МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №5
по дисциплине «Алгоритмы и структуры данных»
Тема: Слабая куча

Студент(ка) гр. 9382 _____ Голубева В.П.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Разобраться в структуре слабой кучи и научиться работать с ней.

Задание.

31. Дан массив пар типа «число — бит». Предполагая, что этот массив представляет слабую кучу, вывести её на экран в наглядном виде.

Основные теоретические положения.

Обычная куча — дерево, в котором любой родитель больше (или равен) чем любой из его потомков. В слабой куче это требование ослаблено — любой родитель больше (или равен) любого потомка только из своего правого поддерева. В левом поддереве потомки могут быть и меньше и больше родителя.

Также нужен дополнительный битовый массив (назовём его BIT), в котором для **i**-го элемента отмечено, был ли обмен местами между его левым и правым поддеревьями. Если значение для элемента равно 0, то значит обмена не было. Если значение равно 1, значит, левый и правый потомок идут в обратном порядке. А формулы при этом вот такие:

Левый потомок: 2×i+BIT[i]

Правый потомок: 2 × i + 1 − BIT[i]

Функции и структуры данных.

typedef struct couple{

int data;

int bit;

} couple; - структура для записи элементов массива типа число-бит

double log(int a, int b) - вычисляет логарифм от b по основанию a

Для работы со слабой кучей был реализован класс WeakHeap.

couple *heap — указывает на массив, где хранятся элементы кучи

int frnt=0, rear=0 — флажки, указывают на начало и конец кучи

int queue_size — размер кучи

WeakHeap (int) - конструктор, создаёт новую слабую кучу, по количеству элементов в ней

~WeakHeap() - очищает память в кучу

void push (couple f) — добавляет элемент f в кучу

void pop() - удаляет элемент из кучи

void print_heap() - печатает кучу на экран

Описание алгоритма.

Сначала из файла считывался массив, который из условия представлял собой слабую кучу. Была создана структура couple, которая содержит пары число — бит. Причём, хоть биты требуются и не для всех чисел, потому что некоторые из них не имеют потомков, но для корректной работы программы нужно к каждому числу без бита добавить «фиктивный» бит, равный -1. По формулам формируем новый массив, в котором элементы кучи идут в правильном порядке, т.е левый потомок, затем правый.

Затем выводим их, предварительно вычисляя, какое количество чисел будет на текущей строке.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	4 71 -1 43 1 30 -1 11 -1	71 43 11 30	
2.	16 97 -1 83 0	97 83 52 76	

	52 0 76 1 89 1 33 1 73 0 78 0 66 -1 32 -1 20 -1 48 -1 67 -1 58 -1 45 -1 13 -1	89 33 78 73 32 66 48 20 67 58 45 13	
3.	8 69 -1 58 0 95 0 42 0 20 -1 90 -1 11 -1 30 -1	69 58 95 42 20 90 11 30	
4.	2 89 -1 23 -1	89 23	
5.	1 10 -1	10	

Выводы.

Была изучена структура слабой кучи, получены навыки работы с ней.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab5_1.cpp

```
#include "Weak_heap.h"


int main() {
    int count;

    ifstream file("test.txt");//открываем файл для чтения
    file>>count;

       int  count_of_bit=count/2;//вычисляем  количество  элементов,  в
которых введён не фиктивный дополнительный бит
    couple* array=new couple[count];

    for (int i=0;i<count;i++){
        file>>array[i].data>>array[i].bit;//читаем данные из файла
    }

    file.close();

    WeakHeap heap1(count);
    heap1.push(array[0]);
    heap1.push(array[1]);
      for  (int  i=1;i<count_of_bit;i++){//записываем элементы в кучу в
порядке, удобном для вывода на экран
        heap1.push(array[2*i+array[i].bit]);
        heap1.push(array[2*i+1-array[i].bit]);

    }

    heap1.print_heap();//выводим нашу слабую кучу на экран


    delete [] array;
    return 0;
}
```

Название файла: Weak_heap.h

```
#ifndef WEAK_HEAP_H
#define WEAK_HEAP_H

#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cmath>
using namespace std;

//вычисляет логарифм от b по основанию a
double log(int a, int b)
{
```

```
    return log(b) / log(a);
}
//структура для записи элементов массива типа число-бит
typedef struct couple{
    int data;
    int bit;

} couple;

//класс слабой кучи
class WeakHeap {
    couple *heap;
    int frnt=0, rear=0;
    int queue_size;
  public :

    WeakHeap (int) ;
    ~WeakHeap();
    void push (couple f) ;
    void pop();
    void print_heap();

} ;



//Конструктор
WeakHeap::WeakHeap(int size) {
    queue_size=size;
    heap = new couple[queue_size];
}
WeakHeap::~WeakHeap(){
    delete[] heap;
}
void WeakHeap::print_heap(){
    cout<<heap[0].data;
    cout<<"\n";
    int depth=(int)log(2, queue_size);//вычисляем глубину дерева
    int k=0;
    for (int i=0;i<depth;i++){

        for (int j=0;j<pow(2,i);j++){

            cout<<heap[k+1].data<<"  ";
            k++;
        }
        cout<<"\n";
    }

}

//Помещение элемента в куча
void WeakHeap::push (couple f) {

        heap[rear].data=f.data;
        heap[rear].bit=f.bit;
```

```
        rear++;
}
// Извлечение элемента из кучи
void WeakHeap::pop() {
    if ( frnt == rear ) {
        cout << "куча пуста" <<endl ;
        return;
    }
    frnt++;

    }

#endif
```

Название файла: test.txt

4

71 -1

43 1

30 -1

11 -1