



Crittografia riflessa (crittografia)

Limite di tempo: 1.0 secondi

Limite di memoria: 256 MiB

Giorgio si è appassionato alla crittografia, e dopo aver studiato con attenzione tutti i più importanti algoritmi di crittografia simmetrica, ha pensato bene di introdurre un nuovo tipo di crittografia: la *crittografia riflessa*.

Più precisamente, per codificare i suoi messaggi Giorgio intende utilizzare una funzione che dato un numero N , ne calcola il numero speculare (con tutte le cifre invertite dall'ultima alla prima) e lo somma al numero di partenza. Aiuta Giorgio a scrivere la funzione di codifica!

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📁 Tra gli allegati a questo task troverai un template (`crittografia.c`, `crittografia.cpp`, `crittografia.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int codifica(int N);</code>
Pascal	<code>function codifica(N: longint): longint;</code>

In cui:

- L'intero N rappresenta il numero da codificare.
- La funzione dovrà restituire la somma tra il numero N e il suo riflesso, che verrà stampata sul file di output.

Dati di input

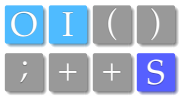
Il file `input.txt` è composto da un'unica riga contenente l'unico intero N .

Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

Assunzioni

- $1 \leq N \leq 1\,000\,000\,000$.



Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $N \leq 99$.
- **Subtask 3 [40 punti]:** Tutte le cifre di N sono 0 oppure 1.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

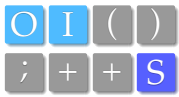
input.txt	output.txt
1023	4224

input.txt	output.txt
56390	65755

Spiegazione

Nel primo caso di esempio, $1023 + 3201$ fa 4224 .

Nel secondo caso di esempio, $56\,390 + 09\,365$ fa $65\,755$.



Allocazione di memoria (potenze)

Limite di tempo: 1.0 secondi
Limite di memoria: 256 MiB

Gabriele si è recentemente appassionato al linguaggio C++ (in particolare, C++11), in cui le uniche variabili ammesse sono array D -dimensionali con tutte le dimensioni uguali tra loro M (una variabile siffatta conterrà quindi M^D celle di memoria). Una delle novità introdotte dal C++11 rispetto al C++99, è che d'ora in poi sarà ammesso dichiarare un'unica variabile, che inoltre deve avere dimensione D almeno pari a 2.

Gabriele sta aggiornando il suo vecchio codice ai nuovi standard, e sta riscontrando qualche difficoltà con la gestione della memoria in alcuni dei suoi programmi più complessi. Sapendo che il suo computer ha una capacità di N celle di memoria, quante ne potrà al massimo sfruttare con un programma di C++11 (e quindi con un'unica variabile di dimensione una potenza M^D con esponente almeno 2)?

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📎 Tra gli allegati a questo task troverai un template (`potenze.c`, `potenze.cpp`, `potenze.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int alloca(int N);</code>
Pascal	<code>function alloca(N: longint): longint;</code>

In cui:

- L'intero N rappresenta il numero di celle di memoria presenti nel computer di Gabriele.
- La funzione dovrà restituire il massimo numero di celle allocabile in C++11 (e quindi la più grande potenza M^D minore o uguale a N con $D \geq 2$), che verrà stampato sul file di output.

Dati di input

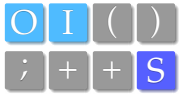
Il file `input.txt` è composto da un'unica riga contenente l'unico intero N .

Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

Assunzioni

- $1 \leq N \leq 100\,000$.



Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $N \leq 10$.
- **Subtask 3 [40 punti]:** $N \leq 1000$.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
10	9

input.txt	output.txt
32	32

Spiegazione

Nel **primo caso di esempio**, la potenza più grande minore o uguale a 10 è $9 = 3^2$.

Nel **secondo caso di esempio**, la potenza più grande minore o uguale a 32 è $32 = 2^5$.



Duplicato mancante (duplicato)

Limite di tempo: 1.0 secondi

Limite di memoria: 256 MiB

Gabriele stava fotocopiando un fascicolo di N pagine, quando arrivato a dover fotocopiare l'ultima pagina una folata di vento ha sollevato tutti i fogli, mescolandoli. Ora sul pavimento della camera di Gabriele sono sparpagliati $2N - 1$ fogli, e Gabriele non sa quale era la pagina che mancava ancora da fotocopiare.

Data la lista dei numeri di pagina dei fogli sparpagliati sul pavimento, aiuta Gabriele a capire qual è l'unico numero di pagina che è presente solo una volta.

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📎 Tra gli allegati a questo task troverai un template (`duplicato.c`, `duplicato.cpp`, `duplicato.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int trova(int N, int P[]);</code>
Pascal	<code>function trova(N: longint; var P: array of longint): longint;</code>

In cui:

- L'intero N rappresenta il numero di pagine del fascicolo
- L'array P , indicizzato da 0 a $2N - 2$, contiene i numeri di pagina dei fogli sparpagliati sul pavimento
- La funzione dovrà restituire il numero di pagina del foglio che Gabriele deve ancora fotocopiare, che verrà stampato sul file di output.

Dati di input

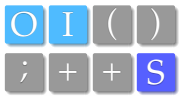
Il file `input.txt` è composto da due righe. La prima riga contiene l'unico intero N . La seconda riga contiene i $2N - 1$ interi P_i separati da uno spazio.

Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

Assunzioni

- $2 \leq N \leq 100\,000$.
- $1 \leq P_i \leq 1\,000\,000\,000$ per ogni $i = 0 \dots 2N - 2$.
- Ogni numero di pagina compare esattamente 2 volte, ad eccezione di un numero di pagina che appare esattamente 1 volta, e che coincide con la risposta al problema.



Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $N \leq 10$; inoltre, i numeri di pagina non superano il valore 100 000.
- **Subtask 3 [40 punti]:** $N \leq 1000$; inoltre, i numeri di pagina non superano il valore 100 000.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
2 1 1 2	2
input.txt	output.txt
5 4 5 4 10 8 10 3 5 8	3

Spiegazione

Nel **primo caso di esempio**, l'unico numero non ripetuto è 2.

Nel **secondo caso di esempio**, l'unico numero non ripetuto è 3.



Finanza creativa (bilancio)

Limite di tempo: 1.0 secondi
Limite di memoria: 256 MiB


La *SteamPower S.P.A.*, azienda leader mondiale nel campo delle macchine a vapore portatili, non accenna ad uscire dal periodo di crisi nonostante i forti e ben ponderati tagli al personale di recente effettuati. Per fortuna il *CEO* ha avuto una nuova geniale idea: affidarsi al massimo esperto mondiale in campo di finanza creativa. L'esperto ha già ricevuto il bilancio dal reparto contabilità, e la situazione è disastrosa: arrivati a questo punto l'unico modo che conosce per rassicurare gli azionisti e prendere tempo è quello di effettuare qualche piccolo ritocco ai libri contabili.

Più precisamente, vuole ricorrere al suo fedele bianchetto per correggere il totale U delle uscite. Inoltre, grazie agli insegnamenti di lunghi anni di esperienza, sa che per contenere i rischi dell'operazione non deve assolutamente eccedere le K cifre cancellate (sulle N complessive del totale U).

Aiuta l'esperto di finanza creativa a trovare il minimo intero ottenibile cancellando K cifre dall'intero U !

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

 Tra gli allegati a questo task troverai un template (`bilancio.c`, `bilancio.cpp`, `bilancio.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>void bianchetta(int N, int K, int U[], int C[]);</code>
Pascal	<code>procedure bianchetta(N, K: longint; var U, C: array of longint);</code>

In cui:

- L'intero N rappresenta il numero di cifre del totale delle uscite.
- L'intero K rappresenta il numero di cifre da cancellare.
- L'array U , indicizzato da 0 a $N - 1$, contiene l'elenco delle cifre del totale delle uscite, con la cifra più significativa in posizione 0 e così via fino alla cifra delle unità in posizione $N - 1$.
- La funzione dovrà riempire l'array C , indicizzato da 0 a $N - K - 1$, con l'elenco delle cifre rimaste dopo la cancellazione (dalla più alla meno significativa), che verrà stampato sul file di output.

Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene i due interi N e K . La seconda riga contiene le N cifre U_i del totale delle uscite U , separate da uno spazio.

Dati di output

Il file `output.txt` è composto da un'unica riga contenente $N - K$ cifre separate da uno spazio, la risposta a questo problema.

Assunzioni

- $1 \leq K < N \leq 1\,000\,000$.
- $0 \leq U_i \leq 9$ per ogni $i = 0 \dots N - 1$, e $U_0 \geq 1$.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $N \leq 10$.
- **Subtask 3 [30 punti]:** $N \leq 200$.
- **Subtask 3 [20 punti]:** $N \leq 10\,000$.
- **Subtask 4 [20 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
5 2 1 9 5 0 3	1 0 3
input.txt	output.txt
9 3 9 4 1 5 7 1 1 2 3	1 5 1 1 2 3
input.txt	output.txt
10 5 1 3 2 0 2 1 8 5 3 6	0 1 5 3 6

Spiegazione

Nel **primo caso di esempio** conviene cancellare le due cifre più alte (5 e 9).

Nel **secondo caso di esempio** conviene cancellare le due cifre più significative (9 e 4), e la restante cifra più alta (il 7).

Nel **terzo caso di esempio** conviene cancellare le tre cifre più significative (1, 3 e 2), il 2 ancora successivo e la restante cifra più alta (l'8).

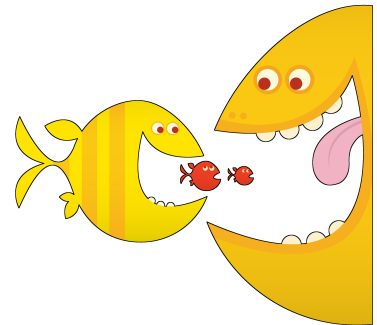
Pesci Mangioni (pesci)

Limite di tempo: 1.0 secondi
 Limite di memoria: 256 MiB

Al corso di *Ittiologia Computazionale* Gabriele sta studiando le dinamiche algoritmiche del mondo dei pesci. Un argomento particolarmente interessante è quello dell'alimentazione dei *Pesci Mangioni*.

Questi voracissimi pesci, di dimensioni molto variegata, sono molto aggressivi e ogni volta che si sentono osservati da un'altro pesce reagiscono mangiandoselo (se possono), senza mai sentire il senso di sazietà. Per compito Gabriele ha stilato un modello semplificato del loro comportamento, che dovrebbe essere valido almeno nel caso in cui i pesci si trovino in un tubo da sperimentazione. Secondo il modello:

- ogni pesce può nuotare da sinistra verso destra o da destra verso sinistra, e stando nel tubo questa direzione non cambierà mai,
- ogni pesce ha una grandezza ben definita, e tutte le grandezze sono distinte,
- un incrocio di pesci è una coppia di pesci adiacenti, per cui il più a sinistra nuota verso destra, e il più a destra nuota verso sinistra,
- durante un incrocio di pesci il pesce più grande mangia il pesce più piccolo e continua nel suo moto, eventualmente incrociando altri pesci.



Per verificare quanto è buono il modello che ha derivato di questa specie curiosa, Gabriele intende ora liberare in un tubo N Pesci Mangioni e monitorare cosa succede. Quanti pesci dovrebbero rimanere vivi dopo che tutti gli incroci si saranno risolti?

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

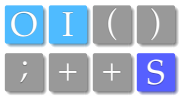
📖 Tra gli allegati a questo task troverai un template (`pesci.c`, `pesci.cpp`, `pesci.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int mangia(int N, int direzione[], int dimensione[]);</code>
Pascal	<code>function mangia(N: longint; var direzione, dimensione: array of longint): longint;</code>

In cui:

- L'intero N rappresenta il numero di pesci liberati da Gabriele.
- L'array `direzione`, indicizzato da 0 a $N - 1$, indica se l' i -esimo pesce nuota da sinistra verso destra (`direzione[i] = 0`) o da destra verso sinistra (`direzione[i] = 1`).
- L'array `dimensione`, indicizzato da 0 a $N - 1$, indica la dimensione dell' i -esimo pesce.
- La funzione dovrà restituire il numero di Pesci Mangioni sopravvissuti dopo che tutti gli incroci sono stati risolti, che verrà stampato sul file di output.



Dati di input

Il file `input.txt` è composto da $N + 1$ righe. La prima riga contiene l'unico intero N . Le successive N righe contengono ciascuna i due interi `direzione[i]`, `dimensione[i]`.

Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

Assunzioni

- $1 \leq N \leq 100\,000$.
- `direzione[i]` $\in \{0, 1\}$ per ogni $i = 0 \dots N - 1$.
- $1 \leq \text{dimensione}[i] \leq 10^9$ per ogni $i = 0 \dots N - 1$; inoltre, le dimensioni dei pesci sono tutte distinte.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]**: Casi d'esempio.
- **Subtask 2 [20 punti]**: $N \leq 10$.
- **Subtask 3 [40 punti]**: $N \leq 100$.
- **Subtask 4 [30 punti]**: Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
2 0 6 1 9	1

input.txt	output.txt
2 1 6 1 9	2

input.txt	output.txt
5 0 12 0 9 1 3 0 4 1 6	2



Spiegazione

Nel **primo caso di esempio** il secondo pesce, di dimensione 9, mangia il primo, di dimensione 6.

Nel **secondo caso di esempio** i pesci non si incrociano.

Nel **terzo caso di esempio** il pesce di dimensione 4 viene mangiato dal pesce di dimensione 6, ma il pesce di dimensione 9 mangia sia il pesce di dimensione 3 che quello di dimensione 6.

Ponti e isole (ponti)

Limite di tempo: 1.0 secondi

Limite di memoria: 256 MiB

A seguito di un violento maremoto alcuni dei ponti che collegano le N isole dell'arcipelago Nowhere sono stati distrutti e il governo deve correre ai ripari per non lasciare che alcune isolette rimangano isolate e irraggiungibili.



Ponte dell'isola Kouri, in Giappone. Immagine originale: <http://www.panoramio.com/photo/95167664>.

Il governo dell'arcipelago Nowhere ha quindi assunto Giorgio per determinare quale è il minimo numero di ponti che è necessario costruire in aggiunta agli M rimasti affinché l'arcipelago sia di nuovo connesso, ovvero sia possibile da ogni isola raggiungere tutte le altre isole. Aiuta Giorgio a svolgere il suo compito!

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📎 Tra gli allegati a questo task troverai un template (`ponti.c`, `ponti.cpp`, `ponti.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

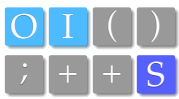
C/C++	<code>int costruisci(int N, int M, int da[], int a[]);</code>
Pascal	<code>function costruisci(N, M: longint; var da, a: array of longint): longint;</code>

In cui:

- L'intero N rappresenta il numero di isole che formano l'arcipelago.
- L'intero M rappresenta il numero di ponti rimasti intatti dopo il maremoto.
- I due array `da` e `a`, indicizzati da 0 a $M - 1$, contenenti all'indice i le due isole collegate dal ponte i .

Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene i due interi N e M . Le successive M righe contengono due interi ciascuna, gli indici `da[i]`, `a[i]` delle isole collegate dall' i -esimo ponte.



Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

Assunzioni

- $1 \leq N \leq 10\,000$.
- $0 \leq M \leq 100\,000$.
- $0 \leq da[i], a[i] < N$ per ogni $i = 0 \dots M - 1$.
- Per ogni coppia di isole esiste al più un ponte che le collega, e i ponti non vengono ripetuti nell'input.
- Nessun ponte collega un'isola a se stessa.
- Le isole sono numerate a partire da 0.
- Se l'arcipelago è già connesso, rispondere il valore 0.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $N \leq 10$.
- **Subtask 3 [40 punti]:** $N \leq 100$.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

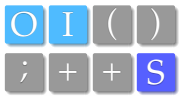
input.txt	output.txt
2 0	1

input.txt	output.txt
4 2 1 3 3 2	1

Spiegazione

Nel **primo caso di esempio** è sufficiente costruire un ponte tra le isole 0 e 1.

Nel **secondo caso di esempio** è sufficiente costruire un ponte tra le isole 0 e 2.



Oro, argento e bronzo (medaglie)

Limite di tempo: 1.0 secondi
Limite di memoria: 256 MiB

In uno spiacevole incidente, Giorgio e Gabriele hanno mescolato tutte le loro innumerevoli medaglie vinte in varie competizioni, e non riescono più a ricostruire quali erano le loro in partenza. Decidono quindi di spartirselo secondo un gioco, detto *oro, argento e bronzo*.

Impilano quindi tutte le N medaglie, ciascuna delle quali può essere d'oro, d'argento o di bronzo. Giorgio, per maggiore anzianità, inizia il gioco e prende una, due o tre medaglie (a sua scelta) dall'inizio della pila. A seguire, Gabriele fa lo stesso e i turni si susseguono alternati finché tutte le medaglie sono state prese da uno dei due.

Giorgio, di recente in difficoltà economiche, ha il segreto piano di rivendere tutte le medaglie che riesce a raccogliere; e sa che potrà ottenere 500 euro per ogni medaglia d'oro, 300 euro per ogni medaglia d'argento e 100 euro per ogni medaglia di bronzo. Sapendo che sia Giorgio che Gabriele giocano nel modo ottimale¹, quanti euro al massimo potrà ottenere Giorgio?

Implementazione

Dovrai sottoporre esattamente un file con estensione `.c`, `.cpp` o `.pas`.

📁 Tra gli allegati a questo task troverai un template (`medaglie.c`, `medaglie.cpp`, `medaglie.pas`) con un esempio di implementazione da completare.

Se sceglierai di utilizzare il template, dovrai implementare la seguente funzione:

C/C++	<code>int gioca(int N, short M[]);</code>
Pascal	<code>function gioca(N: longint; var M: array of integer): longint;</code>

In cui:

- L'intero N rappresenta il numero di medaglie nella pila all'inizio.
- L'array M , indicizzato da 0 a $N - 1$, contiene la descrizione della pila di medaglie, in cui all'indice 0 è indicata la medaglia più alla base, mentre all'indice $N - 1$ è indicata la medaglia più in cima. Precisamente, $M[i]$ vale 0 per indicare una medaglia di bronzo, 1 per indicare una medaglia d'argento e 2 per indicare una medaglia d'oro.
- La funzione dovrà restituire il massimo numero di euro che Giorgio può ottenere dalla vendita delle medaglie guadagnate, che verrà stampato sul file di output. Si consideri che sia Giorgio che Gabriele giocano in maniera ottimale per ottenere a fine partita il massimo valore in euro possibile.

Dati di input

Il file `input.txt` è composto da due righe. La prima riga contiene l'unico intero N . La seconda riga contiene gli N interi M_i separati da uno spazio.

¹In un gioco a informazione perfetta (tutti i giocatori conoscono il complessivo stato del gioco) e deterministico (non ci sono elementi casuali), una strategia S è *ottimale* se in ogni configurazione del gioco ti fa scegliere la mossa con il *risultato garantito* più alto. Il risultato garantito di una mossa è il minimo risultato che si può ottenere a fine partita seguendo la strategia S , al variare della strategia T dell'avversario.



Dati di output

Il file `output.txt` è composto da un'unica riga contenente un unico intero, la risposta a questo problema.

Assunzioni

- $1 \leq N \leq 100\,000$.
- $0 \leq M_i \leq 2$ per ogni $i = 0 \dots N - 1$.

Assegnazione del punteggio

Il tuo programma verrà testato su diversi test case raggruppati in subtask. Per ottenere il punteggio relativo ad un subtask, è necessario risolvere correttamente tutti i test relativi ad esso.

- **Subtask 1 [10 punti]:** Casi d'esempio.
- **Subtask 2 [20 punti]:** $N \leq 10$.
- **Subtask 3 [40 punti]:** $N \leq 100$.
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

Esempi di input/output

input.txt	output.txt
5 2 1 0 1 0	600
input.txt	output.txt
10 1 0 0 0 2 1 0 1 1 2	1600
input.txt	output.txt
10 1 0 0 0 2 1 0 1 0 2	1500

Spiegazione

Nel **primo caso di esempio**, a Giorgio conviene prendere solo la prima medaglia (del valore di 100 euro). A questo punto Gabriele non ha scelta che prendere le tre medaglie seguenti (del valore complessivo di 700 euro), perché qualunque mossa egli faccia sa che Giorgio prenderebbe tutto quello che resta.

Nel **secondo caso di esempio**, a Giorgio conviene prendere le prime due medaglie (del valore di 800 euro), a Gabriele dunque conviene prenderne tre (del valore di 700 euro), Giorgio quindi ne prende una (del valore di 500 euro), Gabriele tre (del valore di 300 euro) e Giorgio infine ancora una (del valore di 300 euro), per un totale di 1600 euro per Giorgio.

Nel **terzo caso di esempio**, a Giorgio conviene prendere la prima medaglia (del valore di 500 euro), a Gabriele dunque conviene prenderne una (del valore di 100 euro), e da qui la partita procede analogamente al precedente esempio ma a parti invertite, e quindi Giorgio raccoglie altri 1000 euro per un totale di 1500 euro per Giorgio.