

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Звіт

про виконання лабораторної роботи №7 (2-3) з дисципліни «Алгоритми та структури даних»

Виконав:

студент I курсу, групи КА-07

Москаленко Максим Геннадійович

Прийняв:

Київ — 2020

|  |  |
| --- | --- |
| МЕТА ТА ВАРІАНТ | 1 |
| ЗАДАЧА | 2 |
| ВИСНОВОК | 5 |

МЕТА

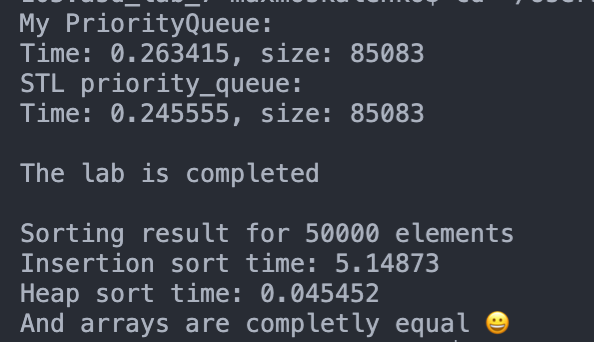
Ознайомитись і дослідити структуру даних бінарна купа. Набути навичок реалізації абстрактної структури даних черга з пріоритетом на основі бінарної купи мовою програмування С++. Ознайомитись з механізмом перевантаження операторів та дослідити особливості сортування купою (пірамідального сортування). Порівняти власну реалізацію пріоритетної черги з готовим бібліотечним рішенням STL.

ВАРІАНТ 18

ЗАДАЧА

1. Створити структуру для зберігання об’єктів з характеристиками відповідно до обраної задачі:
   1. Обрати певний власний критерій для визначення більш пріоритетного об’єкту на основі його характеристик
   2. Перевантажити оператор “менше” (operator<) для порівняння двох об’єктів за критерієм
   3. Перевантажити оператор “дорівнює” (operator==) для визначення рівності двох об’єктів за цим самим критерієм
2. Реалізувати пріоритетну чергу на основі незростаючої бінарної купи:
   1. Створити структуру PriorityQueue, яка буде містити в собі бінарне дерево для роботи бінарної купи. Для зберігання бінарного дерева можна використати звичайний динамічний масив, що вже був реалізований у минулих роботах.
   2. Реалізувати основні методи переміщення по бінарному дереву:
      1. getParent(index) знайти індекс батьківського вузла
      2. getLeftChild(index) знайти індекс лівого дочірнього вузла
      3. getRightChild(index) знайти індекс правого дочірнього вузла
   3. Реалізувати внутрішні методи для підтримки властивості незростаючої (max-heap) купи:
      1. siftUp(index) просіяти елемент вверх по дереву
      2. siftDown(index) просіяти елемент вниз по дереву
   4. Реалізувати основні методи для роботи пріоритетної черги:
      1. push(object) додати новий елемент в чергу
      2. top() отримати верхній елемент з черги
      3. pop() видалити верхній елемент з черги
      4. size() знайти кількість елементів в черзі
      5. empty() перевірити чергу на пустоту
3. Провести тестування, використавши вказану нижче функцію testPriorityQueue(). Перевірити правильність та швидкість роботи, порівнявши з готовим бібліотечним рішенням STL priority\_queue.
4. Продумати реалізацію сортування купою (пірамідального сортування) на базі бінарної купи, вміти обгрунтовано пояснити та показувати в коді.
5. \* Реалізувати сортування купою, порівняти його з готовим бібліотечним рішенням STL std::sort() або з іншими алгоритмами сортувань, реалізованими в минулих роботах, – провести тестування з замірами часу на різних вхідних даних, зробити аналіз та висновки.

Результат виконання програми:

  
Код:

|  |
| --- |
| Весь код у доданому до файлу архіві:  main.cpp - основна задача |

ВИСНОВКИ

У результаті лабораторної роботи було створено та реалізовано структуру даних PriorityQueue на основі бінарної купи. Для динамічного розширення масиву, який представляє собою бінарну купу було використано раніше створену структуру DynamicArray. Час виконання завдання в цілому співпадає з бібліотечною реалізацією. Метод сортування купою було порівняно з методом insertion sort, як і очікувалось (з асимптотичної складності двох алгоритмів) сортування купою виявилось набагато швидшим (у 100 разів для 50 000 елементів), хоча для малої кількості (менше 100) сортування вставкою показало себе краще.