Building Bridges

Time Limit: 3 s *Memory Limit: 128 MB*

In un fiume molto largo si trovano n pilastri di possibilmente differenti altezze che spuntano fuori dall'acqua e che sono dislocati in una linea retta da una sponda all'altra del fiume. Noi vorremmo costruire un ponte che utilizza i pilastri come supporto. Per raggiungere questo obiettivo vogliamo selezionare un sottoinsieme di pilastri e connettere le loro estremità superiori in modo da formare un ponte. Questo sottoinsieme deve includere il primo e l'ultimo pilastro.

Il costo di costruire la sezione del ponte che si trova tra i pilastri i e j è ottenuto dalla formula $(h_i - h_j)^2$, dove h_i rappresenta l'altezza dell'i-esimo pilastro, poichè vogliamo ottenere il ponte più "orizzontale" possibile. Inoltre, è anche necessario rimuovere tutti i pilastri che non sono parte del ponte, poichè ostruiscono il flusso dell'acqua del fiume. Il costo per rimuovere l'i-esimo pilastro è w_i. Nota che questo costo può anche assumere valori negativi (infatti ci sono alcune organizzazioni che sono disposte a pagarti per rimuovere alcuni pilastri). Tutte le altezze h_i e i costi w_i sono interi.

Qual è il minimo costo di costruzione di un ponte che colleghi le due sponde del fiume (ovvero che colleghi i pilastri situati agli estremi)?

Input

La prima riga contiene il numero di pilastri n. La seconda riga contiene le altezze dei pilastri h_i in ordine, separate da uno spazio. La terza riga contiene i w_i (ossia i costi della rimozione dei pilastri i) nel solito ordine.

Output

Stampa il minimo costo di costruzione di un tale ponte e ricorda che il risultato può anche essere un numero negativo.

Limiti

- $2 \leq n \leq 10^5$
- $0 \leq h_i \leq 10^6$
- $0 \leq |w_i| \leq 10^6$

Subtask 1 (30 punti)

- $n \leq 1\,000$

Subtask 2 (30 punti)

- la soluzione ottimale include al massimo 2 pilastri addizionali (oltre al primo e all'ultimo)
- $|w_i| \leq 20$

Subtask 3 (40 punti)

- nessun limite addizionale

Esempio

Input	**Output**
6 3 8 7 1 6 6 0 -1 9 1 2 0	17

Palindromic Partitions

Time Limit: 10 s *Memory Limit: 128 MB*

Una *partizione* di una stringa s è un insieme di una o più sottostringhe di s non vuote e che non si sovrappongono (chiamiamole $a_1, a_2, a_3, \ldots, a_d$), in modo tale che s è ottenuta concatenandole ($s = a_1 + a_2 + a_3 + \ldots + a_d$). Denominiamo queste sottostringhe "*frammenti*" e definiamo la *lunghezza* di una partizione come il numero di frammenti d.

Possiamo rappresentare la partizione di una stringa scrivendo ogni frammento fra parentesi. Ad esempio, la stringa "`decode`" può essere partizionata come `(d)(ec)(ode)` o `(d)(e)(c)(od)(e)` oppure `(decod)(e)` o anche `(decode)` o `(de)(code)` oppure un po' di altri modi.

Una partizione è *palindroma* se i suoi frammenti formano una stringa plindroma, considerando ogni frammento come un'unità atomica. Ad esempio, le uniche partizioni palindrome di "`decode`" sono `(de)(co)(de)` e `(decode)`. Questo mostra inoltre che ogni parola ha una partizione palindroma banale, che corrisponde alla partizione di lunghezza 1.

Il tuo compito è quello di calcolare il massimo possibile numero di frammenti di una partizione palindroma.

Input

L'input inizia con il numero di test case t sulla prima riga. Le successive t righe descrivono i test case individuali che consistono in una sola parola s, contentente solamente lettere minuscole dell'alfabeto inglese. Non ci sono spazi nell'input.

Output

Per ogni testcase, stampa un singolo numero: il massimo numero di frammenti di una partizione palindroma della parola ricevuta in input s. Le risposte di ciascun testcase devono trovarsi ognuna sulla sua riga.

Limiti

Denotiamo la lunghezza della stringa in input s con n.

- $1 \leq t \leq 10$
- $1 \leq n \leq 10^6$

Subtask 1 (15 punti)

- $n \leq 30$

Subtask 2 (20 punti)

- $n \leq 300$

Subtask 3 (25 punti)

- $n \leq 10\,000$

Subtask 4 (40 punti)

- nessun limite addizionale

Esempio

Input	Output

<pre>4
bonobo
deleted
racecar
racecars</pre>
	<pre>3
	5
	7
	1</pre>

Chase

Time Limit: 4 s *Memory Limit: 512 MB*

Il gatto Tom sta nuovamente inseguendo il topo Jerry! Jerry cerca di guadagnare terreno correndo tra gruppi di piccioni, dove per Tom è più difficile seguirlo. Fortunatamente, Jerry è appena arrivato nel parco centrale di Lubiana. Il parco ha n statue, numerate $1 \ldots n$, e $n-1$ percorsi tra di esse che non si intersecano e che le collegano in modo tale che è sempre possibile raggiungere ogni statua da ogni altra statua seguendo alcuni percorsi. Vicino alla statua i ci sono p_i piccioni. Jerry ha v briciole di pane nelle tasche. Se fa cadere una briciola di pane vicino alla statua in cui si trova, i piccioni di tutte le statue vicine voleranno immediatamente a questa statua per mangiare la briciola. Di conseguenza il numero di piccioni p attorno a questa statua e alle statue vicine cambia.

Questo avviene nel seguente ordine: prima Jerry arriva alla statua i e incontra p_i piccioni. Dopodichè, lascia cadere la briciola e lascia la statua. I piccioni da statue vicine si spostano alla statua i prima che Jerry arrivi alla statua successiva (in modo che non contino nel totale dei piccioni che Jerry ha incontrato).

Jerry può entrare nel parco a una statua qualsiasi, percorrere qualche percorso (ma non può utilizzare lo stesso percorso due volte) e poi uscire dal parco dalla statua che preferisce. Dopo che Jerry esce dal parco, Tom entrerà e attraverserà esattamente la stessa sequenza di percorsi. Facendo cadere al massimo v briciole di pane, Jerry vuole massimizzare la differenza tra il numero di piccioni che Tom incontrerà lungo il suo intero percorso nel parco e il numero di piccioni che sono stati incontrati da lui stesso lungo il medesimo percorso. Nota che solo i piccioni che sono presenti in una statua *prima* che uno dei nostri eroi vi arrivi contano per il totale dei piccioni da lui incontrati. Il commento all'esempio fornisce ulteriori spiegazioni.

Limiti

- $1 \leq n \leq 10^5$
- $0 \leq v \leq 100$
- $0 \leq p_i \leq 10^9$

Subtask 1 (20 punti)

- $1 \leq n \leq 10$

Subtask 2 (20 punti)

- $1 \leq n \leq 1000$

Subtask 3 (30 punti)

- Una sequenza di percorsi ottima inizia dalla statua 1.

Subtask 4 (30 punti)

- Nessuna limitazione ulteriore.

Input

La prima riga contiene il numero di statue n e il numero di briciole di pane v separate da uno spazio. La seconda riga contiene n interi separati da uno spazio, ovvero $p_1 \ldots p_n$. Le successive $n-1$ righe descrivono i percorsi con coppie di numeri a_i e b_i, che indicano che c'è un percorso tra le statue a_i e b_i.

Output

Stampa esattamente un numero, la massima possibile differenza tra il numero di piccioni che Tom incontra e il numero di piccioni che Jerry incontra.

Esempio

Input	**Output**

`12 2`
`2 3 3 8 1 5 6 7 8 3 5 4`
`2 1`
`2 7`
`3 4`
`4 7`
`7 6`
`5 6`
`6 8`
`6 9`
`7 10`
`10 11`
`10 12`
	`36`

Commento

Una possibile soluzione è la seguente. Jerry entra nel parco alla statua 6. Lì incontra 5 piccioni. Lascia cadere una briciola di pane, e quindi ora p_6 è 27 e $p_5 = p_7 = p_8 = p_9 = 0$. Dopodichè corre verso la statua 7 e incontra 0 piccioni. Lascia cadere la seconda briciola di pane. p_7 ora è 41 e $p_2 = p_4 = p_6 = p_{10} = 0$. Esce dal parco, avendo incontrato $5 + 0 = 5$ piccioni. Tom lo insegue lungo la stessa sequenza di percorsi ma incontra $p_6 + p_7 = 0 + 41 = 41$ piccioni. La differenza è $41 - 5 = 36$.