Quick sort - sortowanie szybkie

Omawiany algorytm należy do jednego z najszybszych algorytmów sortujących dane wynaleziony w 1960 roku przez Sir Charles Antony Richard Hoare. Jest on chętnie implementowany i wdrażany do systemów informatycznych ze względu na szerokie spektrum zalet:

- złożoność czasowa jest rzędu O(nlogn)
- algorytm nie potrzebuje dodatkowej tablicy do posortowania jak to jest w przypadku sortowania przez scalanie

- jest łatwy w implementacji
- dobrze współpracuje z różnymi typami danych

Algorytm posiada także wady:

- w sytuacji pesymistycznej złożoność może wynosić $O(n^2)$
- jest niestabilny
- jest wrażliwy na błędy w implementacji

Działanie algorytmu

Algorytm wykorzystuje technikę "*dziel i zwyciężaj*". Według ustalonego schematu wybierany jest jeden element w sortowanej tablicy, który będziemy nazywać
pivot.
Pivot może być elementem środkowym, pierwszym, ostatnim, losowym lub wybranym według jakiegoś innego schematu dostosowanego do zbioru danych. Następnie ustawiamy elementy nie większe na lewo tej wartości, natomiast nie mniejsze na prawo. W ten sposób powstaną nam dwie części tablicy (niekoniecznie równe), gdzie w pierwszej części znajdują się elementy nie większe od drugiej. Następnie każdą z tych podtablic sortujemy osobno według tego samego schematu.

Przykład

Do posortowania posłużymy się przykładem ciągu liczb:

2 5 1 3 4 0 6 2 5

W naszym algorytmie element **pivot** będzie środkowym elementem sortowanej tablicy

lewa część tablicy

pivot

prawa część tablicy

2

5

1

3

4

0

6

2

5

Następnie ustawiamy liczniki **i** na pierwszy element tablicy, natomiast **j** na ostatni element tablicy:

lewa część tablicy

pivot

prawa część tablicy

2

5

1

3

4

0

6

2

5

i

j

W kolejnym kroku szukamy pierwszej liczby z lewej strony tablicy poruszając się licznikiem **i** ku wartości
pivot
, która jest nie mniejsza niż
pivot
oraz pierwszej liczby z prawej strony, która jest nie większa niż
pivot.
Stan liczników po wyszukaniu odpowiednich liczb:

lewa część tablicy

pivot

prawa część tablicy

2

5

1

3

4

0

6

2

5

i

j

Teraz wykonujemy zamianę liczb, które stoją po niewłaściwej stronie:

lewa część tablicy

pivot

prawa część tablicy

2

2

1

3

4

0

6

5

5

i

j

Czynności powtarzamy do momentu minięcia się liczników:

lewa część tablicy

pivot

prawa część tablicy

2

2

1

3

4

0

6

5

5

i

j

Zamiana pivota z zerem

lewa część tablicy

pivot

prawa część tablicy

2

2

1

3

0

4

6

5

5

i

j

i minięcie się liczników

lewa część tablicy

pivot

prawa część tablicy

2

2

1

3

0

4

6

5

5

lewy

j

i

prawy

W kolejnym kroku wykonujemy sortowanie dwóch tablic osobno, zawierających się w przedziałach:

pierwsza tablica:[lewy..j]:2 2 1 3 0

druga tablica:[i..prawy] :4 6 5 5.

Zauważmy, że w pierwszej tablicy znajdują się elementy nie większe od drugiej.

Lewa część tablicy

pivot

2

2

1

3

0

i

j

0

2

1

3

2

i

j

pivot

0

1

2

3

2

j

i

Po lewej stronie **pivota** jest tylko jeden element, więc stoi on już na odpowiedniej pozycji. Prawą część tej podtablicy dzielimy ponownie na dwie części:

pivot

2

3

2

i

j

2

3

2

i

j

pivot

2

2

3

j

i

Algorytm dla tej części podtablicy musi się wykonać jeszcze dla dwóch dwójek, ponieważ sortowanie kończy się, gdy podtablica składa się z co najwyżej jednego elementu. W tym przypadku liczby są już posortowane, więc dalsza wizualizacja jest zbędna.

Prawa część tablicy

pivot

4

6

5

5

i

j

4

6

5

5

i

j

pivot

4

5

5

6

i **=** **j**

4

5

5

6

j

i

Po tym minięciu się liczników, tablica zostanie jeszcze podzielona na dwie części, ale są one już posortowane, więc na tym kroku zakończymy.

Gdy popatrzymy globalnie na całą tablice, to zauważymy, że wszystkie jej elementy są poukładane w odpowiedniej kolejności.