DIETI UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

Corso di Laurea Magistrale in Ingegneria Informatica

CORSO DI ALGORITMI E STRUTTURE DATI

Prof. ROBERTO PIETRANTUONO

Prova del 13 Febbraio 2023

Indicazioni

Si consegni un file in **formato .txt** nominandolo *CognomeNome*.txt, in cui è riportata l'implementazione (nel linguaggio scelto) seguita da una indicazione della complessità temporale dell'algoritmo implementato (complessità nel caso peggiore, è sufficiente il limite superiore O(f(n))). Se si utilizzano librerie di cui non si conosce la complessità, lo si indichi nella spiegazione (ad esempio, "la complessità è O(n log n) al netto della complessità dell'algoritmo *x,* che è non nota"). Se si utilizza la randomizzazione, si indichi anche il tempo di esecuzione atteso.

PROBLEMA

Si consideri un algoritmo di tipo Divide et Impera *D* che opera su un array di interi con $1 \leq n \leq 1000$ elementi. Al primo passo l'algoritmo divide l'array in due parti; successivamente continua a dividere solo uno dei due sotto-array. Il numero di divisioni da fare e l'insieme dei *pivot* da usare (ossia i punti in cui si dovrà suddividere l'array) sono dati in ingresso. Ad ogni operazione di suddivisione fatta da *Alg* è associato un costo, che è pari al numero di elementi dell'array (o del sotto-array) da suddividere. È facile notare che selezioni diverse nell'ordine dei *pivot* (ossia dove suddividere prima) possono portare a costi diversi. Ad esempio, si consideri un array di 100 elementi, e siano i possibili *pivot* 25, 50 e 75. Ci sono diverse scelte. L'algoritmo *D* può suddividere prima a 25, poi a 50, poi a 75. Questo porta ad un costo di 100 elementi + 75 elementi + 50 elementi = 225 perché il primo array era di 100 elementi, il secondo, risultante dalla prima suddivisione, di 75 e l'ultima, risultante dalla seconda suddivisione, di 50. Un'altra scelta potrebbe essere scegliere come *pivot* 50, poi a 25, poi a 75. Questo porterebbe a un costo di 100 + 50 + 50 = 200, che è migliore. Si progetti un algoritmo per determinare il costo minimo che *D* impiega per suddividere l'array il numero di volte indicato.

INPUT

L'input è costituito da diversi test. La prima riga di ogni test case conterrà un numero intero positivo *n* che rappresenta la lunghezza dell'array. Si assuma $1 \leq n \leq 1000$. La riga successiva conterrà il numero intero *p* ($n < 50$) di pivot da usare.
La riga successiva è composta da *p* numeri interi positivi c_i ($0 < c_i < n$) che rappresentano i *pivot*, dati in ordine strettamente crescente.
Un caso con $n = 0$ rappresenta la fine dell'input.

OUTPUT

Bisogna stampare il costo della soluzione ottima, cioè il costo totale minimo delle suddivisioni.

Sample Input

```
30
3
2 20 25
10
4
4 5 7 8
```

100
3
25 50 75
0

Sample Output

60
22
200