

## Calcolatrice d'epoca (calcolatrice)

La calcolatrice che Giorgio conserva gelosamente da quando faceva le elementari si è rotta, e ora il suo schermo *LCD* vecchio stile penzola appeso ai tasti dal solo filo di alimentazione. Nonostante questo, funziona ancora perfettamente come un tempo e quindi Giorgio non ha intenzione di smettere di utilizzarla.

Con la sua affezionata calcolatrice, Giorgio ha appena fatto dei lunghi e complessi calcoli i cui risultati potrebbero portare alla soluzione di numerose congetture in innumerevoli campi della matematica, e ne ha trascritto i risultati su un foglio. Tuttavia solo dopo si è accorto che, per via del danno descritto sopra, non è in grado di capire l'orientamento giusto dello schermo e quindi in alcuni casi potrebbe aver trascritto il risultato sbagliato (e cioè come si leggerebbe ruotando lo schermo della calcolatrice di  $180^\circ$ ).

Da un'approssimazione a stima che si è fatto, gli sembra che i risultati dovrebbero essere numeri abbastanza piccoli. Pertanto, dato un numero  $N$ , vuole scrivere un programma che calcoli il corrispondente numero  $M$  ruotato di  $180^\circ$ , e se questo è effettivamente un numero sensato scelga il minore tra i due.



## Implementazione

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

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

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

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

In cui:

- L'intero  $N$  rappresenta il numero segnato sul foglio da Giorgio.
- La funzione dovrà restituire il minimo tra  $N$  e il corrispondente  $M$  ruotato di  $180^\circ$  (se sensato), 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

- $0 \leq N \leq 1\,000\,000\,000$ .

## 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]:**  $N \leq 10$ .
- **Subtask 3 [30 punti]:**  $N \leq 100$ .
- **Subtask 4 [40 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
14	14

input.txt	output.txt
12650	12650

input.txt	output.txt
806129	621908

## Spiegazione

Nel **primo caso di esempio**, il numero 14 ruotato non genera un numero sensato, per via della presenza della cifra 4.

Nel **secondo caso di esempio**, il numero 12650 ruotato genera il numero 05921 che non è da considerarsi sensato, per via della presenza della cifra 0 come prima cifra.

Nel **terzo caso di esempio**, il numero 806129 ruotato genera il numero 621908 che è sensato e minore del precedente.

## Assenza di gravità (gravity)

Gabriele è in visita alla stazione spaziale internazionale (*ISS*), incaricato di aggiornare i sistemi di bordo ai nuovi standard del C—11. Purtroppo l'operazione è andata storta e la stazione è entrata in stato di freeze. Ora bisogna urgentemente riavviare tutti i sistemi di energy e life support prima che gli occupanti rischino la vita!

Gabriele, imbracciata la tuta spaziale, esce sulla parete rettangolare esterna dell'astronave in cui si trovano i grossi interruttori di reset di emergenza di tutti i numerosi sistemi che compongono l'astronave. Gli  $N$  pesanti interruttori si trovano sulla parete uno di seguito all'altro, ma ciascuno ad una altezza differente  $H_i$ . Ogni volta che spingerà un'interruttore per attivare il corrispondente sistema, Gabriele sa che la sua direzione di moto lungo l'asse verticale si invertirà (sbalzato dal principio di azione e reazione), mentre la sua direzione di moto lungo l'asse orizzontale rimarrà invariata. Il problema ora è quindi quello di pianificare una strategia per riavviare più sistemi possibili in una sola passata, guadagnandosi qualche minuto di tempo per poter stabilizzare la situazione!

Aiuta Gabriele a spingere il maggior numero di interruttori. Una sottosequenza di interruttori è ammissibile se è *alternante*, e cioè non vi sono tre interruttori di seguito (nella sequenza) con altezze tutte crescenti o tutte decrescenti. Trova quindi la più lunga sottosequenza ammissibile!

## Implementazione

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

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

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

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

In cui:

- L'intero  $N$  rappresenta il numero di interruttori presenti sulla parete.
- L'array  $H$ , indicizzato da 0 a  $N - 1$ , contiene le altezze  $H_i$  a cui gli interruttori sono situati.
- La funzione dovrà restituire la massima lunghezza per una sottosequenza alternante, che verrà stampata 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 gli  $N$  interi  $H_i$  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



- $1 \leq N \leq 10\,000$ .
- $1 \leq H_i \leq 20\,000$  per ogni  $i = 0 \dots N - 1$ .
- Le altezze  $H_i$  sono tutte differenti.

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

## Spiegazione

Nel **primo caso di esempio**, è possibile premere il primo e il secondo interruttore, ma non è possibile premerne tre.

Nel **secondo caso di esempio**, è possibile premere gli interruttori 5-7-6-8-3.

## Piroette (piroette)

Uno dei tanti passatempi di Giorgio è quello del coreografo. Di recente si è occupato di curare la coreografia della scena più importante di un balletto, in cui i ballerini eseguono un certo numero  $P$  di piroette, ovvero di mezzigiri su una gamba. Giorgio nel copione aveva specificato la quantità di gradi che i ballerini dovevano compiere nella rotazione, cioè la quantità  $G = 180 \cdot P$ .

A causa di un guasto alla stampante, ogni volta che i copioni vengono stampati le cifre di  $G$  vengono permutate casualmente. Per questo, ogni ballerino ha ricevuto un copione che specifica una quantità di gradi diversi. Lo spettacolo sta per debuttare, e restano pochi minuti per trovare una soluzione. Per mettere tutti d'accordo, Giorgio stabilisce che il numero di gradi da compiere è pari al massimo numero divisibile per 180 che si ottiene ripermutando le cifre del numero scritto sui copioni.

In altre, parole, preso un numero  $G$ , è necessario trovare il più grande numero  $\tilde{G}$  divisibile per 180 ottenibile permutando le cifre di  $G$ . Il tempo stringe, aiuta Giorgio a risolvere la situazione!

## Implementazione

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

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

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

C/C++	<code>void permuta(int N, int G[], int Gtilde[]);</code>
Pascal	<code>procedure permuta(N: longint; var G, Gtilde: array of longint);</code>

In cui:

- L'intero  $N$  rappresenta il numero di cifre di  $G$ .
- L'array  $G$ , indicizzato da 0 a  $N - 1$ , contiene le cifre di  $G$ . La cifra più significativa è contenuta nella posizione 0 (*ad esempio, se  $G = 630$ , si ha  $N = 3$  e  $G[0] = 6$ ,  $G[1] = 3$ ,  $G[2] = 0$* ).
- La funzione dovrà scrivere in  $Gtilde$  la permutazione di  $G$  che garantisce il massimo intero divisibile per 180, secondo la convezione che la cifra più significativa si trova nella posizione 0.

## 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  $G[i]$  separati da uno spazio, secondo la convezione che la cifra più significativa si trova nella posizione 0.

## Dati di output

Il file `output.txt` è composto da un'unica riga contenente  $N$  interi, le cifre di  $Gtilde$ , secondo la convezione che la cifra più significativa si trova nella posizione 0.

## Assunzioni

- $3 \leq N \leq 100\,000$ .
- $0 \leq G[i] \leq 9$  per ogni  $i = 0 \dots N - 1$ .
- La prima cifra di  $G$  non è 0.
- Esiste almeno una permutazione delle cifre di  $G$  che conduce ad un numero divisibile per 180.

## 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 [30 punti]:**  $N \leq 8$ .
- **Subtask 3 [40 punti]:**  $N \leq 1000$ .
- **Subtask 4 [20 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
3 6 3 0	3 6 0
input.txt	output.txt
4 4 2 0 3	4 3 2 0

## Spiegazione

Nel **primo caso di esempio** l'unica permutazione delle cifre che conduce ad un numero divisibile per 180 è 360.

Nel **secondo caso di esempio** esistono 4 permutazioni delle cifre che portano ad un numero divisibile per 180:

- 2340,
- 3240,
- 3420,
- 4320.

Il massimo tra questi numeri è 4320.

## PaH TuNZ (remix)

Gabriele ha deciso di buttarsi nel mondo della musica trash. Per questo, ha scritto un software in grado di prendere un normale testo e remixarlo. Il processo è semplice, dal momento che tutto quello che fa il software è inserire nel testo un certo numero di effetti musicali, identificati dalle stringhe ‘PaH’ e ‘TuNZ’. Ad esempio, un testo formato dalle parole ‘Sette Otto’ può essere trasformato nel testo ‘TuNZTuNZSettePaHPaHTuNZOtto’. In generale, gli effetti sonori *possono* essere inseriti prima dell’inizio del testo e dopo la fine, ma non sono obbligatori, mentre tra due parole consecutive è sempre inserito almeno un effetto sonoro.

Giorgio non è un grande fan di questo genere musicale, e non riesce a distinguere bene le parole che formano il testo originale. Per questo è interessato a ripulire la musica dagli effetti sonori e ricostruire il testo originale. Aiutalo!

## Implementazione

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

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

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

C/C++	<code>void ripulisci(int N, char remix[], char testo[]);</code>
Pascal	<code>procedure ripulisci(N: longint; var remix, testo: array of char);</code>

In cui:

- L’intero  $N$  rappresenta il numero di caratteri del remix del testo.
- La stringa `remix`, indicizzata da 0 a  $N - 1$ , contiene il testo remixato.
- La funzione dovrà scrivere nella stringa `testo` il testo originale, 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 la stringa `remix`.

## Dati di output

Il file `output.txt` è composto da un’unica riga contenente una stringa, la risposta a questo problema.



## Assunzioni

- $1 \leq N \leq 100\,000$ .
- Il testo originale non contiene le parole ‘PaH’ e ‘TuNZ’.
- Il testo contiene almeno una parola.

## 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]:**  $N \leq 100$ .
- **Subtask 3 [40 punti]:**  $N \leq 1000$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
27 TuNZTuNZSettePaHPaHTuNZOtto	Sette Otto

input.txt	output.txt
12 PaHXXPaHTuNZ	XX



## Tris in solitaria (solitario)

Gabriele, ancora ossessionato dal gioco del tris, ha deciso che non vuole più ricorrere a un programma per vincere contro di Giorgio e ha quindi iniziato un duro programma di allenamento in solitaria. Per allenarsi, Gabriele prende una griglia da tris composta da  $N \times M$  caselle, e va avanti a segnare X su alcune caselle finché riesce a non creare tris.

Questo programma di allenamento, tuttavia, lo sta lasciando insoddisfatto in quanto non riesce a capire se sta andando bene oppure no. Aiuta Gabriele a valutarsi calcolando quante X può segnare al massimo in una griglia  $N \times M$  senza creare tris!

## Implementazione

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

📎 Tra gli allegati a questo task troverai un template (`solitario.c`, `solitario.cpp`, `solitario.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, int M);</code>
Pascal	<code>function gioca(N, M: longint): longint;</code>

In cui:

- Gli interi  $N$ ,  $M$  rappresentano le dimensioni della griglia.
- La funzione dovrà restituire il numero massimo di X che si possono segnare senza creare tris, che verrà stampato sul file di output.

## Dati di input

Il file `input.txt` è composto da un'unica riga contenente i due interi  $N$  ed  $M$ .

## Dati di output

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

## Assunzioni

- $1 \leq N, M \leq 10$ .
- $N \times M \leq 36$ .

## 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]:**  $N \leq 2$ .
- **Subtask 3 [30 punti]:**  $N \times M \leq 16$ .
- **Subtask 4 [30 punti]:**  $N \times M \leq 25$ .
- **Subtask 5 [10 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
2 6	8

input.txt	output.txt
3 3	6

## Spiegazione

Nel **primo caso di esempio**, si può riempire tutta la griglia tranne la seconda e la penultima colonna.

Nel **secondo caso di esempio**, la configurazione massima è:

×	×	
×		×
	×	×

## Fette di strudel (strudel)

Il dolce preferito di Giorgio è lo strudel, e ne è talmente goloso che sarebbe capace di mangiarne una quantità potenzialmente infinita. Gabriele conosce bene l'amore di Giorgio per lo strudel e per questo, nel famoso viaggio in taxi verso Pinerolo, ha portato con sé uno strudel gigantesco, composto di  $N$  sezioni di differente composizione.

Purtroppo Gabriele non sapeva che Giorgio odia le mandorle, e credendo di fare cosa gradita le ha inserite nell'impasto. Per fortuna non tutto è perduto: Giorgio ama così tanto lo strudel che è disposto a mangiarne una fetta a patto che la quantità di cannella sia superiore alla quantità di mandorle in quella fetta, di modo da celarne l'odiato sapore.

Per questo, dopo una meticolosa ispezione, gli amici hanno determinato quante mandorle `mandorle[i]` e quanta cannella `cannella[i]` sono presenti in ognuna delle  $N$  sezioni di strudel. Sta ora a Giorgio tagliare la sua fetta di strudel, cioè una parte composta da un numero intero di sezioni contigue. Quanto può essere lunga al massimo questa fetta, sapendo che la quantità totale di cannella presente nella fetta scelta non deve essere inferiore alla quantità totale di mandorle?

## Implementazione

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

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

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

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

In cui:

- L'intero  $N$  rappresenta il numero di sezioni in cui è diviso lo strudel.
- L'array `mandorle`, indicizzato da 0 a  $N - 1$ , contiene la quantità di mandorle (in milligrammi) presenti in ogni sezione di strudel.
- L'array `cannella`, indicizzato da 0 a  $N - 1$ , contiene la quantità di cannella (in milligrammi) presente in ogni sezione di strudel.
- La funzione dovrà restituire la massima lunghezza della fetta che Giorgio può mangiare, che verrà stampata sul file di output.

## Dati di input

Il file `input.txt` è composto da tre righe. La prima riga contiene l'unico intero  $N$ . La seconda riga contiene gli  $N$  interi `mandorle[i]` separati da uno spazio. La terza riga contiene gli  $N$  interi `cannella[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

- $1 \leq N \leq 100\,000$ .
- $0 \leq \text{mandorle}[i], \text{cannella}[i] \leq 10\,000$  per ogni  $i = 0 \dots N - 1$ .
- Esiste sempre una fetta mangiabile da Giorgio.

## 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]:**  $N \leq 100$ .
- **Subtask 3 [40 punti]:**  $N \leq 1000$ .
- **Subtask 4 [30 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

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

## Spiegazione

Nel **primo caso di esempio** Giorgio può prendere le prime due sezioni (4mg di mandorle e 5mg di cannella) o le ultime due sezioni (7mg di mandorle e 7mg di cannella), ma non tutte e tre (9mg di mandorle e 8mg di cannella).

Nel **secondo caso di esempio** Giorgio può prendere le quattro sezioni centrali (11mg di mandorle e 15mg di cannella).

## Limiti di velocità (velox)

Gabriele vuole mettere alla prova la sua nuova utilitaria sportiva, e sta quindi cercando il tragitto più lungo possibile da fare in un'unica graduale accelerata. Il compito è reso arduo dalla presenza di un complicato sistema di limiti di velocità e sensi unici, che Gabriele non può assolutamente permettersi di violare.

La zona di Brescia è composta di  $N$  incroci tra cui corrono  $M$  tratti di strada di pari lunghezza, ciascuno con il suo limite di velocità  $V_i$ . Trova il tragitto più lungo composto da tratti di strada con limiti di velocità strettamente crescenti!

## Implementazione

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

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

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

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

In cui:

- L'intero  $N$  rappresenta il numero di incroci, indicizzati da 0 a  $N - 1$ .
- L'intero  $M$  rappresenta il numero di tratti di strada, indicizzati da 0 a  $M - 1$ .
- Gli array `da`, `a`, `V`, indicizzati da 0 a  $M - 1$ , contengono la descrizione dei tratti di strada. Più precisamente, per ogni  $i$  vi è un tratto di strada a senso unico dall'incrocio `da[i]` all'incrocio `a[i]` con limite di velocità `V[i]`.
- La funzione dovrà restituire la massima lunghezza per un percorso lungo il quale i limiti di velocità sono strettamente crescenti, che verrà stampata sul file di output.

## Dati di input

Il file `input.txt` è composto da  $M + 1$  righe. La prima riga contiene i due interi  $M$ ,  $N$ . Le successive  $M$  righe contengono ciascuna i tre interi `da[i]`, `a[i]`, `V[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

- $2 \leq N \leq 10\,000$ .
- $1 \leq M \leq 100\,000$ .
- $1 \leq V_i \leq 1\,000\,000$  per ogni  $i = 0 \dots N - 1$ .
- I tratti di strada sono tutti validi e non ripetuti (non hanno lo stesso **da** e lo stesso **a**).
- Le strade sono da considerarsi a senso unico, a meno che vengano indicate nell'input "due volte" (una per senso di marcia, con **da** e **a** invertiti). I limiti di velocità nei due sensi possono differire.

## 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 [40 punti]:**  $N \leq 10$ .
- **Subtask 3 [30 punti]:**  $N \leq 100$ .
- **Subtask 4 [20 punti]:** Nessuna limitazione specifica.

## Esempi di input/output

input.txt	output.txt
3 5 0 1 90 0 2 90 1 0 90 1 2 90 2 0 90	1
input.txt	output.txt
5 6 0 1 60 1 2 110 2 1 90 2 3 130 3 0 70 3 4 50	3

## Spiegazione

Nel **primo caso di esempio**, è possibile fare un solo tratto.

Nel **secondo caso di esempio** è possibile fare tre tratti in vari modi, 0–1–2–3 oppure 2–1–2–3, ma non è possibile farne quattro.

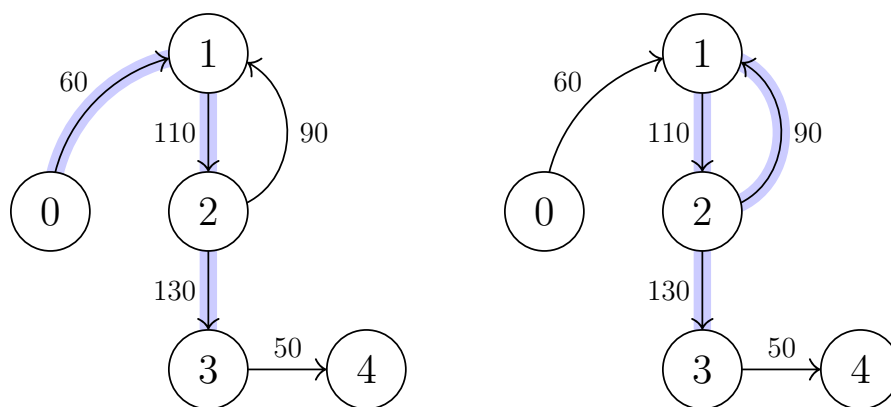


Figura 1: Percorsi ottimali per il secondo caso di esempio.