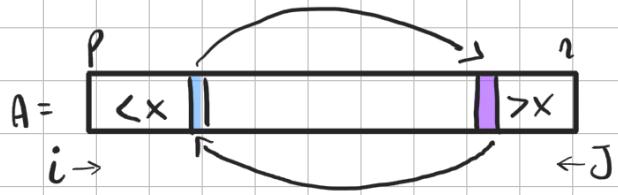


PARTIZIONA(A, p, r)



Fissato l'elemento x (appartenente ad A), rispetto al quale partizioniamo l'array per trovare q , prendiamo due indici $i = p - 1$ e $j = r + 1$. Incrementiamo i finché non troviamo un valore maggiore o uguale di x , specularmente faremo con j .

Terminati entrambi gli indici, finché $i < j$, scambiamo gli oggetti in posizione i e j e ripetiamo ogni volta che fermiamo lo scambio, ovvero quando troviamo oggetti che si trovano della parte sbagliata dell'array.

Ripetiamo quest'operazione finché vale $i < j$

PARTIZIONA(A, p, r)

$$x = A[p]$$

$$i = p - 1$$

$$j = r + 1$$

REPEAT

REPEAT

Sceglio come elemento x il primo dell'array

J--

Scanni j verso l'interno

UNTIL ($A[j] \leq x$)

Finche' non troviamo un elemento da scambiare

REPEAT

i++

Scanni i verso l'interno

UNTIL ($A[i] \geq x$)

Finche' non troviamo un elemento da scambiare

IF ($i < j$) THEN

Se non abbiamo finito di scannere l'array

SWAP (A, i, j)

Scambia i due elementi

UNTIL ($i \geq j$)

Dobbiamo returnare quello che sara' la

fine del nostro prossimo partizionamento

della prima metà, ovvero J. I si potrebbe

trovare già nella seconda metà

OSS

Nel caso di due valori uguali non ci complica le cose
perche' alla fine delle chiamate ricorsive x verrà portato nella sua
posizione corretta. Questa ambiguità è data dal fatto che le
condizioni di scannimento si fermano entrambe quando viene trovato
un valore uguale al pivot (x), escludendo questa condizione
l'algoritmo non funzionerebbe ignorando i valori uguali al
pivot

VERIFICA DELLE PROPRIETA'

Dimostriamo che il nostro algoritmo PARTIZIONA(A, p, r) verifica le due proprietà che vedevamo ovvero:

$$\textcircled{1} \quad p \leq q < r$$

$$\textcircled{2} \quad \forall p \leq i \leq q \quad \forall q+1 \leq j \leq r \quad A[i] \leq A[j]$$

In particolare noi vogliamo che siano verificate alla fine dell'esecuzione dell'algoritmo

\textcircled{1} J parte da $r+1$ quindi non è verificata una "condizione interna" ovvero che $p \leq j < r$.

Questa non è verificata solo se $i = j$ si fermano in r .

J resta fermo su r e il suo ciclo viene ripetuto solo una volta (un solo decremento). i viene incrementato e si trova in p dove si trova la nostra x . Sicuramente l'elemento in r è $\leq x$ (perché j si è fermato) quindi viene scambiato con l'elemento in p (x) che verifica sempre la condizione di uscita del ciclo di i ($A[i] = x \geq x$ sempre)

Anche lo swap il programma controlla che $i < j$ per continuare, ma questo avviene sicuramente e verrà eseguito il ciclo, quindi j decrementerà nuovamente quindi sarà $< j$.

Siccome alla fine dell'algoritmo restiamo j , in quick sort avremo $p \leq j \leq r$ (per nostra scelta $p \neq 1$)

② Al termine dell'algoritmo $i = j$ (uguale se si fermano sul pivot),
a sinistra di j sono tutti $< x$ e a destra $\geq x$

Abbiamo due casi per cui i e j si incrociano (l'algoritmo termina)

- j sovrappone i sconnuovo (i blocca sempre j) ma i incrementa dopo lo scambio di j quindi incrementerà di 1 e si fermerà subito perché a destra di j ci sono solo elementi $\geq x$
- i sovrappone j e si ferma in j o $j+1$ perché j ha già verificato tutte le proprietà alla sua destra.

In entrambi i casi al termine della partizione avremo:

$$\forall p \leq s \leq j \quad A[s] \leq x$$

$$\forall j+1 \leq t \leq r \quad x \leq A[t]$$

Se applichiamo la transitività del \leq avremo

$A[s] \leq A[t]$ quindi tutti gli oggetti a sinistra di x sono minori di quelli a destra di x , che è proprio la proprietà ②

CORRETEZZA DI QUICK SORT

QUICKSORT è un algoritmo ricorsivo, dev'essere verificata la veridicità di questo:

per fare ciò utilizziamo il 2° principio d'induzione.

In generale se vogliamo dimostrare la veridicità di $P(x) \forall x \in \mathbb{N}$ dobbiamo dimostrare che:

① $P_{(0)}$ è vera

② $\forall x > 1 \quad ((\forall 0 \leq y < x \quad P_{(y)}) \Rightarrow P_{(x)})$

Nel nostro $P_{(x)} = (QS(A, m) = A' \Rightarrow A' \text{ è ordinata})$

① $P_{(0)}, P_{(1)}$

Quicksort è banalmente verificata su array vuoti o composti da 1 elemento

② Fissato $m > 1$ arbitrario

$$P_{(m)} = QS(A, 0, m) = A' = QS(A, p, q) + QS(A, q+1, r)$$

L'espressione $\forall 0 \leq y < m \quad P_{(y)}$ significa nel nostro caso che per ogni sequenza di lunghezza compresa tra 0 e y dev'essere verificata.

Per $0 \leq y \leq 1$ già l'abbiamo dimostrato, per $y=2$ QS applica QS

su due sequenze di lunghezza 1 e le unisce quindi sicuramente e' verificata. Per $y=3$ partizionamento ci darà una sequenza di lunghezza 1 e 2 quindi per quanto già detto è verificata.

Ogni istanza può essere suddivisa in istanze minori quindi sarà sempre verificata, questo implica (per ②) $P_{(n)}$.