Grupa lab7

Inżynieria i analiza danych

Politechnika Rzeszowska

Algorytmy I Struktury Danych

Prowadzący: dr inż. Mariusz Borkowski prof. PRz

Sprawozdanie z projektu algorytmu

Autor:

Łucja Telesz nr albumu 166708

Spis treści

[1. Wstęp 3](#_Toc57847175)

[2. Opis problemu 3](#_Toc57847176)

[3. Opis szczegółów implementacji problemu 3](#_Toc57847177)

[4. Rezultaty testów wykonanych przy pomocy programu 3](#_Toc57847178)

[5. Schemat blokowy algorytmu 4](#_Toc57847179)

[6. Pseudokod algorytmu 5](#_Toc57847182)

[7. Wnioski i podsumowanie 5](#_Toc57847183)

[8. Kod programu (algorytmu) 5](#_Toc57847184)

# Wstęp

Algorytm jest to przepis prowadzący do rozwiązania określonego zadania. W ramach zajęć należało przygotować zadanie projektowe w języku programowania C++ w środowisku Code::Blocks IDE. Celem projektu było zaimplementowanie algorytmu.

1. Celem projektu było zaimplementowanie trzech algorytmów sortujących elementy tablicy:
2. sortowania przez scalanie (mergesort), sortowania szybkiego (quicksort) oraz sortowania
3. introspektywnego (introsort). Zastosowanie szablonów (templates) pozwala na sortowanie
4. elementów dowolnego typu. W celu przetestowania poprawności działania program pozwala także
5. generować tablice o określonym parametrami rozmiarze i stopniu posortowania. Po każdym
6. sortowaniu sprawdzana jest poprawność kolejności elementów.
8. Każdy algorytm przetestowano dla następujących tablic o rozmiarze kolejno 10 000, 50 000,
9. 100 000, 500 000, 1 000 000 elementów
10. Celem projektu było zaimplementowanie trzech algorytmów sortujących elementy tablicy:
11. sortowania przez scalanie (mergesort), sortowania szybkiego (quicksort) oraz sortowania
12. introspektywnego (introsort). Zastosowanie szablonów (templates) pozwala na sortowanie
13. elementów dowolnego typu. W celu przetestowania poprawności działania program pozwala także
14. generować tablice o określonym parametrami rozmiarze i stopniu posortowania. Po każdym
15. sortowaniu sprawdzana jest poprawność kolejności elementów.
17. Każdy algorytm przetestowano dla następujących tablic o rozmiarze kolejno 10 000, 50 000,
18. 100 000, 500 000, 1 000 000 elementów
19. Celem projektu było zaimplementowanie trzech algorytmów sortujących elementy tablicy:
20. sortowania przez scalanie (mergesort), sortowania szybkiego (quicksort) oraz sortowania
21. introspektywnego (introsort). Zastosowanie szablonów (templates) pozwala na sortowanie
22. elementów dowolnego typu. W celu przetestowania poprawności działania program pozwala także
23. generować tablice o określonym parametrami rozmiarze i stopniu posortowania. Po każdym
24. sortowaniu sprawdzana jest poprawność kolejności elementów.
26. Każdy algorytm przetestowano dla następujących tablic o rozmiarze kolejno 10 000, 50 000,
27. 100 000, 500 000, 1 000 000 elementów

# Opis problemu

Zadany algorytm miał na celu, dla tablicy A składającej się z liczb całkowitych, stworzenie nowej tablicy B, która będzie tej samej wielkości. Poszczególne elementy tablicy B miały być równe liczbie elementów tablicy A, które znajdowały się na prawo elementu tablicy A o tym samym indeksie, co obecny indeks tablicy B i były od niego większe od tego elementu.

Przykład:

n=6

Wejście: A[n] = [ 1, 2, 3, 4, 5, 6 ]

Wyjście: B[n] = [ 5, 4, 3, 2, 1, 0 ]

# Opis szczegółów implementacji problemu

W programie została zaimplementowana funkcja o nazwie „check”, która jako argumenty przyjmuje tablicę A oraz - jako drugi argument - jej rozmiar. Następnie tworzy tablicę B o tym samym rozmiarze, co tablica A (rozmiar został przekazany jako argument funkcji). Po tym wykonuje się pętla dla każdego elementu tablicy A. W pętli wartość zmiennej „ilosc\_wiekszych” jest ustawiana na 0. W pętli została zainicjowana kolejna pętla, która wykonuje się dla każdego elementu po elemencie, dla którego wykonuje się pierwsza pętla. W tej pętli sprawdzane zostaje, czy wartość danego kolejnego elementu tablicy A jest większa od tego, dla którego wykonuje się pierwsza pętla. Po zakończeniu drugiej pętli do tablicy B dla indeksu, dla którego jest wykonywana pierwsza pętla, zostaje wpisana wartość zmiennej „ilosc\_wiekszych”. Po wykonaniu pętli, funkcja zwraca tablicę B.

# Rezultaty testów wykonanych przy pomocy programu

Algorytm przetestowano dla następujących tablic o rozmiarze kolejno: 10, 100 i 1000 elementów uprzednio zainicjowanych w programie. Wynikami testów są tablice, które są sprawdzane przez program pod kątem wartości - jeśli wartości się zgadzają, testy zostają zakończone pozytywnie oraz program może się wykonywać dalej.

# Schemat blokowy algorytmu

**Dane:** Liczba całkowita: n > 0 (szerokość tablicy).

Tablica tabA[0…n] wypełniona wartościami wejściowymi.

**Wyniki:** Tablica tabB[0…n] zawierająca wartości wynikowe algorytmu.

Wypisz tabA[0…n]

Wypisz tabB[0…n]

tabA[j] > tabA[i]

j<n

I<

j++

j++

ilosc\_wiekszych = 0

j=i

tabB[i]=ilosc\_wiekszych

i++

tabB[0…n]

i=0

START

NIE

NIE

NIE

STOP

TAK

T

ilosc\_wiekszych++

TAK

T

TAK

T

i<n

I<

Wczytaj n

Wczytaj tabA[0…n]

# Pseudokod algorytmu

**Dane:** Liczba całkowita: n > 0 (szerokość tablicy).

Tablica tabA[0…n] wypełniona wartościami wejściowymi.

**Wyniki:** Tablica tabB[0…n] zawierająca wartości wynikowe algorytmu.

**Algorytm:**

tabB[0…n]

**dla** i = 0, 1, 2, …, n **wykonaj**

ilosc\_wiekszych ← 0

**dla** j = i, j+1, …, n **wykonaj**

**jeśli** tabA[j] > tabA[i]

ilosc\_wiekszych ← ilosc\_wiekszych + 1

tabB[i] ← ilosc\_wiekszych

# Wnioski i podsumowanie

Pracując przy projekcie przybliżyłam sobie tematykę działania tablic, odczytu i zapisu danych z pliku, odczytu wierszy i podziału ich na pojedyncze elementy oraz działania wskaźników. Miałam również okazję do przećwiczenia tworzenia schematu blokowego i pseudokodu. Dla czytelności stworzyłam kilka osobnych funkcji, jednak równie dobrze można było stworzyć jedną funkcję główną oraz jedną inicjującą algorytm. Ćwiczenie pozwoliło mi na nabycie wielu nowych umiejętności.

# Kod programu (algorytmu)

int\* check(int\* tab, int size){

int\* new\_tab = new int[size];

for (int i = 0; i < n; i++){

int ilosc\_wiekszych = 0;

for (int j = i; j < n; j++){

if(tab[j]>tab[i]){

ilosc\_wiekszych++;

}

}

new\_tab[i]=ilosc\_wiekszych;

}

return new\_tab;

}