp`:

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

e lanciare il comando come:

```
timeout.exe 3 ./conclave
```

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

Esempi di input/output

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

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

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

`7 12`
`2 0 1`
`2 0 2`
`2 0 3`
`2 0 4`
`2 0 5`
`2 0 6`
`2 1 2`
`2 1 6`
`2 2 3`
`2 3 4`
`2 4 5`
`2 5 6`
	`4 0 1 3 5 #`

[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/`

File input.txt	File output.txt
6 5 3 3 4 5 3 0 1 2 2 0 5 2 2 3 3 1 4 5	3 2 4 5 #

File input.txt	File output.txt
9 5 3 0 1 2 3 1 2 3 3 4 5 6 3 4 7 8 4 3 6 7 8	2 1 4 # 3 7 6 2 #

File input.txt	File output.txt
6 15 3 0 3 4 4 0 1 3 4 4 0 1 2 4 5 1 2 3 4 5 5 0 2 3 4 5 2 1 4 4 0 2 3 4 2 0 5 2 2 3 3 2 4 5 3 0 1 4 3 1 2 4 5 0 1 2 3 4 3 2 3 4 2 3 5	3 0 3 4 #