Quicksort

```
QUICKSORT(A, p, r)
IF p < r THEN
        q=PARTITION(A, p, r)
        QUICKSORT(A, p, q − 1)
        QUICKSORT(A, q + 1, r)
```

Aby posortować tablicę wywołujemy QUICKSORT$(A, 0, n-1)$.

QuickSort

```
PARTITION(A, p, r)
pivot = A[r]
smaller = p
FOR j = p TO r − 1 DO
    IF A[j] ≤ pivot THEN
        Swap A[smaller] and A[j]
        smaller = smaller + 1
    FI
OD
Swap A[smaller] and A[r]
RETURN smaller
```

Przykład

28713564

Przykład

2871356 4

pivot =4, smaller =0.

Przykład

2 8 7 1 3 5 6 4

2<=pivot, smaller=1

Przykład

2 8 71356 4

8 >pivot

Przykład

28 7 1356 4

$7 > pivot$

Przykład

2<u>8</u>7 1 356 4

1<=*pivot*, smaller=2

Przykład

2 8 7 1 3 5 6 4

2 1 7 8 3 5 6 4

Przykład

2178 3 56 4

3<=*pivot*, smaller=3

Przykład

2178 3 56 4

2138 7 56 4

Przykład

2138754 6 4

5>*pivot*

Przykład

213875 6 4

6>*pivot*

Przykład

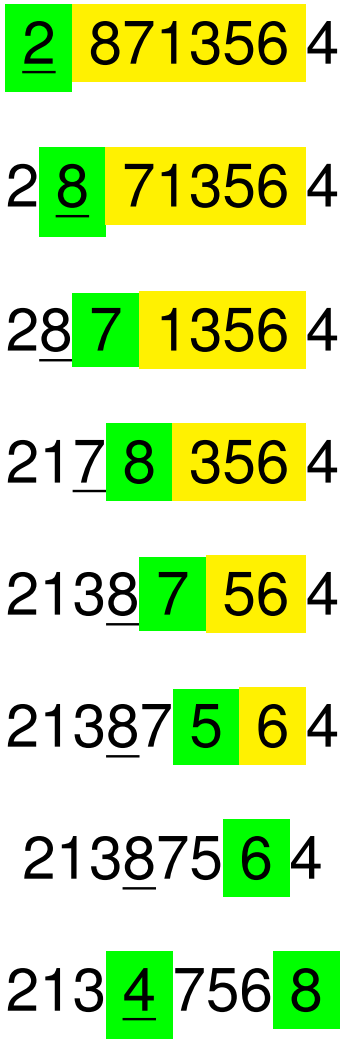
213 8 756 4

zamień $A[r]$ i $A[smaller]$

Przykład

213 4 756 8

Przykład


2 871356 4
2 8 71356 4
28 7 1356 4
217 8 356 4
2138 7 56 4
21387 5 6 4
213875 6 4
213 4 756 8

Przyklad

Rekurencyjne wywołanie dla $[2, 1, 3]$ oraz $[7, 5, 6, 8]$.

QuickSort

- $T(n) = \Omega(n \log n)$.
- $T(n) = O(n^2)$.

Quicksort wersja Hoare'a

PARTITION(A, p, r)
$pivot = A[p]$
$i = p - 1$
$j = r + 1$

while TRUE DO
    repeat j=j-1 until $A[j] \leq pivot$
    repeat i=i+1 until $A[i] \geq pivot$
    if i<j
        Swap $A[i]$ and $A[j]$
    else return j

Zadanie 1

Napisz program, który wyznaczy punkt podziału dla zadanej tablicy oraz indeksów wejściowych p i q.

Zadanie 2

Napisz program, który tablicę nieposortowanych liczb ułoży w ten sposób, że najpierw będą elementy dodatnie (w kolejności występowania), a następnie elementy ujemne (w kolejnosci występowania). Rozwiąż zadanie bez użycia dodatkowych tablic/list. Nie modyfikuj rozmiaru tablicy wyjściowej (nie można dodawać usuwać elementów)

sortowania w czasie liniowym

Counting-sort(A,B,k)

```
for i <- 0 to k
      do C[i] <- 0
for j <- 1 to length[A]
      do C[A[j]] <- C[A[j]]+1
for i <- 1 to k
      do C[i] <- C[i]+ C[i-1]
for j <- length[A] downto 1
      do B[C[A[j]]] <- A[j]
      C[A[j]] <- C[A[j]] - 1
```

Przykład

6,0,2,0,1,3,4,6,1,3,2

Bucket-sort(A)

n <- length[A]
for i <- 1 to n
    do wstaw A[i] na listę B[floor(nA[i])]
for i <- 0 to n-1
    do posortuj listę B[i] przez wstawianie
połącz listy B[0], B[1], .. , B[n-1] z zachowaniem kolejności

Przykład

0,78 ; 0,17; 0,39; 0,26; 0,72; 0,94; 0,21; 0,12; 0,23; 0,68