Iteracyjna i rekurencyjna implementacja wybranych algorytmów – porównanie czasów działania

Liczby Fibonacciego, Algorytm Quicksort, Wieże Hanoi

Wiktor Kalaga

8 czerwca 2024

Plan prezentacji

Wprowadzenie

Przypomnienie 1/2

Algorytm jest pewną ściśle określoną procedurą obliczeniową, która dla właściwych danych wejściowych *produkuje* żądane dane wyjściowe, które stanowią wynik działania algorytmu. W związku z tym algorytm to ciąg kroków obliczeniowych przekształcających dane wejściowe w dane wyjściowe.

Algorytm powinien być:

- deterministyczny,
- skończony,
- poprawny,
- adaptacyjny,
- efektywny.

Przypomnienie 2/2

Struktura danych to środek służący do przechowywania i organizowania danych w celu ułatwienia dostępu do nich oraz ich modyfikacji.

Algorytmy iteracyjne

Wyjaśnienie

Algorytm iteracyjny składa się z instrukcji iteracyjnej, dzięki której pewne operacje są wykonywane określoną liczbę razy lub do osiągnięcia pewnego warunku. Zwykle określane są jako pętle.

Przypomnienie 1/2

Instrukcje iteracyjne można podzielić ze względu na sposób kontrolowania iteracji:

- warunkowe,
- licznikowe,
- kombinacja powyższych.

Przypomnienie 2/2

Typy instrukcji iteracyjnych:

- pętla *for*,
- pętla *while*,
- pętla *do..while*.

Przykład

```
// Instrukcja [for]
int sumFor = 0;
for (int i = 1; i <= 100; i++)
{
    sumFor += i;
}

// Instrukcja [while]
int sumWhile = 0;
int j = 1;
while (j <= 100)
{
    sumWhile += j;
    j++;
}

// Instrukcja [do..while]
int sumDoWhile = 0;
int k = 1;
do
{
    sumDoWhile += k;
    k++;
} while (k <= 100);
```

Rys.: Typy instrukcji iteracyjnych.

Algorytmy rekurencyjne

Wyjaśnienie

Jest to rodzaj algorytmów, stosujących **rekurencję**, co oznacza funkcję lub procedurę której definicja odwołuje się do samej siebie. Podejście rekurencyjne jest charakterystyczne dla algorytmów projektowanych metodą *dziel i zwyciężaj* [s 19]. W programowaniu funkcyjnym zamiast instrukcji iteracyjnych (pętli) stosuje się właśnie rekurencję.

Zalety

- treść funkcji zgodna z definicją matematyczną,
- rozkład problemu na mniejsze podproblemy.

Wady

- duża złożoność obliczeniowa, przepełnienie stosu,
- brak optymalizacji = słaba wydajność.

Rekurencja bezpośrednia i pośrednia

Rekurencja bezpośrednia ma miejsce w sytuacji, kiedy w ciele funkcji (lub procedury) dochodzi do wywołania samej siebie.

W przypadku **rekurencji pośredniej** dochodzi do łańcucha wywołań, co oznacza wywoływanie funkcji w ciele innej funkcji i na odwrót.

Przykład

```
1 reference
public static int Factorial(int n)
{
    if (n == 0)
        return 1;
    else
        return n * Factorial(n - 1);
}
```

```
1 reference
public static void FunctionA(int n)
{
    if (n > 0)
    {
        Console.WriteLine($"Funkcja 'A': {n}. Wywołuję Funkcję 'B'");
        FunctionB(n - 1);
    }
}

1 reference
public static void FunctionB(int n)
{
    if (n > 0)
    {
        Console.WriteLine($"Funkcja 'B': {n}. Wywołuję Funkcję 'A'");
        FunctionA(n - 1);
    }
}
```

Rys.: Przykładowe implementacje rekurencji bezpośredniej oraz pośredniej.

Rekurencja ogonowa i nieogonowa

Ogonowa to taki rodzaj rekurencji, której wywołanie następuje na samym końcu działania funkcji, czyli ostatnią operacją tej funkcji jest wywołanie samej siebie lub zwrócenie końcowego wyniku.

Nieogonowa występuje w sytuacji, kiedy po wywołaniu rekurencyjnym mogą występować inne operacje na danych zwróconych przez poprzednie wywołanie tej funkcji.

Przykład

```
1 reference
public static int FactorialTailRecurssion(int n, int res)
{
    if (n == 0)
        return res;
    else
        return FactorialTailRecurssion(n - 1, n * res);
}
```

```
1 reference
public static int FactorialNonTailRecurssion(int n)
{
    if (n == 0)
        return 1;
    else
        return n * FactorialNonTailRecurssion(n - 1);
}
```

Rys.: Przykładowe implementacje rekurencji ogonowej oraz nieogonowej.

Struktury danych a rekurencja

Rekurencja jest naturalnym narzędziem do przeglądania struktur danych takich jak drzewa i grafy. Algorytmy takie jak **BST** (drzewo przeszukiwań binarnych), **DFS** (algorytm przeszukiwania w głąb) czy **BFS** (algorytm przeszukiwania wszerz) często są implementowane rekurencyjnie.

Liczby Fibonacciego

Liczby Fibonacciego to ciąg liczb naturalnych, z których każda kolejna liczba jest określona rekurencyjnie, więc zależy od poprzednich liczb. Pierwsza liczba ciągu określona jako F_0 i wynosi 0. $F_1 = 1$, z kolei następne wartości ciągu dla liczb od F_2 do F_n jest sumą dwóch poprzednich liczb. Formalny zapis ciągu Fibonacciego jest następujący:

$$F_n = \begin{cases} 0, & \text{dla } n = 0, \\ 1, & \text{dla } n = 1, \\ F_{(n-1)} + F_{(n-2)}, & \text{dla } n > 1. \end{cases}$$

Algorytm wyznaczania n-tego wyrazu ciągu Fibonacciego można zaimplementować zarówno **iteracyjnie** jak i **rekurencyjnie**.

- Implementacja rekurencyjna sprowadza się do przepisania definicji matematycznej, ale jest mocno niewydajna.
- Podejście iteracyjne zakłada wykorzystanie pętli a jego złożoność jest liniowa.

Liczby Fibonacciego - podejście rekurencyjne

```
3 references
static int FibonacciRecursive(int n)
{
    if (n < 3)
        return 1;
    else
        return FibonacciRecursive(n - 1) + FibonacciRecursive(n - 2);
}
```

Rys.: Implementacja rekurencyjna algorytmu wyznaczania n-tego wyrazu ciągu Fibonacciego

Złożoność obliczeniowa: $O(2^n)$

Liczby Fibonacciego - podejście iteracyjne

```
1 reference
static int FibonacciIterative(int n)
{
    if (n <= 1)
        return n;

    int a = 0, b = 1;
    for (int i = 2; i <= n; i++)
    {
        int temp = a + b;
        a = b;
        b = temp;
    }
    return b;
}
```

Rys.: Implementacja iteracyjna algorytmu wyznaczania n-tego wyrazu ciągu Fibonacciego

Złożoność obliczeniowa: $O(n)$

Optymalizacja rekurencyjna

Memoizacja

To technika optymalizacji, polegająca na przechowywaniu wyników już obliczonych wywołań funkcji, aby uniknąć powtarzających się obliczeń.

Algorytm Quicksort

Sortowanie szybkie (*ang.* Quicksort) ostało wynalezione przez angielskiego informatyka, profesora Sir Tony'ego Hoare'a w latach 60-tych ubiegłego wieku.

Rys.: Wynalazca *Quicksort* - sir Charles Antony Richard Hoare

Algorytm sortowania szybkiego jest oparty na technice '*dziel i zwyciężaj*', którą można scharakteryzować w trzech punktach:

1. **Dziel** - podział głównego problemu na podproblemy.
2. **Zwyciężaj** - znalezienie rozwiązania podproblemów.
3. **Połącz** - połączenie rozwiązań podproblemów skutkuje rozwiązaniem nadrzędnego problemu.

Bazując na ww. charakterystyce '*dziel i zwyciężaj*' Quicksort można zdefiniować następująco:

1. **Dziel** - sortowany zbiór jest **dzielony*** na dwie podtablice, tak że każdy element lewej podtablicy jest nie większy niż każdy element prawej podtablicy.
2. **Zwyciężaj** - obie podtablice są sortowane za pomocą rekurencyjnych wywołań algorytmu.
3. **Połącz** - połączenie posortowanych podtablic daje posortowany cały zbiór.

Quicksort - podział tablicy

* - Tworzenie partycji

Podział w algorytmie szybkiego sortowania polega na przestawianiu elementów tablicy w taki sposób, aby utworzyć dwie podtablice względem wybranego elementu rozdzielającego (pivotu). Elementy mniejsze od pivota trafiają na lewo od niego, a większe na prawo. Przestawianie elementów względem pivota odbywa się poprzez iteracyjne porównywanie i zamianę elementów miejscami.

Quicksort - procedura

```
QUICKSORT(A, p, r)
if p < r
  q = PARTITION(A, p, r)
  QUICKSORT(A, p, q - 1)
  QUICKSORT(A, q + 1, r)
```

, gdzie:

- **A** - sortowana tablica,
- **p** - indeks początkowy,
- **r** - indeks końcowy

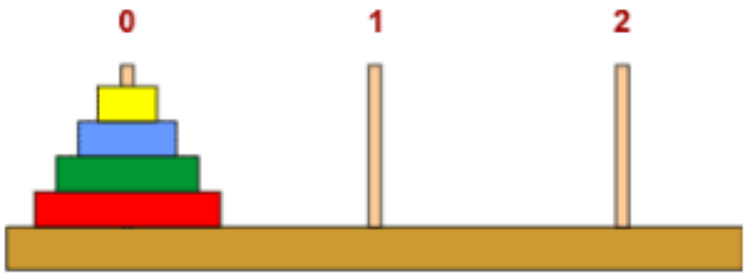
Złożoność obliczeniowa: $O(n \log n)$ **, pes.** $O(n^2)$ (np. dla niesymetrycznego zbioru danych)

Wieże Hanoi

Wieże Hanoi to łamigłówka wymyślona w 1883 roku przez matematyka francuskiego pochodzenia - Edouarda Lucasa. Łamigłówka ta zbudowana jest z deski, na której znajdują się trzy wieże w równych odstępach. Na pierwszej wieży nasunięte są drewniane krążki o coraz mniejszych średnicach.

Wieże Hanoi - cel i zasady

Celem jest przesunięcie wszystkich krążków z pierwszej wieży (oznaczonej indeksem 0) na ostatnią (oznaczonej indeksem 2).



1. Tylko jeden krążek może być przeniesiony na raz.
2. Każdy ruch polega na zdjęciu krążka z wierzchu jednego stosu i umieszczeniu go na wierzchu innego stosu.
3. Krążek nie może być umieszczony na mniejszym krążku.

Wieże Hanoi cd.

W miarę wzrostu ilości krążków do przeniesienia złożoność obliczeniowa rośnie niezwykle szybko. Zarówno algorytm iteracyjny jak i rekurencyjny posiada **złożoność** $O(2^n)$.

Rekurencyjnie - kroki postępowania

Podobnie jak w innych algorytmach rekurencyjnych - dominuje metoda '*dziel i zwyciężaj*'.

1. przenieś (rekurencyjnie) $n-1$ krążków ze słupka 0 na słupek 1 posługując się słupkiem 2,
2. przenieś jeden krążek ze słupka 0 na słupek 2,
3. przenieś (rekurencyjnie) $n-1$ krążków ze słupka 1 na słupek 2 posługując się słupkiem 0

Podsumowanie

- **Algorytmy iteracyjne** - pewne operacje są wykonywane określoną liczbę razy lub do osiągnięcia pewnego warunku. Zastosowanie przy użyciu pętli.
- **Algorytmy rekurencyjne** - funkcje lub procedury, których definicje odwołują się do samej siebie. Rodzaje: bezpośrednia i pośrednia, ogonowa i nieogonowa.
- **Liczby Fibonacciego** - to ciąg liczb naturalnych, z których każda kolejna liczba jest określona rekurencyjnie.
- **Quicksort** - algorytm szybkiego sortowania jest oparty na technice '*dziel i zwyciężaj*'. Dzielimy w nim tablice na dwie mniejsze podtablice.
- **Wieże Hanoi** - gra polegająca na przeniesieniu krążków z pierwszej wieży na ostatnią z zachowaniem pewnych warunków.

Quiz - Pytanie 1

Co charakteryzuje algorytm iteracyjny?

A. Wykonuje operacje w sposób rekurencyjny, bez określonej liczby powtórzeń.

B. Wykonuje operacje w określonej liczbie kroków lub do spełnienia pewnego warunku, używając pętli.

C. Wykonuje operacje jednorazowo, bez powtórzeń.

D. Wykonuje operacje tylko na danych wejściowych bez warunków iteracyjnych.

Quiz - Odpowiedź 1

Co charakteryzuje algorytm iteracyjny?

A. Wykonuje operacje w sposób rekurencyjny, bez określonej liczby powtórzeń.

B. Wykonuje operacje w określonej liczbie kroków lub do spełnienia pewnego warunku, używając pętli.

C. Wykonuje operacje jednorazowo, bez powtórzeń.

D. Wykonuje operacje tylko na danych wejściowych bez warunków iteracyjnych.

Quiz - Pytanie 2

Jakie podejście jest często używane w algorytmach projektowanych metodą dziel i zwyciężaj oraz w programowaniu funkcyjnym zamiast instrukcji iteracyjnych?

A. Algorytmy genetyczne
B. Programowanie dynamiczne
C. Rekurencja
D. Algorytmy zachłanne

Quiz - Odpowiedź 2

Jakie podejście jest często używane w algorytmach projektowanych metodą dziel i zwyciężaj oraz w programowaniu funkcyjnym zamiast instrukcji iteracyjnych?

A. Algorytmy genetyczne
B. Programowanie dynamiczne
C. Rekurencja
D. Algorytmy zachłanne

Quiz - Pytanie 3

Jaka jest złożoność obliczeniowa algorytmu iteracyjnego pozwalającego na wyznaczenie n-tego wyrazu ciągu Fibonacciego ?

A. liniowa - $O(n)$
B. logarytmiczna - $O(logn)$
C. kwadratowa - $O(n^2)$
D. liniowo logarytmiczna - $O(n\,logn)$

Quiz - Odpowiedź 3

Jaka jest złożoność obliczeniowa algorytmu iteracyjnego pozwalającego na wyznaczenie n-tego wyrazu ciągu Fibonacciego ?

A. liniowa - $O(n)$
B. logarytmiczna - $O(logn)$
C. kwadratowa - $O(n^2)$
D. liniowo logarytmiczna - $O(n\,logn)$

Quiz - Pytanie 4

Na czym polega etap podziału w algorytmie szybkiego sortowania ?

A. Na znalezieniu najmniejszego elementu w nieposortowanej części tablicy i zamianie go z pierwszym elementem tej części.

B. Na podzieleniu tablicy na dwie równe części i sortowaniu każdej części osobno.

C. Na rekursywnym łączeniu mniejszych posortowanych fragmentów w jeden większy posortowany fragment.

D. Na zamianie miejscami elementów, tak aby wszystkie elementy mniejsze od wybranego elementu (pivot) znalazły się po jego lewej stronie, a większe po prawej stronie.

Quiz - Odpowiedź 4

Na czym polega etap podziału w algorytmie szybkiego sortowania ?

A. Na znalezieniu najmniejszego elementu w nieposortowanej części tablicy i zamianie go z pierwszym elementem tej części.

B. Na podzieleniu tablicy na dwie równe części i sortowaniu każdej części osobno.

C. Na rekursywnym łączeniu mniejszych posortowanych fragmentów w jeden większy posortowany fragment.

D. Na zamianie miejscami elementów, tak aby wszystkie elementy mniejsze od wybranego elementu (pivot) znalazły się po jego lewej stronie, a większe po prawej stronie.

Quiz - Pytanie 5

Zakładając, że podczas rozwiązywania zagadki Wież Hanoi przenoszenie krążka zajmuje 1s, to po jakim czasie zagadka zostanie rozwiązana, kiedy do dyspozycji mamy 10 krążków ?

A. W następnym stuleciu.
B. Po 10 sekundach.
C. Po około 17 minutach.
D. Po 2047 sekundach.

Quiz - Odpowiedź 5

Zakładając, że podczas rozwiązywania zagadki Wież Hanoi przenoszenie krążka zajmuje 1s, to po jakim czasie zagadka zostanie rozwiązana, kiedy do dyspozycji mamy 10 krążków ?

A. W następnym stuleciu.
B. Po 10 sekundach.
C. Po około 17 minutach.
D. Po 2047 sekundach.

Bibliografia

- Wykłady z przedmiotu Koncepcje Języków Programowania,
- Wprowadzenie do algorytmów, PWN,
- Codenga: Co to jest rekurencja ?,
- Korepetycje z informatyki: Ciąg Fibonacciego,
- Memoizacja, czyli optymalizacja na szybko,
- YouTube: Algorytmy - Quicksort - Sortowanie szybkie,
- Algorytmy Sortujące - Sortowanie szybkie,
- Wieże Hanoi