Poldo si concede il bis (`poldo_bis` [86 punti])

Sia $\mathbb{N}_n = \{0, 1, ..., n-1\}$ l'insieme degli indici di una sequenza $S = s_0, ..., s_{n-1}$ di n interi non-negativi. Dove il valore di un $I \subseteq \mathbb{N}_n$ è dato dalla massima cardinalità di un sottoinsieme di I che non contenga due interi consecutivi, trova un I di massimo valore tra quelli con le seguenti proprietà:

1. la sottosequenza di S individuata da I è strettamente crescente
2. se $i - 1 \notin I$ mentre gli interi nell'intervallo $[i, j]$ appartengono tutti ad I e sono in numero dispari, allora $j + 1 \in I$

Motivazione: Poldo ha promesso al dottore che si sarebbe attenuto a una qualche dieta di propria scelta. Ha quindi adottato la seguente regola: d'ora in poi, nella sequenza di panini che mangerà nell'arco di un pasto, il contenuto di colesterolo dovrà sempre aumentare. (Questo spiega la Condizione 1.) Per festeggiare, l'amico Bracciodiferro lo porta al Mac-All-You-Can-Donald dove, Poldo, attento al colesterolo, riguarda i panini in arrivo sul nastro trasportatore come alla sequenza S di cui sopra. Si accorge ben presto che quando decide di mangiarsi il panino i ed $S_{i+1} > S_i$ non è proprio capace di non mangiarsi anche il panino $i + 1$, anche perchè la sua dieta lo consente. Riesce però, se lo vuole, a non proseguire col panino $i + 2$ dato che quella di mangiarsi il panino $i + 1$ non è stata davvero una sua decisione e così niente spike di dopamina (ma se decide di sua sponte di mangiare il panino $i + 2$ rientrerà in modalità compulsiva e non potrà fare a meno di mangiarsi anche anche il panino $i + 3$ se di valore maggiore, e così via). Questo spiega perchè ha introdotto la Condizione 2 nella sua formulazione del problema. Poldo realizza quindi che questa sua debolezza, di cui ha preso atto, è un comportamento compulsivo, e, avendo cominciato a ragionare di queste cose, comprende che il secondo panino manco lo gusta davvero non essendo una sua libera scelta. (Ecco come è giunto a formulare la sua particolare nozione sul valore reale di un $I \subseteq \mathbb{N}_n$, che risulta ammissibile solo se rispetta le Condizioni 1 e 2.)

Input

Si legga l'input da `stdin`. La prima riga contiene T, il numero di testcase (istanze) da risolvere. Ogni istanza consta di due righe: la prima contiene n mentre la seconda contiene gli n numeri $s_0, s_1, ..., s_{n-1}$ nell'ordine e separati da spazio. Inclusa la prima riga, il server scriverà un totale di $2T + 1$ righe.

Output

Per ciascuna istanza, prima di leggere l'istanza successiva o per concludere la sessione col server, scrivi su `stdout` il tuo output strutturato su due righe: la prima riga contiene i due numeri $|I|$ e $\text{val}(I)$ separati da spazio mentre la seconda contiene gli $|I|$ indici contenuti in I, ordinati e separati da spazio.

suggerimento: comincia col calcolare la cardinalità e il valore di una soluzione ottima I. A ciascuna riga dell'output è associato un diverso goal, e non serve che sia corretta la risposta sulla seconda riga per ricevere i punti del primo goal (si deve comunque rispettare il formato di input e output per non far saltare la comunicazione col server).

Esempio di Input/Output

Input da `stdin`

```
3
 8
 4 1 5 3 6 2 7 7
 10
 3 4 7 5 1 6 1 8 2 9
 14
 11 10 12 10 13 10 14 10 1 2 3 4 5 6
```

Output su `stdout`

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

Spiegazione: Dei 3 testcase analizziamo insieme il secondo, quello con 10 panini. La soluzione ottima è unica e comporta che Poldo mangi 6 panini e raccolga 5 punti di soddisfazione. Raccontiamone la storia: Poldo decide di mangiare il primo panino, quello di indice 0. Essendo questa una scelta consapevole ne ricava un primo punto di soddisfazione. Siccome il panino immediatamente successivo batte il precedente in quanto a contenuto di colesterolo (4>3), Poldo non riesce a non mangiare anche questo, ma siccome il suo è stato un comportamento compulsivo non ne ricava soddisfazione alcuna. Inoltre il valore del prossimo panino che deciderà di mangiare dovrà ora eccedere 4, che rabbia! In seguito mangerà infatti un panino di valore 5 (quello di indice 4), ricavandone un secondo punto di soddisfazione in quanto atto libero. Siccome il panino immediatamente successivo non ha valore maggiore di questo, quindi Poldo non può mangiarlo (meglio, perchè avesse avuto un valore maggiore il mangiarlo non gli avrebbe dato comunque alcuna soddisfazione col solo effetto di limitarlo nelle scelte future). In seguito deciderà di sua volontà e non per compulsione (quindi acquisirà un terzo punto di soddisfazione) il panino di indice 5 e valore 6, e non dovrà mangiare il panino immediatamente successivo che vale meno. Poi idem col panino di indice 7 e valore 9, e infine con quello di indice e valore 9. In pratica, dei 6 panini che ha mangiato l'unico mangiato per compulsione che non gli ha quindi dato alcuna soddisfazione è il panino di indice 1. Conoscendo i propri limiti, se l'è comunque gestita in modo da massimizzare la soddisfazione ottenuta. Per altro questa soluzione ottima era anche unica.

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. **[10 pts← 5 istanze da 1 + 1 punti] `tiny`:** $n \leq 10$
3. **[10 pts← 5 istanze da 1 + 1 punti] `small`:** $n \leq 20$
4. **[20 pts←10 istanze da 1 + 1 punti] `medium`:** $n \leq 100$
5. **[10 pts← 5 istanze da 1 + 1 punti] `big`:** $n \leq 1000$
6. **[30 pts←15 istanze da 1 + 1 punti] `large`:** $n \leq 10000$

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
       poldo_bis  -- python my_solution.py
```

vengono valutati, nell'ordine, i subtask:

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

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

`esempi_testo`, `tiny`, `small`.

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

Consigli

1. Nota che anche una soluzione ricorsiva semplice, benchè scritta in un linguaggio interpretato quale python, ti consegna i primi tre subtask. Questo perchè il numero delle soluzioni ammissibili sarà sufficientemente contenuto da poterle di fatto considerare tutte.
2. Con la memoizzazione, ovvero con la tecnica della programmazione dinamica, può venirti naturale, se ne hai competenza, progettare ex-novo una soluzione $\Theta(n^2)$ che basta per $n \leq 1000$.
3. Per i punti platino serve progettare una soluzione $\Theta(n \log n)$. Puoi prendere spunto dagli approcci che già hai incontrato per il problema di massima sottosequenza crescente (il Poldo classico), può essere adattato sia quello di avvalersi di una struttura dati dinamica che quello di produrre un'algoritmo primale-duale, ma richiede la sua maturità ed attenzione ed è quindi un giocarsela.