Московский ордена Ленина, ордена Октябрьской Революции
и ордена Трудового Красного Знамени.
ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н. Э. БАУМАНА

Факультет: Информатики и систем управления

Кафедра: Проектирование и технология производства электронной аппаратуры (ИУ4)

Домашнее задание

«Задача о периодичности совершения рейсов. Программирование на языке С»

По курсу: Системное программирование

Студент: Онорато Д. ИУ4-93
(фамилия, инициалы) (индекс группы)

Юлдашев М.Н. ИУ4-93
(фамилия, инициалы) (индекс группы)

Яшков М.П. ИУ4-93
(фамилия, инициалы) (индекс группы)

Руководитель: Чернов М.М.
(фамилия, инициалы)

Москва
2014

СОДЕРЖАНИЕ

СПИСОК УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СОКРАЩЕНИЙ И ТЕРМИНОВ

АПК – Аппаратно-программный комплекс
АСП – Алгоритм сортировки пузырьком
БСА – Блок-схема алгоритма
ВС – Вычислительная сложность
ГОСТ – Государственный стандарт
КТП – Конструкторско-технологическое проектирование
ОСТ – Отраслевой стандарт
ПО – Программное обеспечение
САПР – Система автоматизированного проектирования
СП – Системное программирование
ТЗ – Техническое задание
УГО – Условное графическое обозначение
ЭВА – Электронная вычислительная аппаратура
ЭВС – Электронно-вычислительное средство

ВВЕДЕНИЕ

Как правило, объектно-ориентированное программирование не сводится сугубо к написанию просто работоспособной программы без учета ее эффективности, быстродействия, сложности и сопутствующих параметров, которые прямо или косвенно влияют на работу с программой и влиют непосредственно на использование заказчиком данного программного продукта.

В настоящее время современная рыночная экономика предъявляет принципиально иные требования к качеству выпускаемой продукции, чем это было на заре производственной эры и поэтому выживаемость любой фирмы, ее устойчивое положение на рынке товаров и услуг определяются уровнем конкурентоспособности и успехом в борьбе за потребителя. В свою очередь конкурентоспособность связана с двумя показателями – уровнем цены и уровнем качества продукции (см. рис. В.1).

Стоит отметить, что последние годы отмечены беспрецедентным ростом внимания к проблеме качества. Мировой опыт показывает, что научно-технический прогресс во многих странах был определен прорывом именно в качестве продукции.

В соответствии с этим, стандарт ISO 8420 дает следующее определение качества:

Качество – совокупность свойств продукции, обусловливающих её пригодность удовлетворять определённые потребности в соответствии с её назначением.

Качество в понимании производителя и качество в понимании потребителя в системе управления качеством взаимосвязаны. Для производителя качество – это авторитет фирмы, увеличение прибыли, рост процветания, поэтому работа по управлению качеством предприятия является важнейшим видом деятельности для всего персонала, от руководителя до конкретного исполнителя.

Рисунок В.1 – Зависимость в рыночной экономике уровня авторитета фирмы от стоимости С и уровня ценности (качества) V

Для того, чтобы учесть необходимые параметры качества создаваемого нами ПО, необходимо комплексно учесть наиболее влияющие факторы на создание и работоспособность нашей программы:

1) Вычислительная сложность;
2) Устойчивость алгоритма;
3) Объем рабочего кода программы;
4) Быстродействие алгоритма

Наиболее полно нас будет интересовать самый первый параметр программы – вычислительная сложность алгоритма программы, так как если плох этот показатель качества работы ПО, то об остальном тем более нет смысла говорить.

Вычислительной сложностью работы программ занимается специальный раздел информатики – теория алгоритмов. Теория алгоритмов — раздел информатики, изучающий общие свойства и закономерности алгоритмов и разнообразные формальные модели их представления. Целью анализа трудоёмкости алгоритмов является нахождение оптимального алгоритма для решения данной задачи. В качестве критерия оптимальности алгоритма выбирается трудоемкость алгоритма, понимаемая как количество элементарных операций, которые необходимо выполнить для решения задачи с помощью данного алгоритма. Функцией трудоемкости называется отношение, связывающие входные данные алгоритма с количеством элементарных операций.

Трудоёмкость алгоритмов по-разному зависит от входных данных. Для некоторых алгоритмов трудоемкость зависит только от объёма данных, для других алгоритмов — от значений данных, в некоторых случаях порядок поступления данных может влиять на трудоемкость. Трудоёмкость многих алгоритмов может в той или иной мере зависеть от всех перечисленных выше факторов.

Два основополагающих понятия теории алгоритмов являются понятия класса сложности и вычислительной сложности.

В теории алгоритмов классами сложности называют множество задач, примерно одинаковых по скорости вычисления. Говоря более узко, каждый класс сложности (в узком смысле) определяется как множество предикатов, обладающих некоторыми свойствами. В качестве ресурсов обычно берутся время вычисления (количество рабочих тактов машины Тьюринга) или рабочая зона (количество использованных ячеек на ленте во время работы). Языки, распознаваемые предикатами из некоторого класса (то есть множества слов, на которых предикат возвращает 1), также называются принадлежащими тому же классу.

Вычисли́тельная сло́жность — понятие в информатике и теории алгоритмов, обозначающее функцию зависимости объёма работы, выполняемой некоторым алгоритмом, от размера входных данных. Раздел, изучающий вычислительную сложность, называется теорией сложности вычислений. Объём работы обычно измеряется абстрактными понятиями времени и пространства, называемыми вычислительными ресурсами. Время определяется количеством элементарных шагов, необходимых для решения задачи, тогда как пространство

определяется объёмом памяти или места на носителе данных. Таким образом, в этой области предпринимается попытка ответить на центральный вопрос разработки алгоритмов: «как изменится время исполнения и объём занятой памяти в зависимости от размера входа?». Здесь под размером входа понимается длина описания данных задачи в битах, а под размером выхода — длина описания решения задачи.

Основные классы сложности применяемые при анализе:

- f(n) = O(1) константа
- f(n) = O(log(n)) логарифмический рост
- f(n) = O(n) линейный рост
- f(n) = O(n*log(n)) квазилинейный рост
- f(n) = O(n^m) полиномиальный рост
- f(n) = O(2^n) экспоненциальный рост

1 ТЕОРЕТИЧЕСКИЙ АНАЛИЗ СЛОЖНОСТИ АЛГОРИТМА ПРОГРАММЫ ОПРЕДЕЛЕНИЯ ПЕРИОДИЧНОСТИ ПОЛЕТА РЕЙСОВ

Реализуемая программа имеет следующие функциональные части: ввод данных, сортировка данных, алгоритм поиска смежных рейсов, вовод результатов. Наиболее ресурсозатратным функциональным блоком является сортировка данных, при этом остальные блоки имеют линейную сложность. Поэтому далее мы будем рассматривать сложность сортировка ввыбором.

1.1Анализ общего случая

На массиве из n элементов имеет время выполнения в худшем, среднем и лучшем случае Θ(n2), предполагая, что сравнения делаются за постоянное время.

Сортировка выбором требует фиксированного числа проходов. На i-ом проходе производится вставка минимального элемента из подсписка A[i+1]–A[n] и вставляется на место A[i]; требуют в среднем i/2 сравнений. Общее число сравнений равно:

$$n + (n-1) + (n-2) + (n-3) + ... + 1 = 1/2 * (n^2+n) = O(n^2).$$

В отличие от других методов, сортировка выбором не использует обмены. Сложность алгоритма измеряется числом сравнений и равна $O(n^2)$.

1.2Анализ наилучшего случая

Наилучший случай – когда исходный список уже отсортирован. На i-ом проходе производится n-i сравнений на подсписке A[i+1]–A[n] и ничего не меняется; сложность составляет O(n^2).

1.3Анализ наихудшего случая

Наихудший случай возникает, когда список отсортирован по убыванию. На i-ом проходе производится n-i сравнений на подсписке A[i+1]–A[n] и A[n-i] элемент вставляется на место A[i]; сложность составляет O(n^2).

Подтвердим расчет наиболее худшего случая, когда массив отсортирован по убыванию на конкретном примере.

Псевдокод алгоритма сортировки выбором:

```
1 for i = 1,2, 3, ..., n:
2       key := i
3       j := i +1
4       for j =j,j+1 … n
5               if A[key] > A[j]
6                       key = j;
7       if (key != i)
8               swap(A[key], A[i])
```

Операция	Количество повторений T	Время выполнения C
1	n	C1
2	n	C2
3	n	C3
4	[n*(n+1)]/2	C4
5	[n*(n+1)]/2	C5
6	∑t	C6
7	n	C7
8	n	C8

Из таблицы общая сложность алгоритма:

$$T(n) = C1*n + C2*n+C3*n+C4*[n*(n+1)]/2 + C5*[n*(n+1)]/2+ C6*\sum t + C7*n+ C8*n;$$

t для худшего и лучшео случаяслучая: [n*(n+1)]/2 =>
В пределе при n -> ∞ можно утверждать, что сложность алгоритма, действительно, будет иметь сложность $O(n^2)$.

2 РЕАЛИЗАЦИЯ АЛГОРИТМА НА ЯЗЫКЕ С

2.1Листинг кода программы

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

struct Ttime {
	int year;
	int month;
	int day;
};

struct Trip {
	struct Ttime start;
	struct Ttime finish;
};

int greater(struct Ttime tr1, struct Ttime tr2) {
	return tr1.year > tr2.year ||
		tr1.year == tr2.year && tr1.month > tr2.month ||
		tr1.year == tr2.year && tr1.month == tr2.month && tr1.day > tr2.day;
}

struct Ttime neighbor(struct Ttime tr1) {
	time_t rawtime;
	struct tm * stTimeInfo;
	time ( &rawtime );
	stTimeInfo = localtime ( &rawtime );
	stTimeInfo->tm_year = tr1.year - 1900;
	stTimeInfo->tm_mon = tr1.month - 1;
	stTimeInfo->tm_mday = tr1.day-1;
	mktime ( stTimeInfo );
	tr1.year = stTimeInfo->tm_year+1900;
	tr1.month = stTimeInfo->tm_mon+1;
	tr1.day = stTimeInfo->tm_mday;
	return tr1;
}

void selSort(struct Trip *array, int n) {
	int i;
	for ( i = 0 ; i < ( n - 1 ) ; i++ ) {
		int pos = i;
		int d;
		for ( d = i + 1 ; d < n ; d++ )
			if (greater(array[pos].start, array[d].start)) pos = d;
		if ( pos != i ) {
			struct Trip swap = array[i];
			array[i] = array[pos];
			array[pos] = swap;
		}
	}
}

void print(struct Trip data) {
```

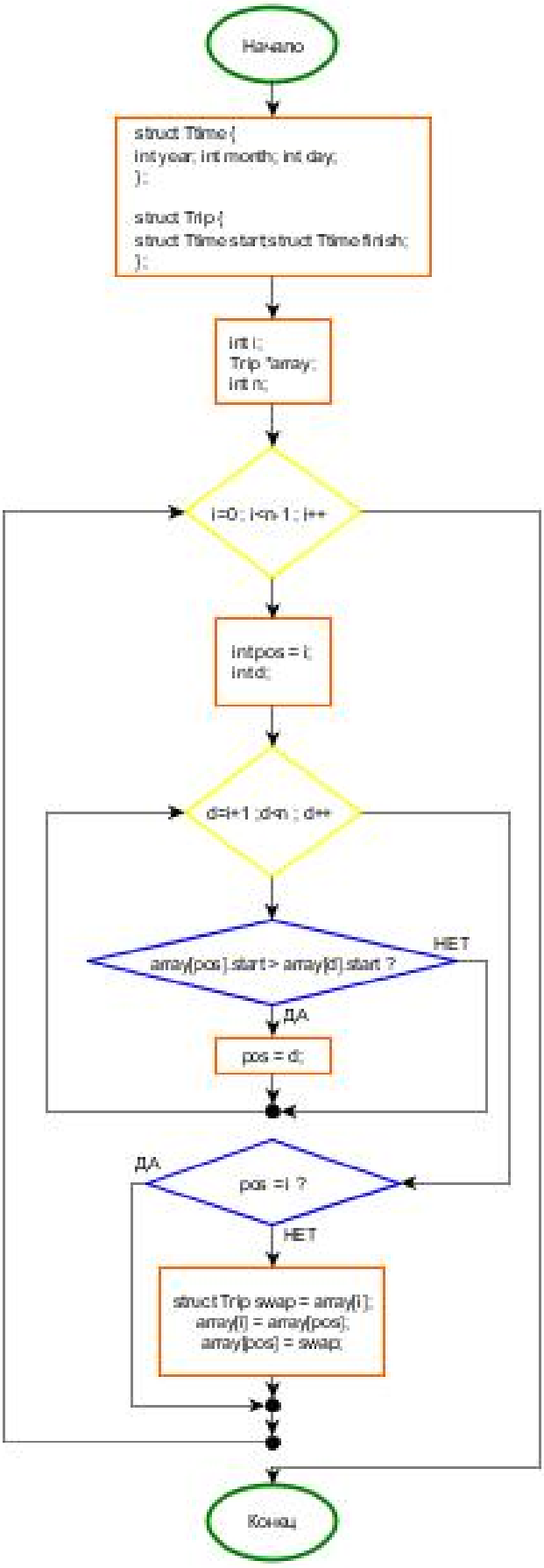
```
	printf("%d-",data.start.year);
	printf("%02d-",data.start.month);
	printf("%02d ",data.start.day);
	printf("%d-",data.finish.year);
	printf("%02d-",data.finish.month);
	printf("%02d\n",data.finish.day);
}

int main(int argc, char** argv) {
	if ( argc < 2 ) {
		printf("The first argument have to be a name of file\n");
		return 0;
	} else {
		FILE *file;
		file = fopen(argv[1],"rt");
		if ( !file ) {
			printf("File doesn't exist\n");
			return 0;
		}
		clock_t begin, end;
		double time_spent;
		begin = clock();
		int size = 2;
		struct Trip *trips = (struct Trip *) calloc(size, sizeof(struct Trip));
		int i = 0;
		while (fscanf(file,"%d-%d-%d %d-%d-%d",
			&trips[i].start.year, &trips[i].start.month, &trips[i].start.day,
			&trips[i].finish.year, &trips[i].finish.month, &trips[i].finish.day) != EOF) {
			i++;
			if ( i == size - 1) {
				size *= 2;
				trips = (struct Trip *) realloc (trips, size* sizeof(struct Trip));
			}
		}
		int n = i;

		selSort(&trips[0],n);

		struct Trip current = trips[0];
		for (i = 1; i < n; i++) {
			if (!greater(neighbor(trips[i].start), current.finish)) {
				if (greater(trips[i].finish, current.finish))
					current.finish = trips[i].finish;
			} else {
				print(current);
				current = trips[i];
			}
		}
		print(current);
		free(trips);
		fclose(file);
		end = clock();
		time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
		printf("Time of program: %f\n", time_spent);
		return 0;
	}
}
```

Блок-схема алгоритма сортировки, применяющейся в программе



2.2Псевдокод алгоритма сортировки

Вход: массив A, состоящий из элементов A[0], A[1], A[2], ..., A[n]

```
for i = 1,2, 3, ..., n:
        key := i
        j := i +1
        for j =j,j+1 … n
                if A[key] > A[j]
                        key = j;
        if (key != i)
                swap(A[key], A[i])
```

3 ПРАКТИЧЕСКОЕ ИССЛЕДОВАНИЕ РАБОТЫ ПРОГРАММЫ

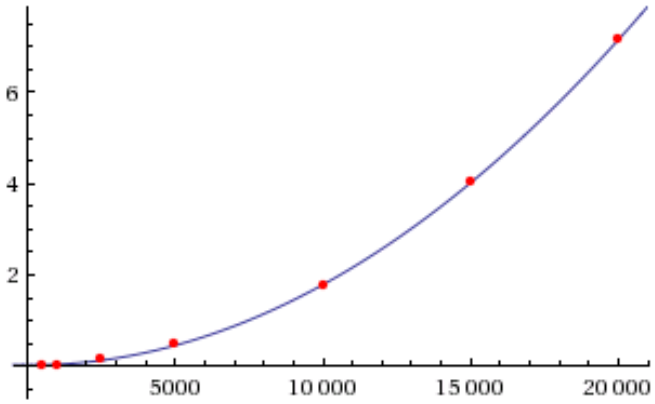
3.1 Экспериментальное исследование зависимости времени выполнения алгоритма от мощности входа

По результатам выполнения программы на различных наборах данных (data%N%.txt) была получена таблица результатов эксперимента.

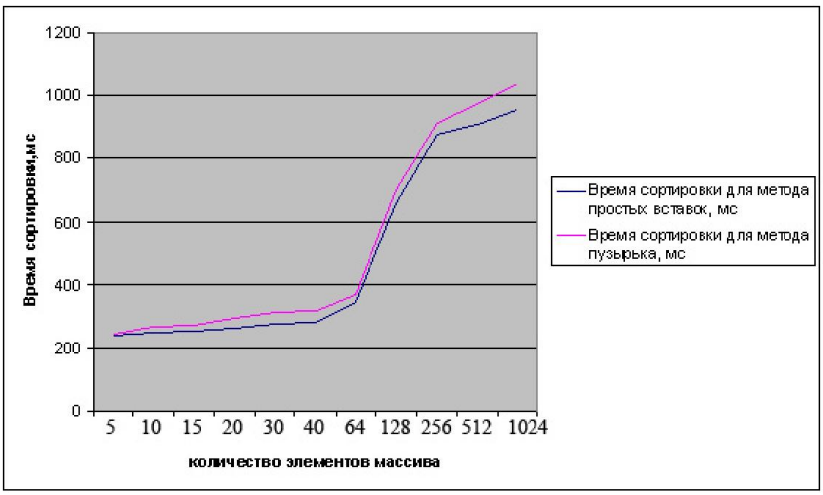
N, мощность выборки	500	1000	2500	5000	10000	15000	20000
T, время выполнения программы, с	0.01	0.03	0.15	0.46	1.76	4.03	7.15

3.2 Сравнение результатов работы алгоритма сортировки программы

По результатам построения графиков по таблице из прошлого пункта отчетливо видна квадратичная зависимость как подтверждение изначального тезиса о сложности алгоритма $O(n^2)$.



Так же если сравнивать алгоритм сортировки выбором с обычным методом сортировки пузырьком, то разница не очень велика, так как порядок сложности алгоритмов в общем случае одинаков.



ВЫВОДЫ

В данной работе была разработана и проанализирована программа решения задачи определения периодичности совершения рейсов на основе метода сортировки выбором. Был сделан краткий теоретический обзор существующих положений в современной теории алгоритмов и вычислительной информатики, а так же упомянута связь данной работы с управлением качества разработки прикладного ПО на объектно-ориентированном языке С.

Так же был подробно рассмотрен алгоритм сортировки выбором и подтверждены все теоретические тезисы на его счет, и в заключении хотелось бы отметить, что данный алгоритм имеет большую сложность на любых выборках и лучше всего использовать более быстрые алгоритмы, такие как quicksort, mergesort.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дональд Кнут Искусство программирования, том 3. Сортировка и поиск = The Art of Computer Programming, vol.3. Sorting and Searching. — 2-е изд. — М.: «Вильямс», 2007. — С. 824. ISBN 0-201-89685-0
2. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн Алгоритмы: построение и анализ = INTRODUCTION TO ALGORITHMS. — 2-е изд. — М.:«Вильямс», 2006. — С. 1296. ISBN 0-07-013151-1
3. https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%B2%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0%D0%BC%D0%B8
4. Язык программирования С, 2-е издание. Год: 2009
   Автор: Kernighan B., Ritchie D. /Керниган Б., Ритчи Д. Издательство: Вильямс
   ISBN: 978-5-8459-0891-9