МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра информационных систем

Отчет по дисциплине Основы алгоритмизации и программирование

Домашняя работа №14

Выполнил студент гр. ИСЭбд-11

Гапон Николай Иванович

Преподаватель

Власенко Олег Федосович

ССЫЛКА НА GIT https://github.com/NikGapon/1year/tree/main/lab14

ДОМАШНЯЯ РАБОТА

1. Доделать все задачи из классной работы. (задачи 13 - 16)

//#define \_CRT\_SECURE\_NO\_WARNINGS

//

//#include <stdio.h>

//#include <string.h>

//#include <ctype.h>

//#include <time.h>

//#include <stdlib.h>

//

//#define MAX\_LEN 25

//

//

//struct NodeTree {

// char \* word;

// struct NodeTree \* left;

// struct NodeTree \* right;

//};

//struct Dictionary {

// struct NodeTree \* root;

// int cnt\_words;

//};

//

//struct Dictionary \* create();

//void destroy(struct Dictionary \* dict);

//void addWord(struct Dictionary \* dict, char \* word);

//int contains(struct Dictionary \* dict, char \* word);

//

//

//struct Dictionary \* create()

//{

// struct Dictionary \* dict = (struct Dictionary \*)

// malloc(sizeof(struct Dictionary));

// dict->root = NULL;

// dict->cnt\_words = 0;

// return dict;

//}

//

//struct NodeTree \* addElement(

// struct NodeTree \*p,

// char\* word)

//{

//

//

// int cond;

//

// if (p == NULL) {

// p = (struct NodeTree\*)malloc(

// sizeof(struct NodeTree));

// p->word = (char \*)calloc(strlen(word) + 1,

// sizeof(char));

// strcpy(p->word, word);

// p->left = p->right = NULL;

// }

// else if ((cond = strcmp(word, p->word)) == 0) {

// // âñòàâëÿåìîå ñëîâî ñîâïàäàåò

// // ñ óæå èìåþùèìñÿ - íè÷åãî íå äåëàåì

// }

// else if (cond < 0) {

// // âñòàâëÿåìîå ñëîâî ìåíüøå

// // êîðíÿ ïîääåðåâà

// p->left = addElement(p->left, word);

// }

// else {

// // âñòàâëÿåìîå ñëîâî áîëüøå

// // êîðíÿ ïîääåðåâà

// p->right = addElement(p->right, word);

// }

// return p;

//}

//

//

//void addWord(struct Dictionary \* dict, char \* word)

//{

// static int n = 0;

// n++;

// if (n %100 == 0)

// printf("%d\n", n);

//

//

// dict->root = addElement(dict->root, word);

// dict->cnt\_words++;

//}

//

//void clearTree(struct NodeTree \*p)

//{

// if (p != NULL) {

// clearTree(p->left);

// clearTree(p->right);

// free(p->word);

// free(p);

// }

//}

//

//void destroy(struct Dictionary \* dict) {

// clearTree(dict->root);

// free(dict);

//}

//

//int containElement(struct NodeTree \* p, char \*word)

//{

// int cond;

// if (p == NULL) {

// return 0;

// }

// else if ((cond = strcmp(word, p->word)) == 0) {

// return 1;

// }

// else if (cond < 0) {

// return containElement(p->left, word);

// }

// else {

// return containElement(p->right, word);

// }

//}

//

//int contains(struct Dictionary \* dict, char \* word)

//{

// return containElement(dict->root, word);

//}

//

//

//

//

//

//

//

//

//

//

//

//

//

//

//

//

//int loadDictionary(struct Dictionary \* dict,

// const char \* filename) {

// // îòêðûòü ôàéë

// FILE \* fin;

// char s[MAX\_LEN];

//

// fin = fopen(filename, "rt");

// if (fin == NULL)

// {

// return 0;

// }

// // â öèêëå äëÿ âñåõ ñòðîê

// while (!feof(fin)) {

// // çàãðóçèòü ñòðîêó

// if (fgets(s, MAX\_LEN - 1, fin) != NULL) {

// if (s[strlen(s) - 1] == '\n')

// s[strlen(s) - 1] = '\0';

// if (!contains(dict, s)) {

// addWord(dict, s);

// }

// }

// }

// // çàêðûòü ôàéë

// fclose(fin);

// return 1;

//}

//

//

//int convertTextToHtml(

// struct Dictionary \* dict,

// const char \* text\_in\_filename,

// const char \* text\_out\_filename)

//{

// char s[MAX\_LEN];

//

// // îòêðûòü ôàéëû

// FILE \*fin = fopen(text\_in\_filename, "rt");

// if (fin == NULL)

// {

// return 0;

// }

//

// FILE \*fout = fopen(text\_out\_filename, "wt");

// if (fout == NULL)

// {

// fclose(fin);

// return 0;

// }

// fprintf(fout, "<!DOCTYPE html>");

// fprintf(fout, "<html>");

// fprintf(fout, "<head>");

// fprintf(fout, "<meta http - equiv = \"Content-Type\" content = \"text/html; charset=utf-8\" />");

// fprintf(fout, "<title>HTML Document</title>");

// fprintf(fout, "</head>");

// fprintf(fout, "<body>");

//

//

//

// char ch;

// int is\_letter = 0;

// char word[81];

// int word\_len = 0;

//

// while ((ch = getc(fin)) != EOF) {

// if (isalpha((unsigned char)ch)) {

// if (!is\_letter) {

// word\_len = 0;

// }

// is\_letter = 1;

// word[word\_len++] = ch;

// }

// else { // if (!isalpha(ch)) {

//

//

// if (is\_letter) {

// word[word\_len] = '\0';

// if (contains(dict, word))

// fprintf(fout, "<b>%s</b> ", word);

// else

// fprintf(fout, "%s", word);

// }

// is\_letter = 0;

// fprintf(fout, "%c", ch);

// if (ch == '\n')

// fprintf(fout, "<br>");

// }

//

// } // while ((ch = getc(fin)) != EOF)

// fclose(fin);

//

// fprintf(fout, "</body>");

// fprintf(fout, "</html>");

// fclose(fout);

// return 1;

//} // convertTextToHtml- êîíåö!!!

//

//

//

//

//void main() {

// long t0, t1, t2;

// t0 = clock();

// printf("t0 = %f sec \n", t0 / (float)CLOCKS\_PER\_SEC);

//

// struct Dictionary\* dict = create();

// loadDictionary(dict, "dict1.txt");

//

// t1 = clock();

// printf("t1 = %f sec \n", t1 / (float)CLOCKS\_PER\_SEC);

//

// convertTextToHtml(dict,

// "Alice.txt",

// "Alice\_out\_sort\_dict11.html");

//

// destroy(dict);

//

// t2 = clock();

// printf("t2 = %f sec \n",

// t2 / (float)CLOCKS\_PER\_SEC);

//

// printf("Run time = t2 - t0 = %f sec \n",

// (t2 - t0) / (float)CLOCKS\_PER\_SEC);

// ////-------------------------------------------

// t0 = clock();

// printf("t0 = %f sec \n", t0 / (float)CLOCKS\_PER\_SEC);

//

// dict = create();

// loadDictionary(dict, "dict12.txt");

//

// t1 = clock();

// printf("t1 = %f sec \n", t1 / (float)CLOCKS\_PER\_SEC);

//

// convertTextToHtml(dict,

// "Alice.txt",

// "Alice\_out\_sort\_dict12.html");

//

// destroy(dict);

//

// t2 = clock();

// printf("t2 = %f sec \n",

// t2 / (float)CLOCKS\_PER\_SEC);

//

// printf("Run time = t2 - t0 = %f sec \n",

// (t2 - t0) / (float)CLOCKS\_PER\_SEC);

// ////-------------------------------------------

// t0 = clock();

// printf("t0 = %f sec \n", t0 / (float)CLOCKS\_PER\_SEC);

//

// dict = create();

// loadDictionary(dict, "dict2.txt");

//

// t1 = clock();

// printf("t1 = %f sec \n", t1 / (float)CLOCKS\_PER\_SEC);

//

// convertTextToHtml(dict,

// "Alice.txt",

// "Alice\_out\_sort\_dict21.html");

//

// destroy(dict);

//

// t2 = clock();

// printf("t2 = %f sec \n",

// t2 / (float)CLOCKS\_PER\_SEC);

//

// printf("Run time = t2 - t0 = %f sec \n",

// (t2 - t0) / (float)CLOCKS\_PER\_SEC);

// ////-------------------------------------------

// t0 = clock();

// printf("t0 = %f sec \n", t0 / (float)CLOCKS\_PER\_SEC);

//

// dict = create();

// loadDictionary(dict, "dict22.txt");

//

// t1 = clock();

// printf("t1 = %f sec \n", t1 / (float)CLOCKS\_PER\_SEC);

//

// convertTextToHtml(dict,

// "Alice.txt",

// "Alice\_out\_sort\_dict22.html");

//

// destroy(dict);

//

// t2 = clock();

// printf("t2 = %f sec \n",

// t2 / (float)CLOCKS\_PER\_SEC);

//

// printf("Run time = t2 - t0 = %f sec \n",

// (t2 - t0) / (float)CLOCKS\_PER\_SEC);

// //-------------------------------------------

// t0 = clock();

// printf("t0 = %f sec \n", t0 / (float)CLOCKS\_PER\_SEC);

//

// dict = create();

// loadDictionary(dict, "dict31.txt");

//

// t1 = clock();

// printf("t1 = %f sec \n", t1 / (float)CLOCKS\_PER\_SEC);

//

// convertTextToHtml(dict,

// "Alice.txt",

// "Alice\_out\_sort\_dict31.html");

//

// destroy(dict);

//

// t2 = clock();

// printf("t2 = %f sec \n",

// t2 / (float)CLOCKS\_PER\_SEC);

//

// printf("Run time = t2 - t0 = %f sec \n",

// (t2 - t0) / (float)CLOCKS\_PER\_SEC);

// //-------------------------------------------

// t0 = clock();

// printf("t0 = %f sec \n", t0 / (float)CLOCKS\_PER\_SEC);

//

// dict = create();

// loadDictionary(dict, "dict32.txt");

//

// t1 = clock();

// printf("t1 = %f sec \n", t1 / (float)CLOCKS\_PER\_SEC);

//

// convertTextToHtml(dict,

// "Alice.txt",

// "Alice\_out\_sort\_dict32.html");

//

// destroy(dict);

//

// t2 = clock();

// printf("t2 = %f sec \n",

// t2 / (float)CLOCKS\_PER\_SEC);

//

// printf("Run time = t2 - t0 = %f sec \n",

// (t2 - t0) / (float)CLOCKS\_PER\_SEC);

// {

// int x;

// scanf("%d", &x);

// }

//}ВЫВОД

В данной лабораторной работе мы научились измерять время работы алгоритмов, а так же сравнили их время производительности