Algorytmy

Grzegorz Koperwas Mateusz Randak Piotr Kołodziejski

28 października 2020

Czym jest algorytm?

Definicja

Algorytm to zestaw instrukcji opisujących krok po kroku jak wykonać pewne zadanie.

Mamy z nimi do czynienia w codziennym życiu wykonując różne czynności według pewnych schematów. Przykładowo gdy gotujemy i podążamy za pewnym przepisem.
Algorytmy odgrywają niezwykle ważną rolę w informatyce. Są one potrzebne do optymalizacji oprogramowania, tak aby działało jak najefektywniej. Specyficznnym rodzajem algorytmów są algorytmy rekurencyjne.

Definicja

Rekurencja polega na odwołaniu się funkcji do samej siebie.

Wyszukiwanie binarne:

Wyszukiwanie binarne - algorytm stosowany do wyszukiwania elementów w posegregowanych zbiorach.
Cechuje się złożonością czasową $O(\log_2 n)$, podczas gdy wyszukiwanie liniowe (element po elemencie), ma złożoność $O(n)$.

Definicja

Złożoność czasowa - Jak długo zajmuje wykonanie algorytmu w zależności od wielkości wejścia (n)

Zasada działania

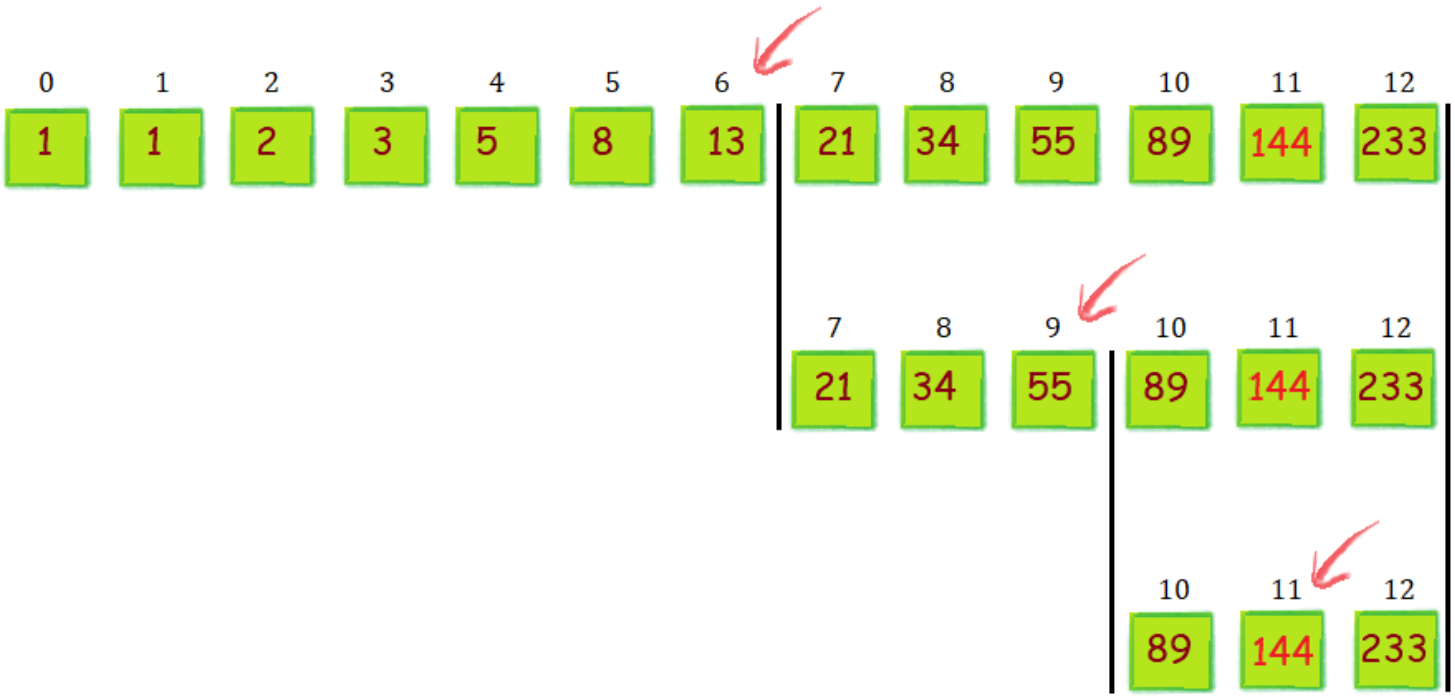
Główną ideą wyszukiwania binarnego jest podział elementów w przeszukiwanym uporządkowanym zbiorze danych (np. tablicy) na coraz to mniejsze zbiory, tak by optymalnie ograniczyć zakres poszukiwania.

1. gdy środkowy element jest mniejszy od szukanej, za dolną granicę uznaje się kolejny element, a górna pozostaje bez zmian
2. gdy środkowy element jest większy od szukanej, za górną granicę uznaje się poprzedni element, a dolna pozostaje bez zmian

Następnie powyższe kroki zostają powtórzone. Jeżeli element nie zostanie znaleziony, zwracana jest odpowiednia wartość wskazująca na błąd (np. -1 lub *null*).

```
# searches for specified item in chosen collection
def binary_search(item, collection):
    lower_bound = 0
    upper_bound = len(collection) - 1

    while lower_bound <= upper_bound:
        mid = (lower_bound + upper_bound) // 2

        if collection[mid] == item:
            return mid

        if collection[mid] < item:
            lower_bound = mid + 1
        else:
            upper_bound = mid - 1

    # if item not found
    raise Exception("not found")
```

Przykład:


0 1 2 3 4 5 6 7 8 9 10 11 12
1 1 2 3 5 8 13 21 34 55 89 144 233
7 8 9 10 11 12
21 34 55 89 144 233
10 11 12
89 144 233

Quicksort:

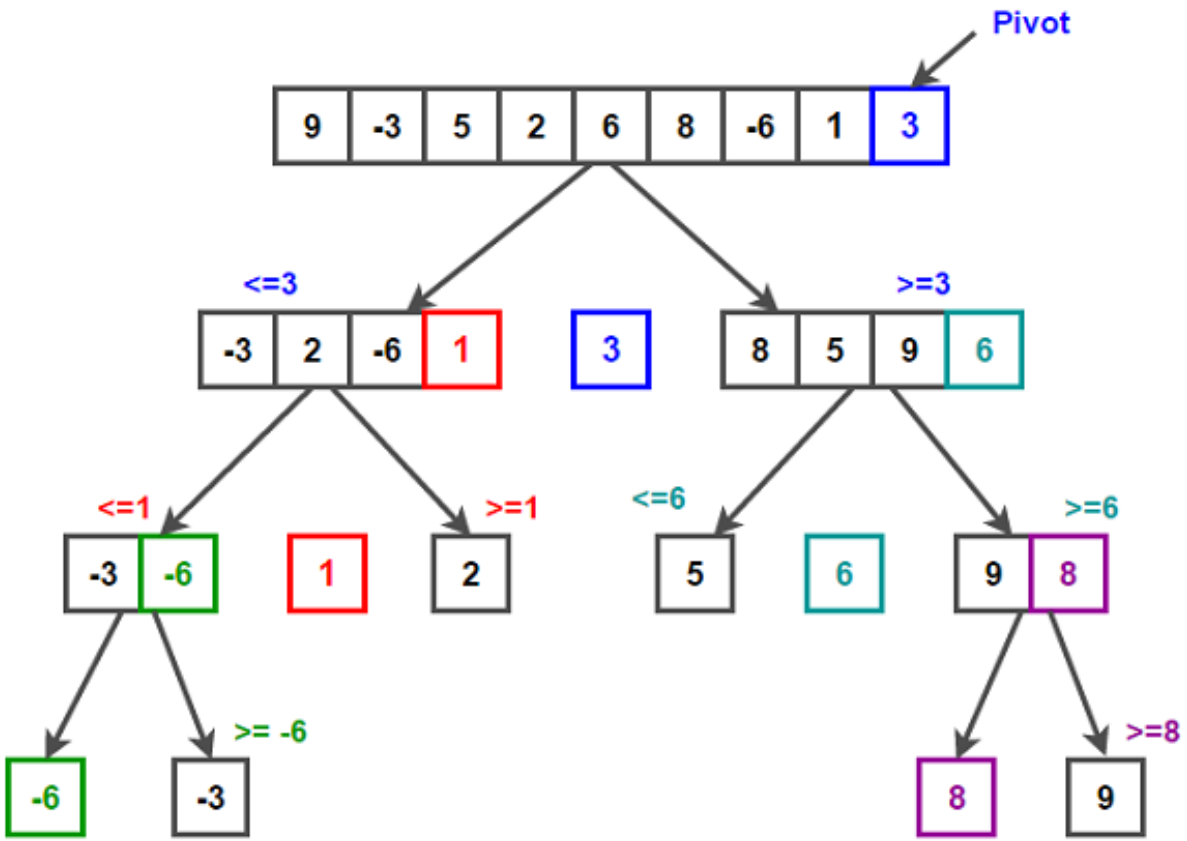
Quicksort to rekurencyjny algorytm sortujący, działający poprzez dzielenie początkowego zbioru na coraz mniejsze podzbiory.

Działanie:

1. Wybierz element z *danego zbioru*, zwany *elementem rozdzielającym.*
2. Elementy mniejsze od *elementu rozdzielającego* przenieś do zbioru A, większe do C, a równe do B.
3. Posortuj zbiory A i C quicksortem.
4. Wynikiem jest połączenie zbiorów $A + B + C$.

```
def quicksort(zbiór):
    if len(zbiór) <= 1:
        # nie trzeba sortować zbiorów pustych
        return zbiór
    else:
        element_rozdzielający = zbiór[-1] # ostatni element
        a = list() # mniejsze
        b = list() # równe
        c = list() # większe
        for element in zbiór:
            if element < element_rozdzielający:
                a.append(element)
            elif element > element_rozdzielający:
                c.append(element)
            else:
                b.append(element)
        a = quicksort(a)
        c = quicksort(c)
        return a + b + c
```

Przykład:



Koniec

Dziękujemy za uwagę!