Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Czerwono-Czarni

autor	Aleksander Augustyniak
prowadzący	mgr inż. Marek Kokot
rok akademicki	2019/2020
kierunek	informatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium	piątek, 11:45 – 13:15
sekcja	22
termin oddania sprawozdania	2020-01-27

1 Treść zadania

Napisać program sortujący liczby rzeczywiste w pewnym zbiorze. Liczby podawane są w dość specyficzny sposób. Liczba może być dodana lub usunięta ze zbioru. Dodanie liczb jest realizowane przez komendę `add`, po której może wystąpić jedna lub więcej liczb (rozdzielonych białymi znakami). Komenda `remove` usuwa podaną po niej liczbę (lub liczby rozdzielone białymi znakami) ze zbioru. Komenda `print` powoduje wypisane liczb zawartych w zbiorze w porządku rosnącym. Po komendzie tej można podać nazwę pliku, wtedy wartości zostaną zapisane do tegoż pliku zamiast na standardowe wyjście. Komenda `graph` wypisuje drzewo w postaci graficznej – głębsze poziomy drzewa są wypisywane z coraz większym wcięciem. Dodatkowo wartości w węzłach czarnych są wypisywane w nawiasach kwadratowych, np. `[13]`, w węzłach czerwonych – w nawiasach okrągłych, np. `(13)`. Podobnie jak w przypadku komendy print po komendzie graph można podać nazwę pliku do zapisu. Jeżeli komendy `print` i `graph` są użyte do zapisu do pliku, możliwe jest poprzedzenie nazwy pliku znakiem `+`. Wtedy plik nie zostanie nadpisany, ale nowa treść zostanie dopisana na końcu pliku, nie niszcząc dotychczasowej jego zawartości. Znak `%` rozpoczyna komentarz do końca linii. Każda komenda jest zapisana w osobnej linii. Jeżeli zostanie podana niepoprawna komenda, program ignoruje ją.

Przykładowy plik wejściowy:

```
% przykladowy plik wejsciowy
add -3.14 43.9 4
graph % wypisane z wcieciami i z oznaczeniami kolorow wezlow
remove 4
print % wypisane na ekran
add 3.45 -0.32
print test-1.txt % wypisanie do pliku

add 490 32.3

print % na ekran
print + test-1.txt % dopisanie do pliku
```

Po komendzie `graph` zostanie wypisane drzewo:

```
   (-3.14)
[4]
   (42.9)
```

Program uruchamiany jest z linii poleceń z wykorzystaniem jednego prze-

łącznika:

`-i` plik wejściowy

Do przechowywania liczb należy wykorzystać drzewa czerwono-czarne. Jest to warunek sine qua non.

2 Analiza zadania

Zadanie skupia się wokół problemu sortowania liczb rzeczywistych, zapisanych w pliku o podanej przez użytkownika nazwie. Struktura danych powinna wykorzystywać dynamiczne alokowanie pamięci, w celu szybszego wykonywania operacji na danych. Jednym z warunków przystąpienia do napisania programu jest umiejętność korzystania z pamięci w ten sposób, aby nie powodować wycieku danych.

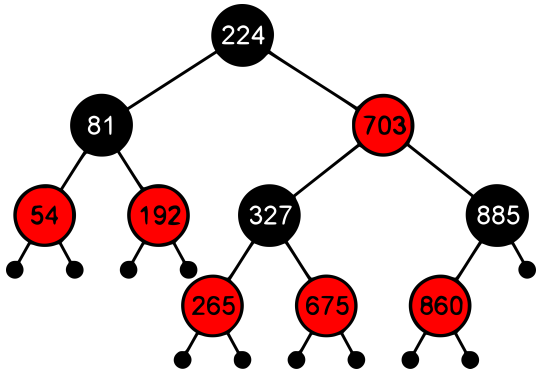
Do zrealizowania zadania potrzebne jest zarezerwowanie jednego bitu pamięci dla każdego z węzłów zawartych w strukturze danych, w celu opisania koloru danego węzła.

2.1 Struktury danych

Program wykorzystuje drzewo czerwono–czarne do przechowywania wartości. Wartość jest przypisywana węzłowi o odpowiednim kolorze. Węzeł może mieć jednego, dwóch potomków lub ich nie mieć. W lewym poddrzewie wybranego węzła występują wartości mniejsze od wartości przechowywanej przez *węzeł–rodzic*, zaś wartości niemniejsze są umieszczone w prawym poddrzewie węzła. Rysunek 1 przedstawia wizualizację drzewa czerwono–czarnego.

Podstawowe zasady panujące w drzewie czerwono–czarnym: [1]

1. Każdy z węzłów jest czerwony lub czarny.
2. Korzeń drzewa jest czarny.
3. Wszystkie liście (NIL) są czarne.
4. Jeżeli węzeł jest czerwony, to oba jego potomki są czarne.
5. Każda ścieżka od wybranego węzła do liścia zawiera tę samą ilość czarnych węzłów.



Rysunek 1: Wizualizacja drzewa czerwono–czarnego przechowującego liczby rzeczywiste. Węzły reprezentują wartości wprowadzone w następującej kolejności: 327, 54, 224, 81, 703, 885, 192, 860, 265, 675.

Powyższe zasady – opisujące wzajemny stosunek węzłów drzewa – pomagają zapanować nad wartościami przechowywanymi w węzłach oraz równoważyć jego wysokość. Pozwala to na skrócenie czasu poszukiwania, usuwania oraz dodawania wartości do struktury.

2.2 Algorytmy

Dodawanie wartości do drzewa czerwono–czarnego jest realizowane podobnie, jak w drzewie poszukiwań binarnych, lecz dodatkowo program wykonuje lokalne operacje na węzłach, takie jak: rotacje w lewo/prawo oraz zmiana kolorów węzłów, by przywrócić właściwości drzewa czerwono–czarnego.

Gwarantowana złożoność obliczeniowa dla operacji dodawania, wyszukiwania, usuwania węzłów wynosi średnio, jak i w najgorszym przypadku $O(\log n)$, gdzie n oznacza ilość wszystkich elementów znajdujących się w strukturze danych. [3] Złożoność obliczeniowa algorytmów rotacji w lewo, w prawo oraz zmiany kolorów węzłów również wynosi $O(\log n)$. [2]

By zrealizować operacje wypisywania i usuwania wartości węzłów, program rekurencyjnie przechodzi przez wszystkie węzły drzewa.

3 Specyfikacja zewnętrzna

Aby uruchomić program, należy wprowadzić do linii poleceń, w odpowiedniej kolejności:

```
nazwa_programu -i plik_wejściowy
```

Jeżeli program nie został prawidłowo uruchomiony, pojawi się stosowny ko-

munikat z ewentualną instrukcją prawidłowego uruchomienia programu.

4 Specyfikacja wewnętrzna

Program realizuje pojedyncze komendy usuwając z wczytanego łańcucha znakowego niepotrzebne znaki, np. znaki białe, znak rozpoczęcia komentrza (i znaki znajdujące się bezpośrednio za nim, aż do końca linii).

W trakcie zczytywania pojedynczych wartości program decyduje o tym, czy wczytana wartość jest prawidłowa, tzn. czy nie zawiera znaków nie będących cyframi. Ponadto, jeżeli funkcja zczytująca dane dostanie informację o liczbie, wykraczającej poza zakres określony dla jej typu, liczba ta zostanie zignorowana, a program dalej będzie realizowany.

4.1 Ogólna struktura programu

Funkcja główna sprawdza, czy ilość wprowadzonych parametrów do linii poleceń jest większa od dwóch. Jeżeli nie – program kończy się – w przeciwnym wypadku zostaje uruchomiona funkcja ReadFromFile, przyjmująca jeden plik wejściowy. Funkcja ta zczytuje – linia po linii – każde polecenie i wywołuje odpowiednie podfunkcje: Add, Delete, Print, Graph, itd. – w zależności od znajdującej się w pliku wejściowym komendy.

Program – po wczytaniu wszystkich komend – dealokuje całą zajętą przez drzewo pamięć i zamyka się.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji znajduje się w załączniku.

5 Testowanie

Program został przetestowany na plikach:

- typowych, poprawnych – w których komendy następowały po sobie w różnym porządku;
- nietypowych i poprawnych – o znacznej ilości danych;
- niepoprawnych – wynikiem tego było ignorowanie źle wypisanych komend i liczb zawierających znaki inne niż zarezerwowane dla cyfr.

Przetestowane zostały również funkcje wykorzystujące rekurencyjne przechodzenie przez węzły.

W tabeli 1 przedstawiono wyniki testów sprawdzających zapotrzebowanie pamięci na określoną ilość wprowadzonych danych:

Program został również sprawdzony pod kątem wycieków pamięci.

6 Uzyskane wyniki

ilość liczb dodanych do struktury	pamięć przydzielona procesowi
10000	3 MB
20000	6 MB
30000	9 MB
50000	15 MB
75000	18.4 MB
100000	25.2 MB
250000	60.6 MB
500000	120 MB
8000000	1.9 GB

Tablica 1: Zestawienie przybliżonej ilości zapotrzebowania na pamięć, potrzebnej do dodania do struktury określonej ilości danych.

7 Wnioski

Program wykorzystujący drzewa czerwono–czarne jako strukturę danych nie jest programem łatwym do napisania. Szczególną trudność sprawiło mi obsłużenie sytuacji usuwania wartości z drzewa, które wymagało przywrócenia jego własności. 2.1

W napisaniu projektu pomogło mi dobre nastawienie, solidna, codzienna praca. Przed przystąpieniem do projektu pisałem wiele innych programów, które były zlecone do wykonania przez prowadzącego na laboratorium. Robiłem również dodatkowe zadania, pozwalające uzyskać odpowiednią sprawność w kodowaniu. Cały wysiłek, który przeznaczyłem przez ostatni semestr na programowanie, zaowocowało zdobytą przeze mnie wiedzą, z której będę mógł w odpowiednim momencie skorzystać.

Niestety, nie udało mi się przetestować programu na maszynach z innym systemem operacyjnym niż Windows 10.

Literatura

[1] Thomas Cormen, Charles Leiserson, Ronalds Rivest, Clifford Stein. *„13". Introduction to Algorithms (3rd ed.).* 2009.

[2] John Morris. „red–black trees". data structures and algorithms., 1998.

[3] James Paton. Red-black trees.

Dodatek
Szczegółowy opis typów i funkcji