Esercizio 1: grado di popolarità

Sia A[1;n] una sequenza contenente il grado di popolarità degli n amici di una persona P.
Definiamo il grado di popolarità di una persona P come il più grande numero h tale che P ha almeno h amici ciascuno con grado di popolarità almeno h.
L'array A quindi contiene n interi non-negativi.
Si scriva un algoritmo che calcoli il grado di popolarità di P e se ne analizzi la complessità.

Quindi in output vogliamo
h = max{j | P ha almeno j amici di popolarità almeno j, per 0≤j≤n}.
Si analizzi la complessità dell'algoritmo proposto.
Esempio:

A=

1	0	5	8	10	2	12	6	0
1	2	3	4	5	6	7	8	9

grado di popolarità

Quindi in output vogliamo
h = max{j | P ha almeno j amici di popolarità almeno j, per 0≤j≤n}.
Esempio:

A=

1	0	5	8	10	2	12	6	0
1	2	3	4	5	6	7	8	9

per j = 1 ci sono 7 amici con grado di popolarità almeno 1
per j = 2 sono 6 ≥2
per j = 3 sono 5 ≥3
per j = 4 sono 5 ≥4
per j = 5 sono 5
per j = 6 sono 4 quindi non va, dovrebbero essere almeno 6

La risposta è 5 che è il massimo valore di j per cui ci sono almeno j amici con grado di popolarità almeno j.

Esercizio di analisi di un algoritmo

Si consideri il seguente frammento di codice:

```
count = 0
n = A.length
ArraySort(A)
for  i = 1 to  n  do
    if (binarySearch(A, A[1]+A[i])) then count = count+1
```

Si analizzi asintoticamente il tempo di esecuzione del frammento di codice sia nell'ipotesi che ArraySort sia il SelectionSort, il bubbleSort, l'insertionSort, l'heapSort o il quickSort specificando il caso peggiore e il caso migliore.