Algorytmy wyszukiwania

Wyszukiwanie liniowe/sekwencyjne- (ang. linear search), zwane również sekwencyjnym (ang. sequential search) polega na przeglądaniu kolejnych elementów zbioru Z. Jeśli przeglądany element posiada odpowiednie własności (np. jest liczbą o poszukiwanej wartości), to zwracamy jego pozycję w zbiorze i kończymy. W przeciwnym razie kontynuujemy poszukiwania aż do przejrzenia wszystkich pozostałych elementów zbioru Z.

W przypadku pesymistycznym, gdy poszukiwanego elementu nie ma w zbiorze lub też znajduje się on na samym końcu zbioru, algorytm musi wykonać przynajmniej n obiegów pętli sprawdzającej poszczególne elementy. Wynika z tego, iż pesymistyczna klasa złożoności obliczeniowej jest równa O(n), czyli jest liniowa - stąd pochodzi nazwa metody wyszukującej.

Często chcemy znaleźć wszystkie wystąpienia w zbiorze poszukiwanej wartości elementu. W takim przypadku algorytm na wejściu powinien otrzymywać dodatkowo pozycję (indeks) elementu, od którego ma rozpocząć wyszukiwanie. Pozycję tę przy kolejnym przeszukiwaniu podajemy zawsze o 1 większą od ostatnio znalezionej. Dzięki temu nowe poszukiwanie rozpocznie się tuż za poprzednio znalezionym elementem.

Algorytm wyszukiwania liniowego/sekwencyjnego

Wejście

n - liczba elementów w tablicy Z[ ], n ∈N

Z[ ] - tablica zawierająca elementy do przeszukania. Indeksy elementów rozpoczynają się od 0, a kończą na n-1

p - indeks pierwszego elementu Z[ ], od którego rozpoczniemy poszukiwania. p ∈C

k - poszukiwana wartość, czyli tzw. klucz, wg którego wyszukujemy elementy w Z[ ]

Wyjście:

Pozycja elementu zbioru Z[ ] o kluczu k lub -1 w przypadku nie znalezienia elementu.

Zmienne pomocnicze

i - przebiega przez kolejne indeksy elementów Z[ ]. i ∈C

Lista kroków:

K01: Dla i = p,p+1,...,n-1: wykonuj K02 ; przeglądamy kolejne elementy w zbiorze

K02: Jeśli Z[i] = k, to zakończ zwracając i ; jeśli napotkamy poszukiwany element, zwracamy jego pozycję

K03: Zakończ zwracając -1 ; jeśli elementu nie ma w tablicy, zwracamy -1

Wyszukiwanie binarne

Problem

W posortowanym rosnąco zbiorze Z wyszukać element o wartości (kluczu) k.

Postąpimy w sposób następujący

Wyznaczymy element środkowy zbioru. Sprawdzimy, czy jest on poszukiwanym elementem. Jeśli tak, to element został znaleziony i możemy zakończyć poszukiwania. Jeśli nie, to poszukiwany element jest albo mniejszy od elementu środkowego, albo większy. Ponieważ zbiór jest uporządkowany, to elementy mniejsze od środkowego będą leżały w pierwszej połówce zbioru, a elementy większe w drugiej połówce. Zatem w następnym obiegu zbiór możemy zredukować do pierwszej lub drugiej połówki - jest w nich o połowę mniej elementów. Mając nowy zbiór postępujemy w sposób identyczny - znów wyznaczamy element środkowy, sprawdzamy czy jest poszukiwanym elementem. Jeśli nie, to zbiór dzielimy znów na dwie połowy - elementy mniejsze od środkowego i elementy większe od środkowego. Poszukiwania kontynuujemy w odpowiedniej połówce zbioru aż znajdziemy poszukiwany element lub do chwili, gdy po podziale połówka zbioru nie zawiera dalszych elementów.

Ponieważ w każdym obiegu pętli algorytm redukuje liczbę elementów o połowę, to jego klasa złożoności obliczeniowej jest równa O(log n). Jest to bardzo korzystna złożoność. Na przykład w algorytmie wyszukiwania liniowego przy 1000000 elementów należało wykonać aż 1000000 porównań, aby stwierdzić, iż elementu poszukiwanego nie ma w zbiorze. W naszym algorytmie wystarczy 20 porównań.

Oczywiście algorytm wyszukiwania liniowego może być zastosowany dla dowolnego zbioru. Nasz algorytm można stosować tylko i wyłącznie w zbiorze uporządkowanym. Ze względu na podział zbioru na kolejne połówki, ćwierci itd. algorytm nosi nazwę wyszukiwania binarnego (ang. binary search).

Musimy uściślić sposób podziału zbioru na coraz mniejsze fragmenty. Zbiór Z odwzorowujemy w tablicy Z[ ] składającej się z n elementów ponumerowanych kolejno od 0 do n-1. Określmy dwa indeksy:

ip - indeks pierwszego elementu zbioru
ik - indeks końcowego elementu zbioru

indeksy elementów tablicy Z[ ]	0	1	2	...	n-2	n-1
indeksy początku i końca	ip					ik

Indeksy ip i ik określają obszar zbioru w tablicy Z[ ]. Początkowo będą oczywiście równe: ip = 0 oraz ik = n - 1.

Na podstawie tych dwóch indeksów obliczamy indeks elementu środkowego isr:

$$isr = \frac{ip + ik}{2}$$

Jeśli zbiór Z jest uporządkowany rosnąco, to:

Z[0]	Z[1]	...	Z[isr-1]	Z[isr]	Z[isr+1]	...	Z[n-2]	Z[n-1]

ip ← isr → ik
elementy ≤ Z[isr] elementy ≥ Z[isr]

Zatem w przypadku, gdy k < Z[isr], to przechodzimy do pierwszej połówki, w której indeksy przebiegają od ip do isr - 1. Element Z[isr] pomijamy, gdyż wiadomo, że o niego nam nie chodzi.

Jeśli k > Z[isr], to przechodzimy do drugiej połówki o indeksach od isr + 1 do ik.

Algorytm wyszukiwania binarnego

Wejście

ip - indeks pierwszego elementu w tablicy Z[ ], ip ∈C
ik - indeks ostatniego elementu w tablicy Z[ ], ik ∈C
Z[ ] - tablica do wyszukania elementu. Indeksy od ip do ik
k - wartość poszukiwanego elementu - tzw. klucz

Wyjście:

p = indeks elementu o kluczu k lub
p = -1, jeśli nie odnalezienia elementu o takim kluczu.

Zmienne pomocnicze

isr - indeks elementu środkowego. isr∈C

Lista kroków:

K01: p ← -1 ; zakładamy, iż k nie występuje w zbiorze
K02: Dopóki ip ≤ ik wykonuj K02...K07 ; w pętli poszukujemy elementu o wartości k
K03:
isr = ip + ik
2
; wyznaczamy element środkowy
K04: Jeśli k ≠ Z[isr], to idź do K07
K05: p ← isr ; element znaleziony, kończymy
K06: Idź do K11
K07: Jeśli k < Z[isr], to idź do K10 ; wybieramy odpowiednią połówkę zbioru na dalsze poszukiwania
K08: ip ← isr + 1 ; k > Z[isr] → druga połówka
K09: Następny obieg pętli K02
K10: ik ← isr - 1 ; k < Z[isr] → pierwsza połówka
K11: Zakończ zwracając p