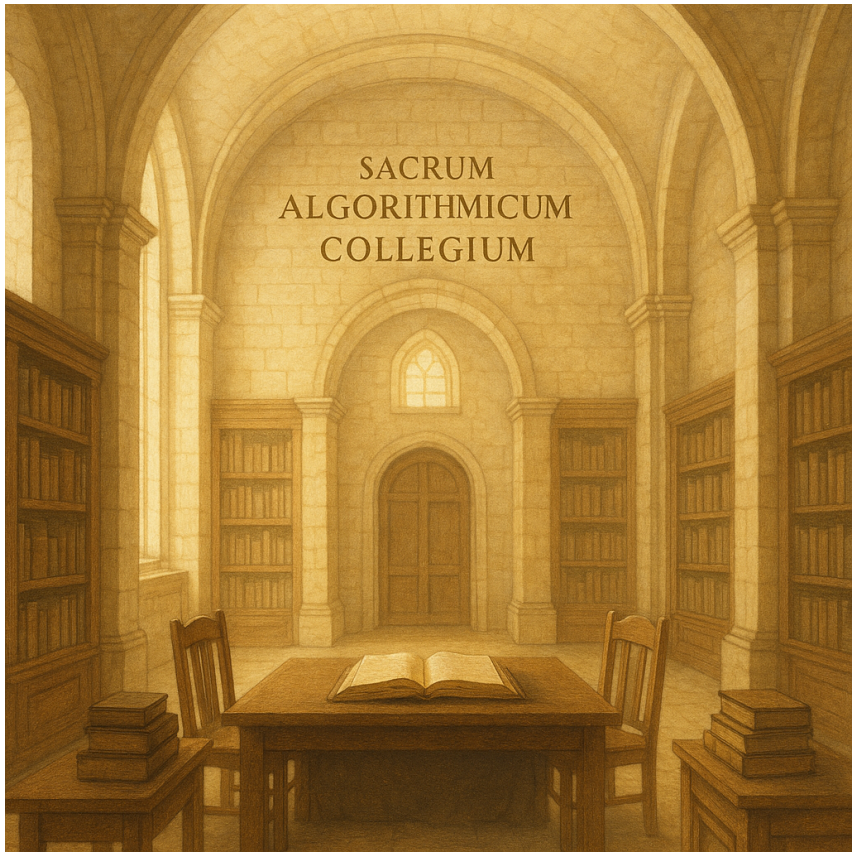
Secondo Progetto ASD 2024/2025

Il Grande Conclave Accademico
P=NP

Il Grande Conclave (I)

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.



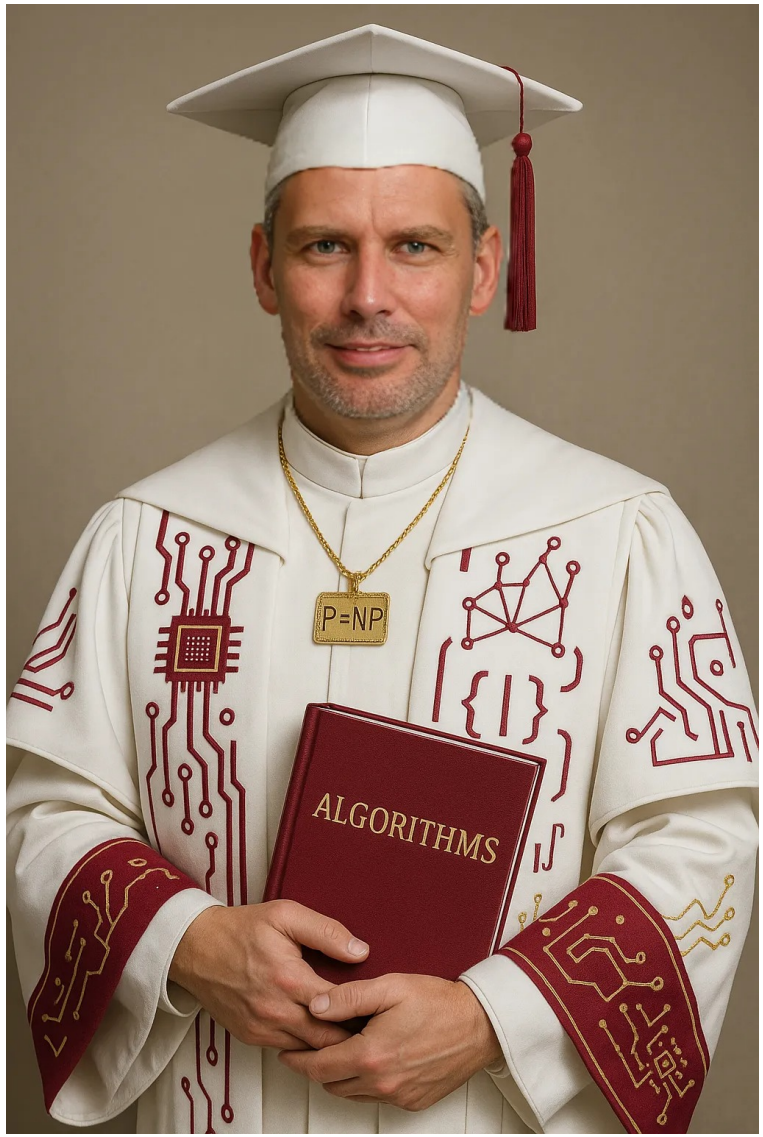
Il Grande Conclave (II)

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 (I)

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 delle fazioni (II)

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...

NB: ogni Votante può partecipare a multiple riunioni di diversa composizione durante il Conclave.

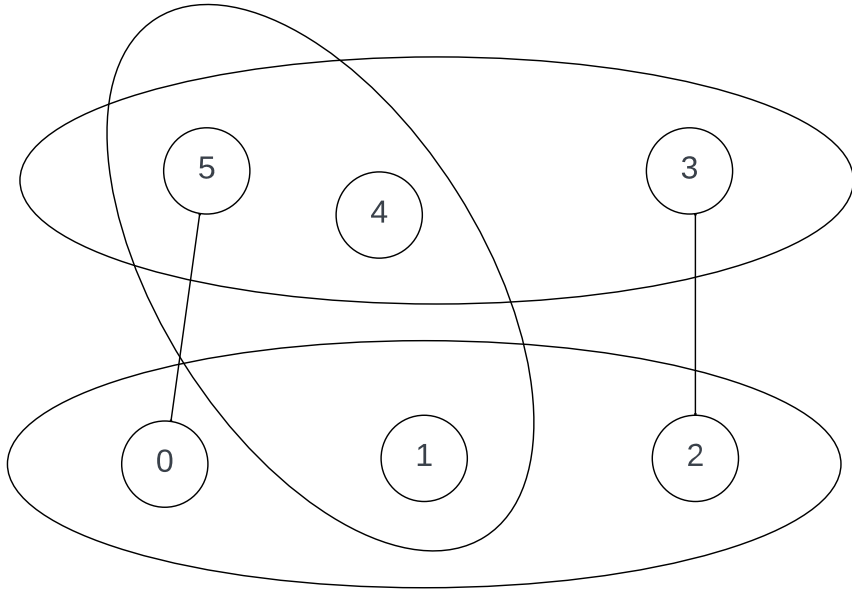


Figura: Conclave composto da 6 Votanti e 5 riunioni. Votanti nello stesso insieme o legati da un link formano una riunione.

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.

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.

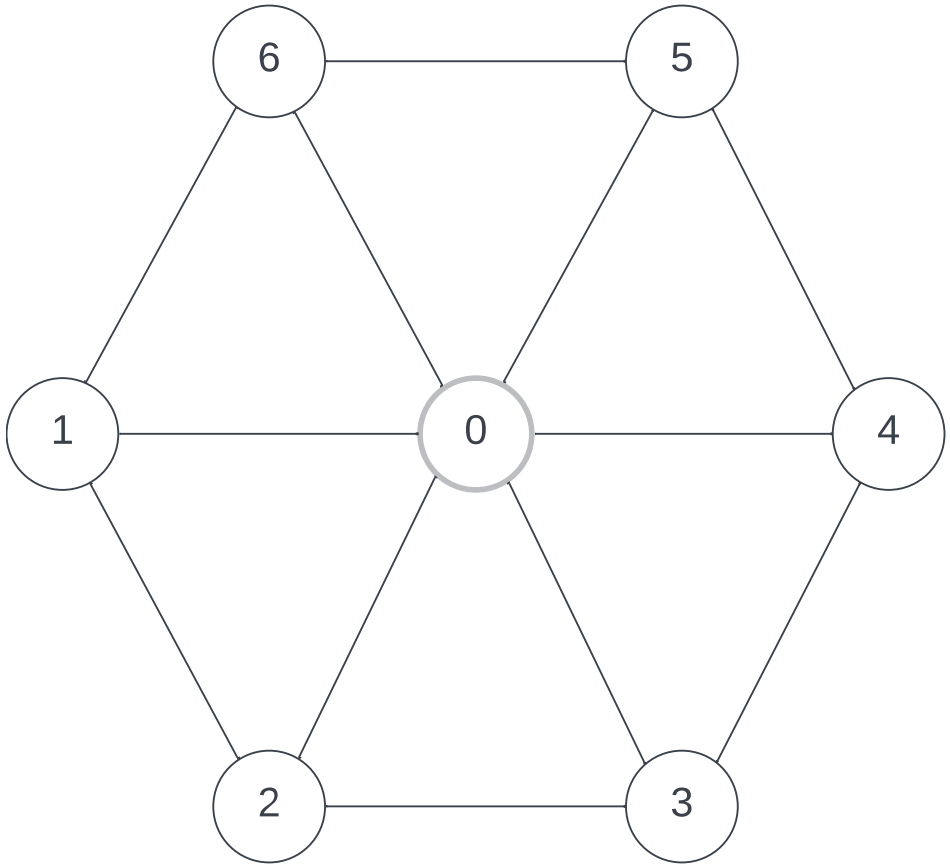
Output

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 #

Esempio I - solo riunioni tra due votanti

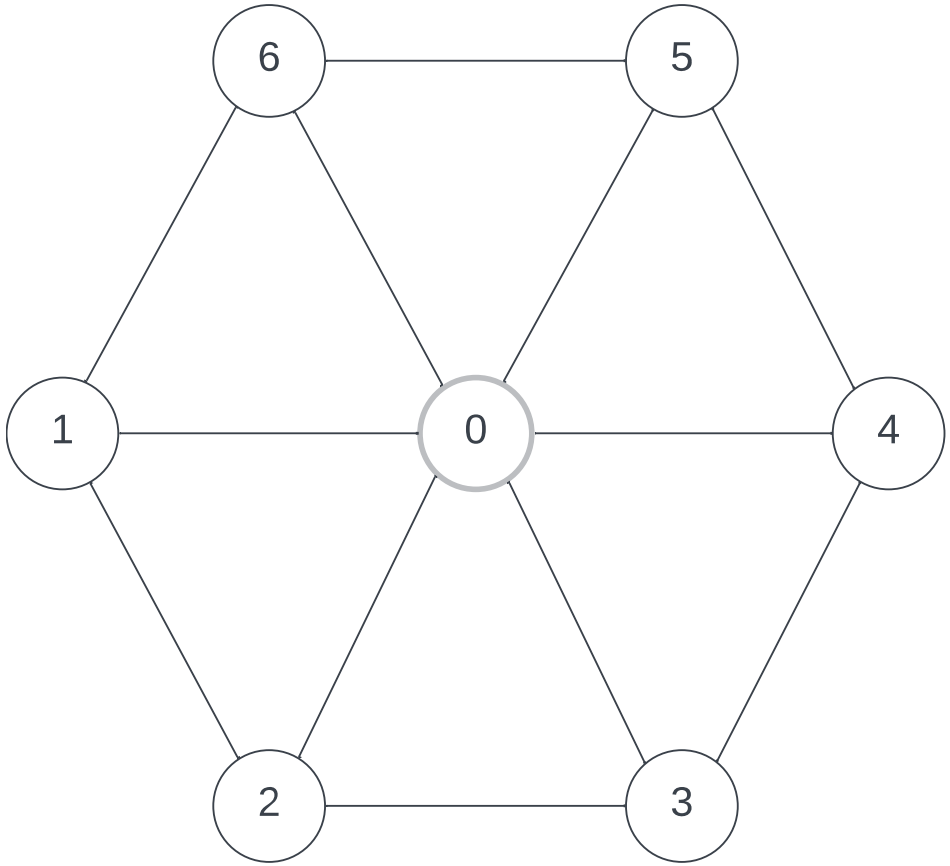
- 7 votanti, 12 riunioni
- **Tutte le riunioni coinvolgono esattamente** 2 **votanti.**
- Rappresentiamo la situazione come un grafo in cui i vertici sono i votanti e gli archi uniscono votanti coinvolti insieme in una riunione.



Esempio I - input

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
```

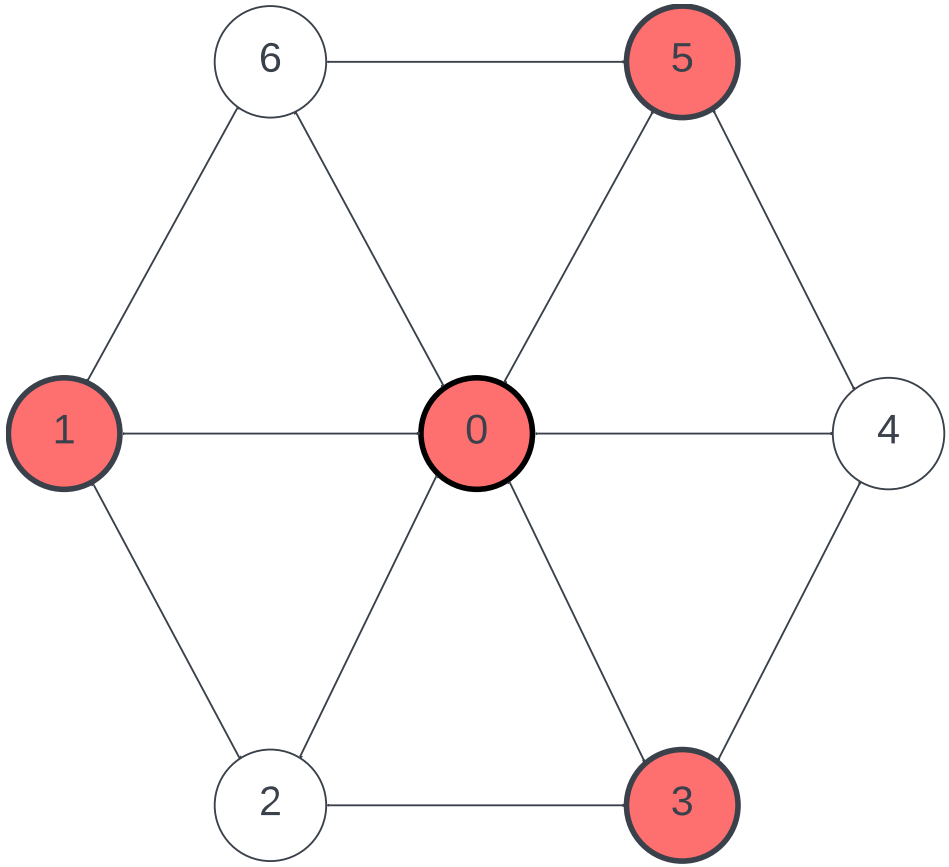


Esempio I - output valido e ottimo

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.

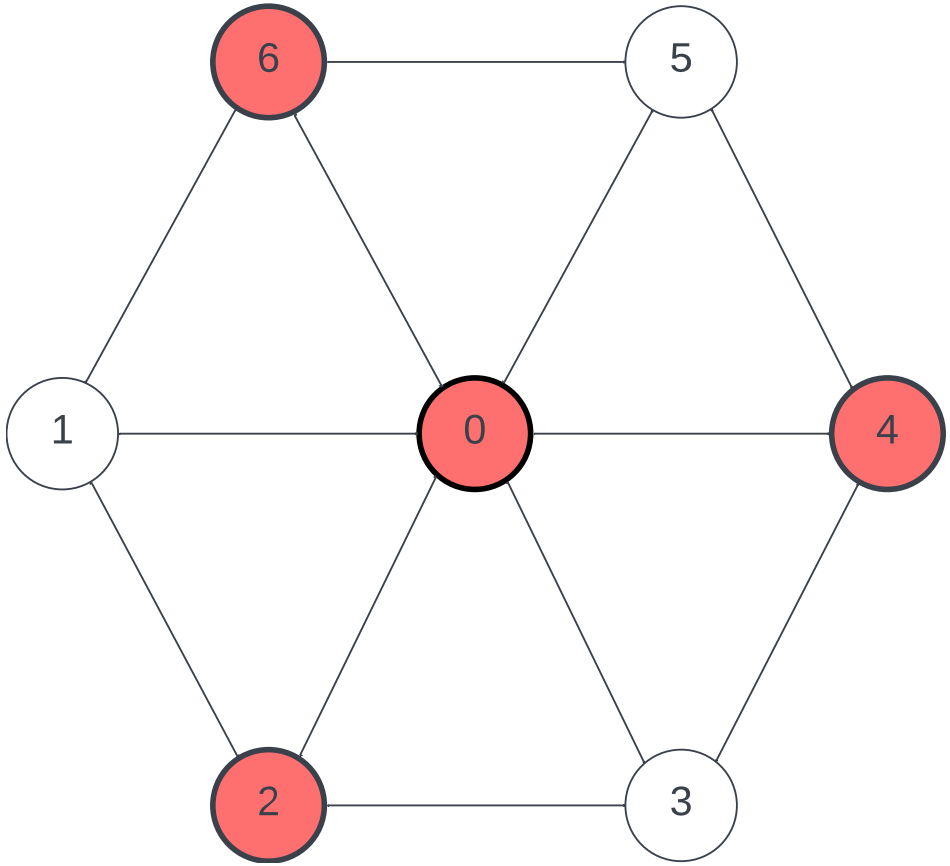


Esempio I - altro output valido e ottimo

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.

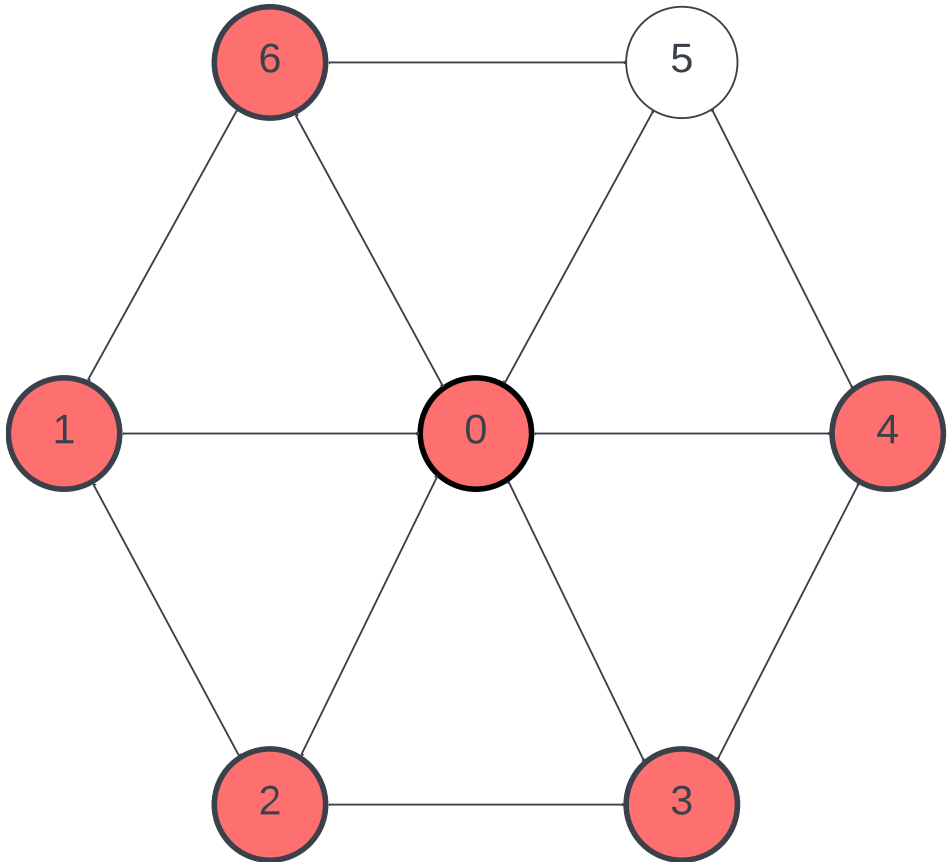


Esempio I - output valido ma subottimo

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.

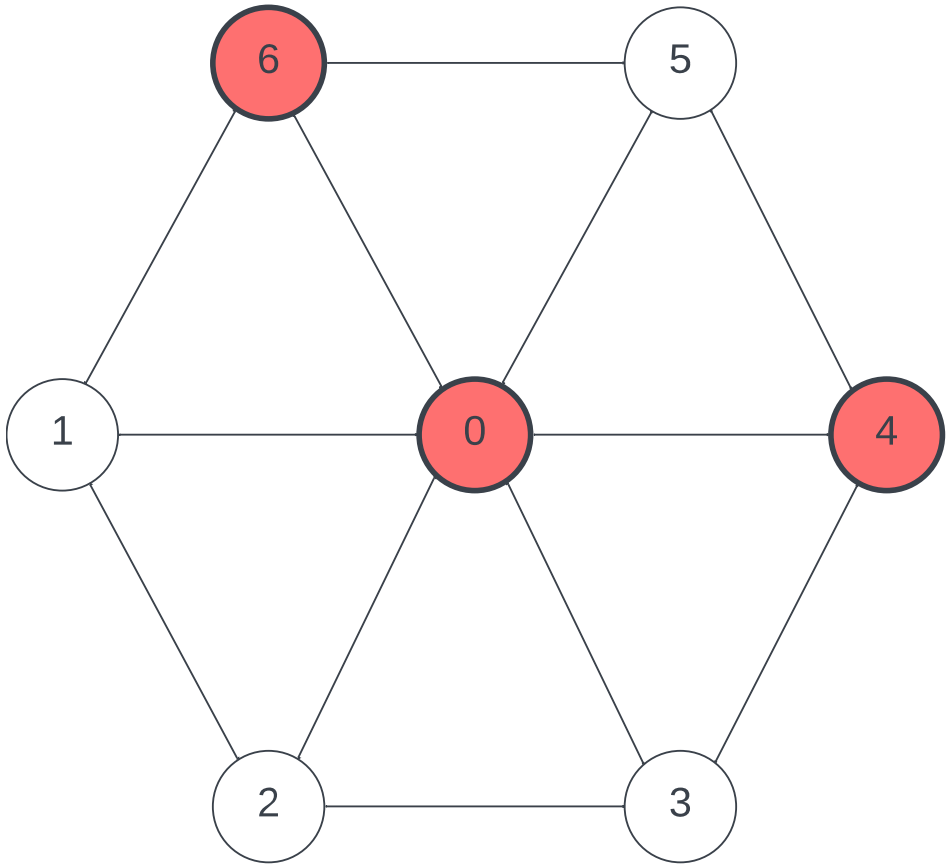


Esempio I - output errato

Nell'esempio precedente, un **output errato** sarebbe il seguente:

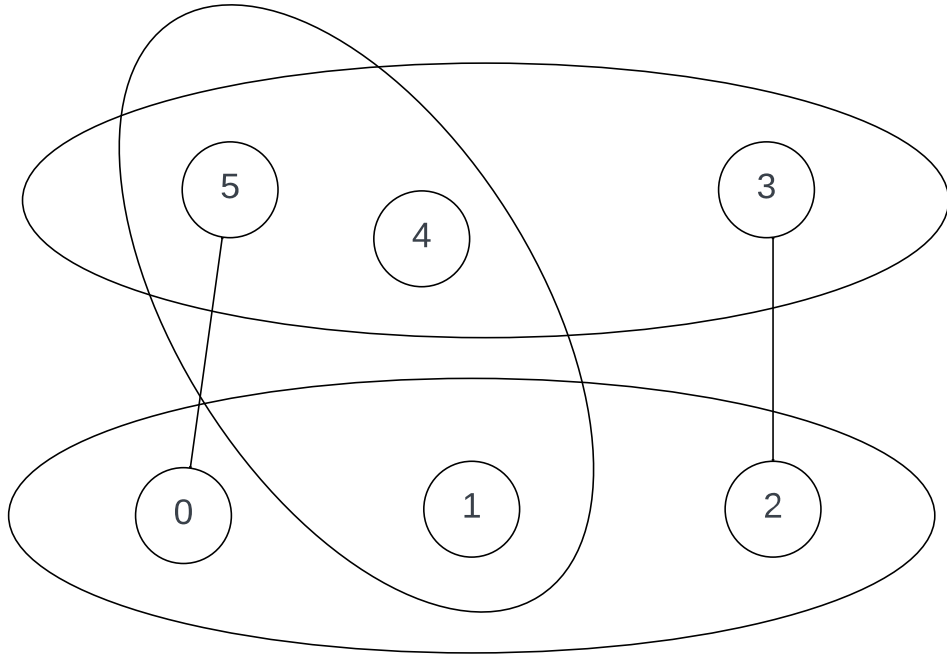
```
3 0 4 6 #
```

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

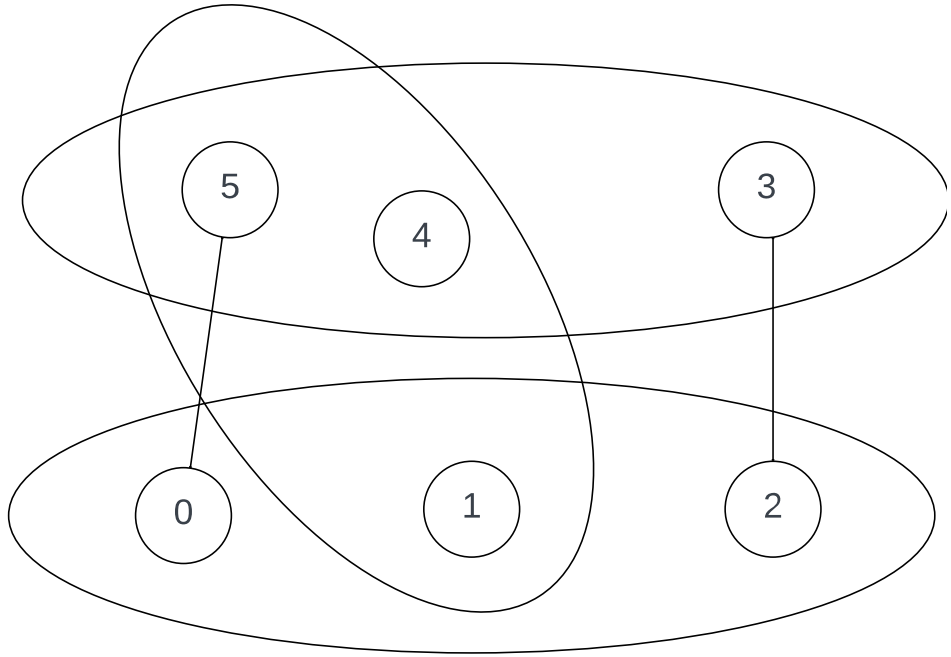
- 6 votanti, 5 riunioni.
- Visualizziamo i votanti come vertici e le riunioni come insiemi generici (o archi, se riunioni tra 2 votanti).
- Per esempio, 0 e 5 hanno fatto una riunione insieme.
- Ma anche 3, 4 e 5 hanno fatto una riunione insieme.



Esempio II - input

L'input che trovereste sarebbe il seguente:

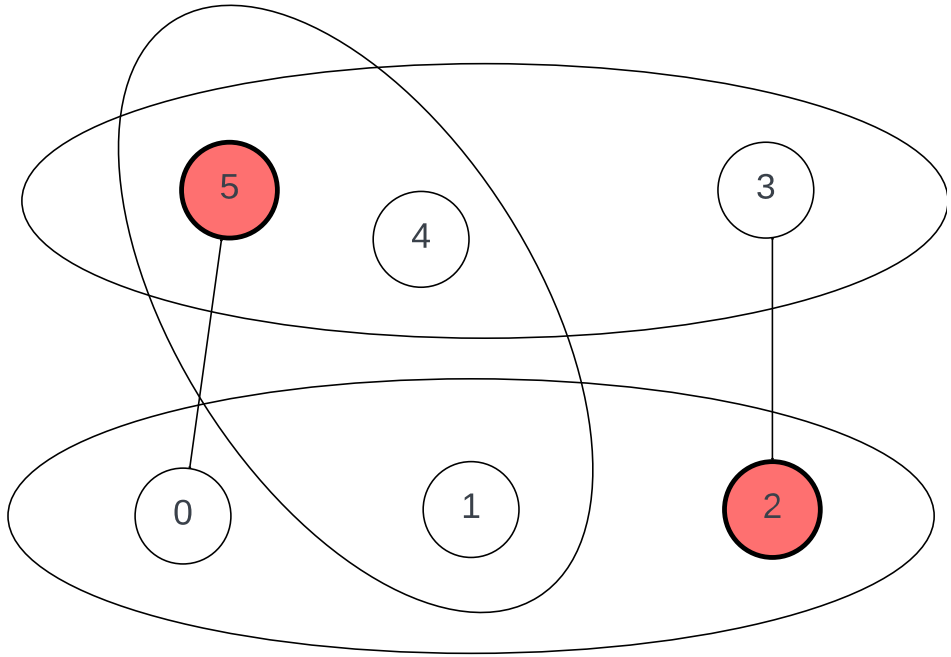
```
6 5
3 3 4 5
3 0 1 2
2 0 5
2 2 3
3 1 4 5
```



Esempio II - output ottimo

Un output valido e ottimo sarebbe il seguente:

```
2 2 5 #
```

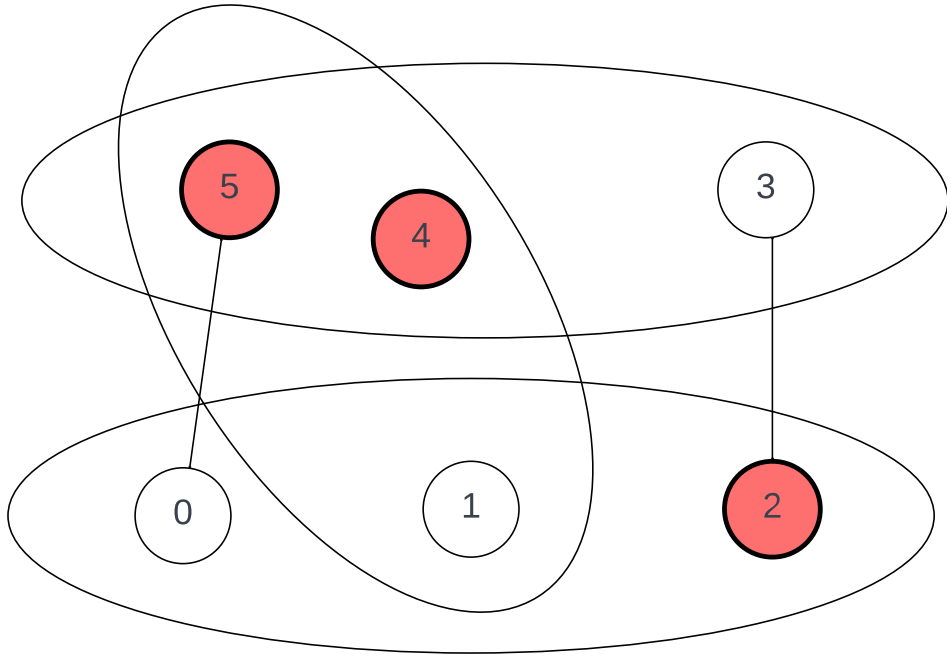


Infatti assoldando solo questi 2 votanti siamo in grado di avere sempre un votante in ogni riunione. Non si può far meglio di così. Per esempio, lasciando fuori il 2 non copriremmo tra le altre la riunione tra 2 e 3.

Esempio II - output valido ma subottimo

Un output valido ma subottimo sarebbe il seguente:

```
3 2 4 5 #
```



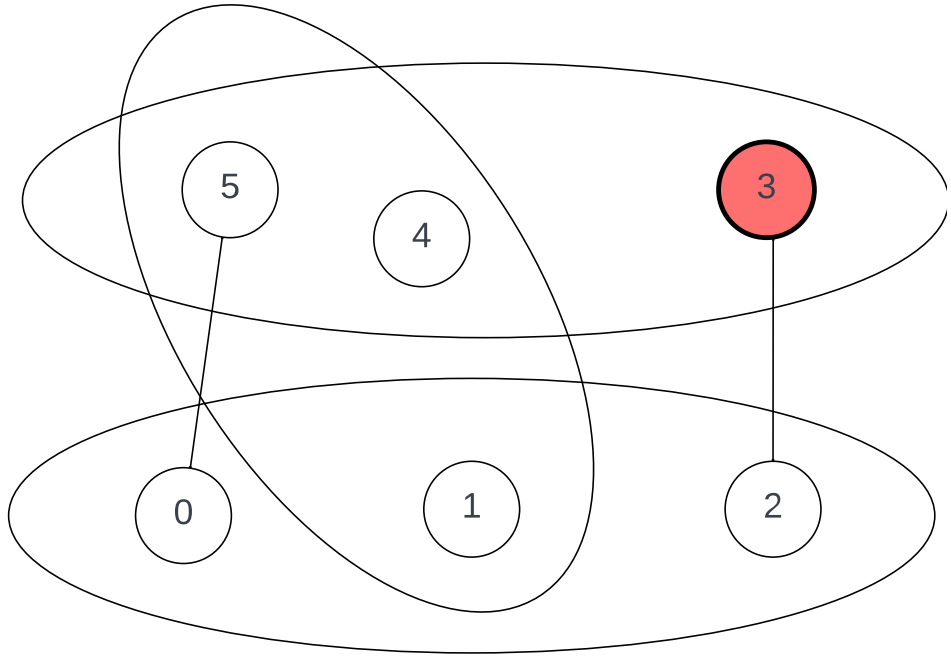
Questo output è valido perché ogni riunione è coperta, tuttavia aver assoldato anche il votante 4 è **ridondante** in quanto tutte le riunioni a cui partecipa erano già coperte dal votante 5.

Esempio II - output errato

Un **output errato** invece sarebbe il seguente:

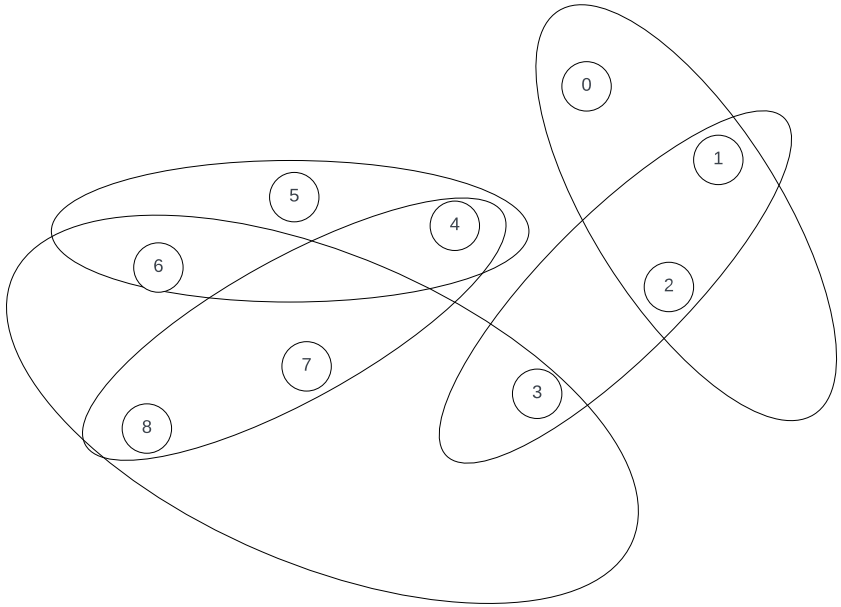
```
1 3 #
```

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

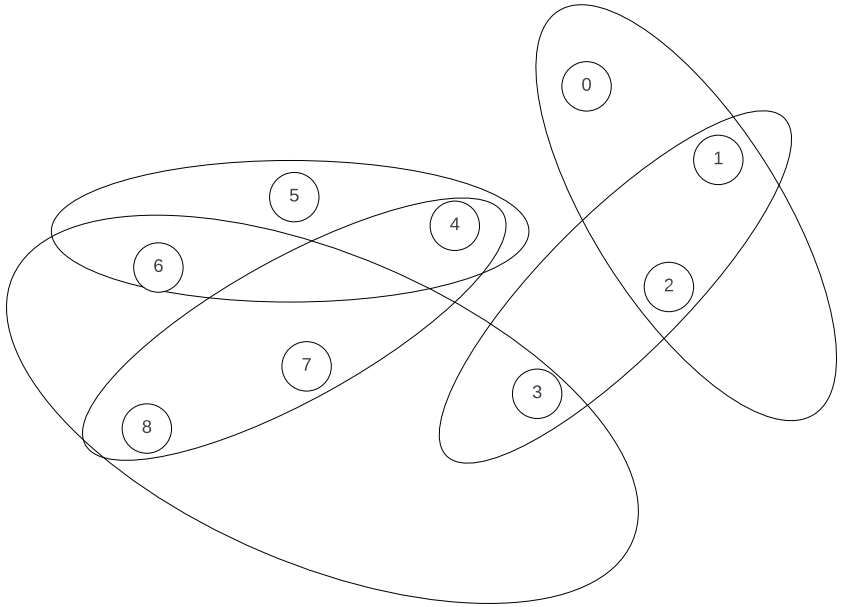
- 9 votanti, 5 riunioni.
- Visualizziamo i votanti come vertici e le riunioni come insiemi generici.
- Per esempio 3, 6, 7 e 8 hanno fatto una riunione insieme.



Esempio III - input

L'input che trovereste sarebbe il seguente:

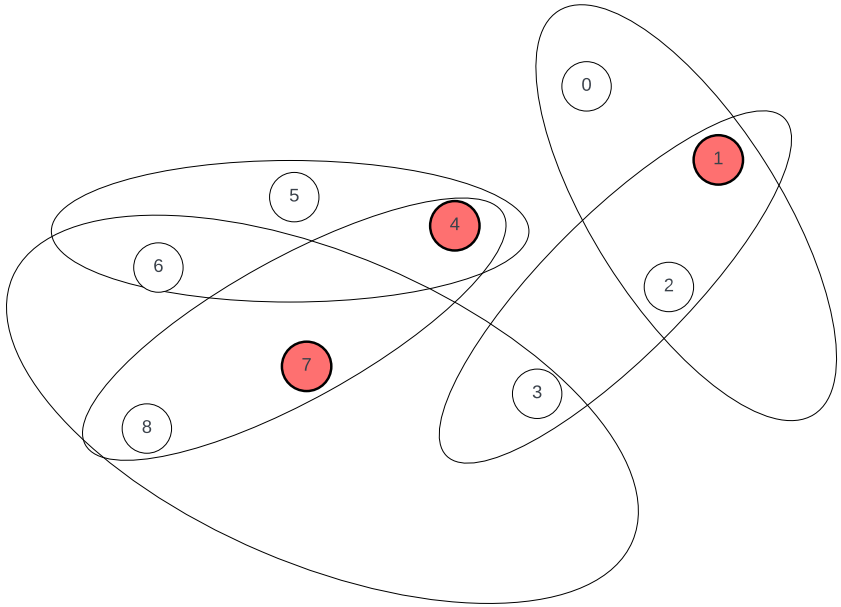
```
9 5
3 0 1 2
3 1 2 3
3 4 5 6
3 4 7 8
4 3 6 7 8
```



Esempio III - output ottimo

Un output valido e ottimo sarebbe il seguente:

```
3 1 4 7 #
```

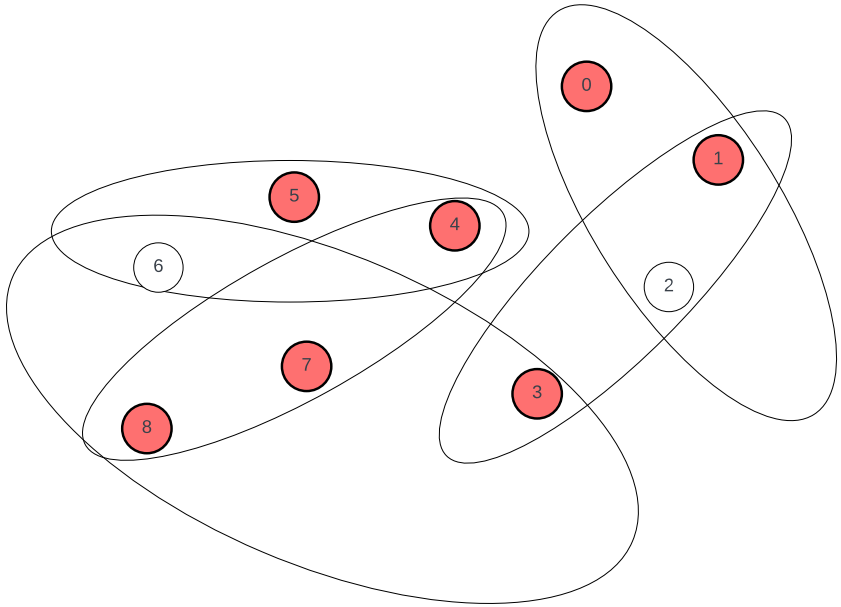


Infatti assoldando solo questi 3 votanti siamo in grado di avere sempre un votante in ogni riunione. Non si può far meglio di così.

Esempio III - output valido ma subottimo

Un output valido ma subottimo sarebbe il seguente:

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



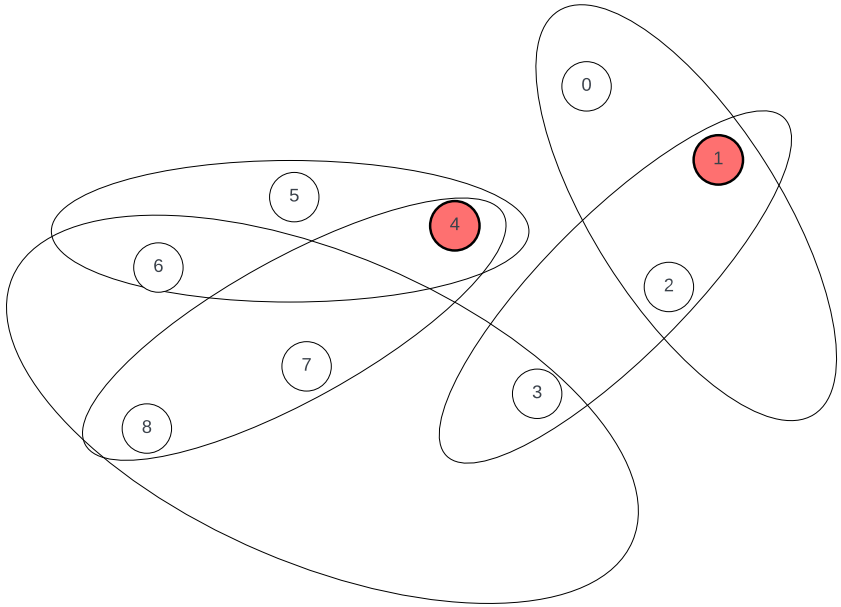
Ovviamente c'è una certa ridondanza qui.

Esempio III - output errato

Un **output errato** sarebbe il seguente:

```
2 1 4 #
```

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



Note su Input

Assunzioni generali (I)

- $1 < N \leq 15\,000$
- $1 < M \leq 50\,000$
- $1 \leq R_i \leq 3\,000$

Note su Input

Assunzioni generali (II)

- 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, $N \leq 24$;
- ★★☆ In 4 casi su 20, $R_i = 2$ per ogni riunione i;
- ★★★ Nel restante dei casi non ci sono particolari limitazioni.

Per la sufficienza, per esempio si possono risolvere in maniera esatta I 6 casi di difficoltà ★☆☆.

Limiti di tempo e memoria

I limiti di tempo e memoria sono:

- Tempo limite massimo: 3 secondi (timeout: 3.1 secondi).
- Memoria massima: 128 MB.

$\Rightarrow$ Limite di 40 sottoposizioni per gruppo.

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

$\Rightarrow$ **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.

Output

Potete stampare le soluzioni in maniera incrementale:

- Importate `conclave.h` (scaricabile da judge).
- 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 "conclave.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 `conclave.cpp`):

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

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

I file `conclave.cpp`, `grader.cpp` e `conclave.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

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;
- $\min S$: lower bound al numero di Votanti tali che ogni ad riunione partecipa almeno un votante. Nei casi in cui la soluzione ottima esatta (optS) è nota $\mathrm{minS} = \mathrm{optS}$.
- $\max S = \min(N, M)$: 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**.

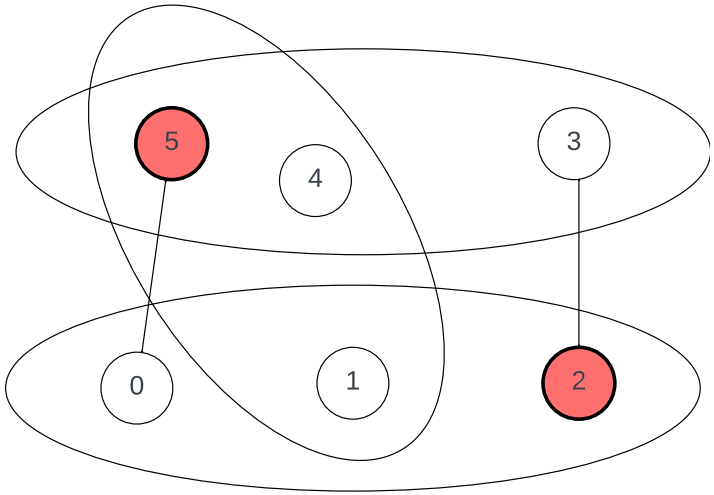
Upper bound

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 calcolo punteggio I

In questo esempio avevamo come output:

```
2 2 5#
```

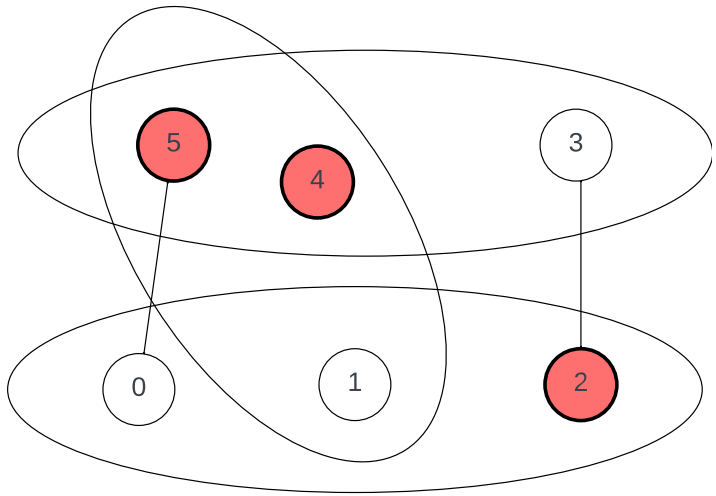


Che come abbiamo visto è valido e ottimo.
In questo caso $S = \min S = 1$, quindi $\mathcal{P} = 5 \times 1 = 5$

Esempio calcolo punteggio II

In questo output dallo stesso esempio precedente:

```
3 2 4 5 #
```

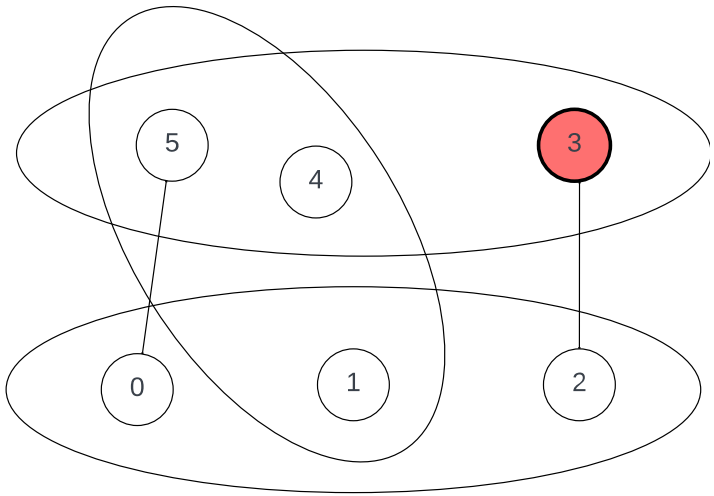


abbiamo una soluzione subottima, infatti ne esisterebbe una di cardinalità 2. Siamo nella situazione $\min S < S < \max S$ dove $\min S = 2$, $S = 3$ e $\max S = \min(6, 5)$, quindi $\mathcal{P} = 1 - \frac{3-2}{\min(6,5)-2} = 1 - \frac{1}{3} = 0.67$.

Esempio calcolo punteggio III

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 $\mathcal{P} = 0$.

Punteggio

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

✗ se una qualsiasi informazione data in output non è corretta (ad esempio il numero di Votanti selezionati non corrisponde con la dimensione della lista di identificativi data), si ottengono 0 punti.

✗ se una vostra soluzione dovesse mancare di coprire una riunione, verranno assegnati 0 punti per quel caso di test.

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: giovedì 05 giugno 2025 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 (giovedì 29 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

- giovedì 29 maggio 2025 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.