Fai cadere tutte le tessere del domino (`domino` [72 punti])

“Può essere che la pigrizia si riveli essere la via se non la più sublime delle virtù.”

Un vettore h di interi positivi riporta le altezze in metri di una fila di n tessere di domino disposte lungo una linea, a distanza di 1 metro l'una dall'altra. Diremo che la tessera j è coperta dalla tessera $i < j$ se e solo se $i + h[i] > j$. Sappiamo tutti come il gioco funziona: la tessera 0 viene ad un certo punto spinta da un agente esterno per farla cadere lungo la linea nella direzione delle altre tessere e, in generale secondo un processo a valanga, quando una tessera i cade sono poi tenute a cadere, nella stessa direzione, tutte le tessere che essa copre.

Pierino si è accorto che purtroppo non tutte le tessere cadranno. Fortunatamente sei ancora in tempo a modificare (solo incrementi interi) l'altezza di una o più tessere in modo che le tessere cadano tutte fino all'ultima. Con quanta efficienza sai rispondere alle seguenti domande?

[**single_act**]: Puoi modificare l'altezza di una singola tessera a tua scelta. Stabilisci il minimo incremento in altezza che è necessario apportare ad una tessera opportunamente scelta affinchè ogni tessera cada. Specifica inoltre una qualsisi tessera che basti innalzare di tale valore.

[**multi_acts**]: Vuoi minimizzare la somma degli incrementi apportati. Specifica le nuove altezze delle tessere.

Input

Si legga l'input da `stdin`. La prima riga contiene T, il numero di testcase (istanze) da risolvere. Seguono T istanze del problema, dove ogni istanza è descritta in due righe: la prima dà il numero di tessere in fila ($n \in \mathbb{N}$), la seconda riporta gli n numeri naturali $h_0, ..., h_{n-1}$.

Output

Per ciascuna istanza, prima di leggere l'istanza successiva, scrivi su `stdout` il tuo output così strutturato:

[**goal 1, single_act**]: la prima riga contiene due numeri naturali separati da spazio: l'entità del singolo incremento da apportare e l'indice della tessera interessata.

[**goal 2, multi_acts**]: la seconda riga contiene, nell'ordine e separate da spazi, le n nuove altezze delle tessere dopo che l'altezza di alcune di esse sia stata incrementata in modo che tutte le tessere cadano, avendo cura di minimizzare la somma degli incrementi spesi.

Oltre alle dimensioni delle istanze, ogni subtask precisa quanti punti competono a ciascuno dei due goal. Per ogni istanza di quel subtask si otterranno tutti i punti dei goal correttamente evasi (purchè si rispetti almeno il formato in tutte le righe di output, incluse quelle che competono agli altri goal – altrimenti salta il protocollo di comunicazione tra il tuo programma risolutore e il server).

Esempio di Input/Output

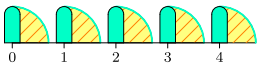
Input da `stdin`

```
-start-
3
5
1 1 1 1 1
-more-
```

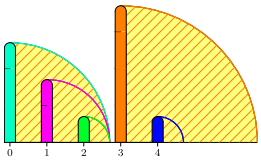
```
5
3 2 1 4 1
4
1 2 1 1

 -end-
```

Output su `stdout`

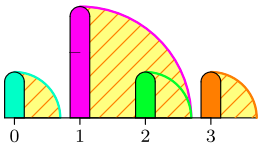
```
4 0
2 2 2 2 1
1 2
3 2 2 4 1
3 0
2 2 2 1
```



Prima istanza: è necessario agire sulla prima delle tessere. Col goal **single_act** dobbiamo alzarla di 4 se vogliamo assicurarci che anche l'ultima tessera cada. Col goal **multi_acts**, pur potendo agire su più tessere, non è possibile ridurre la somma degli incrementi al di sotto di 4 ed avremmo pertanto potuto confermare la modifica proposta per il primo goal anche per il secondo. Nell'output di esempio nel testo abbiamo preferito offrire una seconda soluzione ottima: alzare di 1 le 4 tessere $0, 1, 2, 3$.



Seconda istanza: potendo agire su una singola tessera (goal **single_act**) conviene alzare di una singola unità una qualsiasi delle prime tre tessere (indici tra 0 e 2). Col goal **multi_acts** le possibili soluzioni ottime rimangono queste stesse 3. Nell'output di esempio del testo abbiamo scelto di agire sulla tessera 2 per entrambi i goal.



Terza istanza: è necessario agire sulla prima delle tessere. Col goal **single_act** dobbiamo alzarla di 3. Col goal **multi_acts** riduciamo a 2 la somma degli incrementi. Vi sono due modi per farlo: alzare di 1 le tessere 0 e 1 oppure le tessere 0 e 2.

Subtask

Il tempo limite per istanza (ossia per ciascun testcase) è sempre di 1 secondo.

I testcase sono raggruppati nei seguenti subtask.

1. **[6 pts← 3 istanze da 1 + 1 punti] `esempi_testo`:** i tre esempi del testo
2. **[6 pts← 3 istanze da 1 + 1 punti] `small_bad_start`:** $n \leq 20, h[0] = 0$
3. **[6 pts← 3 istanze da 1 + 1 punti] `medium_bad_start`:** $n \leq 500, h[0] = 0$
4. **[6 pts← 3 istanze da 1 + 1 punti] `big_bad_start`:** $n \leq 200,000, h[0] = 0$
5. **[6 pts← 3 istanze da 1 + 1 punti] `small_just_one_meter`:** $n \leq 20, h[0] = 0$
6. **[6 pts← 3 istanze da 1 + 1 punti] `medium_just_one_meter`:** $n \leq 500, h[0] = 0$
7. **[6 pts← 3 istanze da 1 + 1 punti] `big_just_one_meter`:** $n \leq 200,000, h[0] = 0$
8. **[10 pts← 5 istanze da 1 + 1 punti] `small`:** $n \leq 20, h[0] = 0$
9. **[10 pts← 5 istanze da 1 + 1 punti] `medium`:** $n \leq 500, h[0] = 0$
10. **[10 pts← 5 istanze da 1 + 1 punti] `big`:** $n \leq 200,000, h[0] = 0$

In generale, quando si richiede la valutazione di un subtask vengono valutati anche i subtask che li precedono, ma si evita di avventurarsi in subtask successivi fuori dalla portata del tuo programma che potrebbe andare in crash o comportare tempi lunghi per ottenere la valutazione completa della sottomissione. Ad esempio, chiamando[1, 2] :

```
rtal -s <URL>  connect -x <token> -a size=small_just_one_meter
               domino  -- python my_solution.py
```

[1] <URL> server esame: `wss://ta.di.univr.it/esame`

[2] <URL> server esercitazioni e simula-prove: `wss://ta.di.univr.it/algo`

vengono valutati, nell'ordine, i subtask:

`esempi_testo`, `small_bad_start`, `medium_bad_start`, `big_bad_start`, `small_just_one_meter`.

Il valore di default per l'argomento `size` è `big` che include tutti i testcase.

Consigli

1. Un espediene classico per entrare nella struttura di un problema consta nell'isolare casi particolari e porsi buone domande. I subtask propongono casi particolari nella speranza che essi possano aiutare anche in questo senso.
2. Sia nella comprensione che nella codifica, puoi sempre affrontare il problema per gradi, ad esempio partendo dal goal che ti è più congeniale.