МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ №1

«СРАВНИТЕЛЬНОЕ ИССЛЕДОВАНИЕ ФУНКЦИЙ ТРУДОЕМКОСТИ АЛГОРИТМОВ МЕТОДОМ ИНТЕРВАЛЬНОГО АНАЛИЗА»

1. Введение

Лабораторная работа ставит своей целью изучение поведения функций трудоемкости количественно-зависимых алгоритмов в реальных интервалах значений мощности множества исходных данных, для определения предпочтений использования того или иного алгоритма. Для сравнения функций трудоемкости используется аппарат интервального анализа, реализованный в виде вычисляемой EXCEL таблицы.

2. Аппарат интервального анализа функций

Выбор того или иного вычислительного алгоритма для решения некоторой задачи требует проведения исследования претендующих алгоритмов не только в целях получения асимптотических оценок функции трудоемкости, т.е. оценки сложности, но и детального исследования для определения размерности и иных характеристик множества реальных исходных данных задачи, которые существенно влияют на рациональный выбор алгоритма.

Проблема состоит в том, что не всегда алгоритм, имеющий асимптотически оптимальную трудоемкость (в смысле оценок O или Θ) имеет лучшие показатели для реального множества исходных данных из-за существенного различия констант, скрывающихся за O или Θ асимптотическими оценками.

Решение данной проблемы лежит в области практического сравнительного анализа вычислительных алгоритмов и связано с выполнением следующих этапов:

- детальный анализ трудоемкости алгоритмов, т.е. получение в явном виде функции трудоемкости, отметим, что асимптотической оценки сложности недостаточно для сравнительного анализа.
- сравнительный анализ функций трудоемкости претендующих алгоритмов с целью выбора рационального алгоритма решения данной задачи при реальных ограничениях множества исходных данных.

Реализация второго этапа - сравнительного анализа функций трудоемкости предполагает определение тех характеристических значений реального множества исходных данных задачи (**Da**) при которых предпочтение может быть отдано одному из анализируемых алгоритмов. В ходе дальнейшего изложения будем считать, что таким характеристическим значением является размерность множества исходных данных - **n** , а трудоемкость алгоритма в

основном определяется количеством исходных данных - **f(Da) = f(n)**. Такое предположение охватывает достаточно широкий класс количественно-зависимых алгоритмов, а так же количественно-параметрических алгоритмов со слабым параметрическим влиянием.

Таким образом сравнительный анализ функций трудоемкости предполагает определение тех интервалов аргумента функций трудоемкости, при которых данный алгоритм может быть выбран в качестве рационального. В связи с этим предлагаемая ниже методика названа *интервальным анализом функций* (трудоемкости алгоритмов) по аналогии с асимптотическим анализом - определением сложности алгоритмов.

Исходными данными для интервального анализа функций являются известные функции трудоемкости алгоритмов, а результатами - предпочтительные интервалы характеристических значений множества исходных данных задачи для рационального применения того или иного алгоритма.

Реально (в смысле дополнительных критериев целесообразности) выбор данного алгоритма, как предпочтительного, может быть произведен даже тогда. когда его трудоемкость на данном интервале несколько хуже, не более, чем на порог φ, чем трудоемкость других алгоритмов, или же несколько алгоритмов могут быть использованы на этом интервале эквивалентно, с расхождением на более чем на порог φ. Это приводит к необходимости введения *специальных обозначений* в аппарате интервального анализа.

2.1. Система обозначений в интервальном анализе функций.

Пусть трудоемкости двух алгоритмов заданы в явном виде функциями $f(n)$ и $g(n)$ соответственно, пусть задан также порог φ допустимого расхождения значений функций на интервале, мера $\pi(f(n),g(n))$ расхождения функций при данном n и интервал (a,b) значений n, тогда будем говорить, что:

а) $f(n)=\Delta_{\varphi}(g(n)),\quad$ *если* $\min\limits_{n\in(a,b)}\{\pi(f(n),g(n))\}\geq\varphi$

б) $f(n)=\Theta_{\varphi}(g(n)),\quad$ *если* $\max\limits_{n\in(a,b)}|\{\pi(f(n),g(n))\}|<\varphi$

в) $f(n)=O_{\varphi}(g(n)),\quad$ *если* $\max\limits_{n\in(a,b)}\{\pi(f(n),g(n))\}\leq-\varphi$

2.2. Методика интервального анализа

На основании введенной системы обозначений может быть предложена следующая методика интервального анализа функций трудоемкости алгоритмов:

- Определение для данной конкретной задачи допустимого интервала изменения размерности множества исходных данных (a,b);
- Получение в явном виде функций трудоемкости анализируемых алгоритмов $f(n)$ и $g(n)$;
- Разбиение интервала (a,b) на подинтервалы, в каждой целочисленной точке которых явно выполняется одно из следующих соотношений для принятой меры $\pi(f(n),g(n))$:
  i. Если $\varphi - \pi(f,g) < 0$, то $f(n) = \Delta_{\varphi}(g(n))$ и предпочтительным на данном подинтервале является алгоритм, имеющий функцию трудоемкости $g(n)$.
  ii. Если $|\pi(f,g)| - \varphi < 0$, то $f(n) = \Theta_{\varphi}(g(n))$ и оба алгоритма с точностью до порога φ могут быть использованы на этом подинтервале.
  iii. Если $\pi(f,g) + \varphi < 0$, то $f(n) = O_{\varphi}(g(n))$ и предпочтительным на данном подинтервале является алгоритм, имеющий функцию трудоемкости $f(n)$.

2.3. Мера $\pi(f(n),g(n))$

Сформулируем требования, которым должна удовлетворять мера расхождения значений функций $f(n)$ и $g(n)$ при фиксированном *n:*

-антисимметричность, т.е. $\pi(f(n),g(n)) = -\pi(g(n),f(n))$;

-эквивалентность, т.е. $\pi(f(n),g(n)) = 0, если\ f(n) = g(n)$.

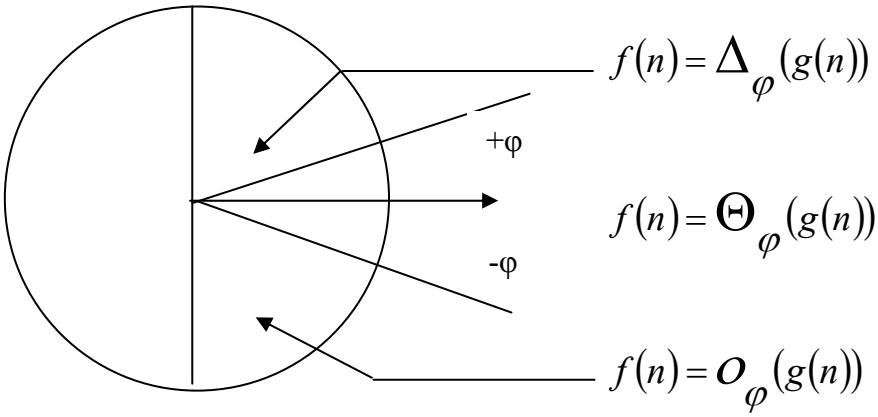
Может быть определено несколько различных мер, удовлетворяющих данным требованиям. В данной работе предлагается следующая функция меры, которая может быть интерпретирована, как угловое расхождение значений аргументов:

$$\pi(f(n),g(n)) = arctg\frac{f(n)}{g(n)} - arctg\frac{g(n)}{f(n)};$$

Поскольку значения функций трудоемкости положительны, то при таком определении меры значения $\pi(f,g)$ ограничены между π/2 и - π/2, а пороговое значение φ может быть интерпретировано, как максимальное значе-

ние угла расхождения, для которого алгоритмы могут быть признаны равноприменимыми на интервале.

Может быть предложена следующая графическая интерпретация методики интервального анализа функций трудоемкости:



3. Порядок выполнения лабораторной работы

3.1. Для указанной в задании пары функций трудоемкости, указанного интервала и значения φ, определить каково соотношение между функциями трудоемкости на заданном интервале. Для вычисления значений функций и определения соотношений использовать таблицу EXCEL -Приложение 1.

3.2. Путем подбора значений аргумента определить интервалы, на которых выполняется соотношение: $f(n)=\Theta_{\varphi}(g(n))$

3.3. Построить графики заданных функций на указанном интервале.

4. Задания к лабораторной работе

4.1 Выбираемые значения φ

1) $\pi/32$
2) $\pi/28$
3) $\pi/24$

4.2 Выбираемые интервалы, для которых необходимо определить соотношение между функциями трудоемкости :

$f=\Delta_{\varphi}(g),\ f=\Theta_{\varphi}(g),\ f=o_{\varphi}(g)$

1. (10,16)
2. (24.36)
3. (50,70)
4. (100,120)
5. (200,230)
6. (500,540)
7. (1000,1060)
8. (2000,2200)

4.3 Варианты функций f, g

	$f(n)$	$g(n)$
1	$2{,}5n^2 + 3n\log n + 18n$	$17n\log n + 38n$
2	$8n^3 + 4n^2 + 4n$	$48n^{\log_2 7}$
3	$\ln n^{\ln n}$	$2^{\sqrt{n}}$
4	$2{,}5n^2 + 6n$	$7{,}5n\sqrt{n} + 22n$
5	$14n\ln n + 22n$	$4n\sqrt{n} - 11n$
6	$n^{\frac{\ln n}{2}}$	$e^{1{,}5\sqrt{n}}$
7	$6 \cdot 2^{n-3}$	$18n^4 + 48n^3 + 324n^2$
8	$44\ln^3 n + 24\ln^2 n$	$1{,}5n + 8$
9	$19\sqrt{n} + 54$	$28\log_2^2 n$
10	$11n\log_2^2 n + 7n\log_2 n$	$14n\sqrt{n} + 89\sqrt{n}$
11	$\sqrt{\frac{n}{8}}^{\sqrt{\frac{n}{8}}}$	$n^{\ln\ln n}$
12	$\ln\ln n^{\sqrt{n}}$	$\sqrt{n}\ln\ln n$
13	$22n^2\sqrt{n} + 7n\ln n$	$\frac{8n^3}{\ln n} + 16\sqrt{n}^3$
14	$\frac{2^{n-4}}{\sqrt{n}}$	$1{,}618^n$
15	$\ln\ln n^{\sqrt{n}}$	$e^{\sqrt{\ln n \cdot \ln\ln n}}$
16	$17n^3 + 19n^2\ln n$	$3n^4 - 24n^2\sqrt{n}$
17	$27n^{\log_2 5}$	$37n^2\ln n$

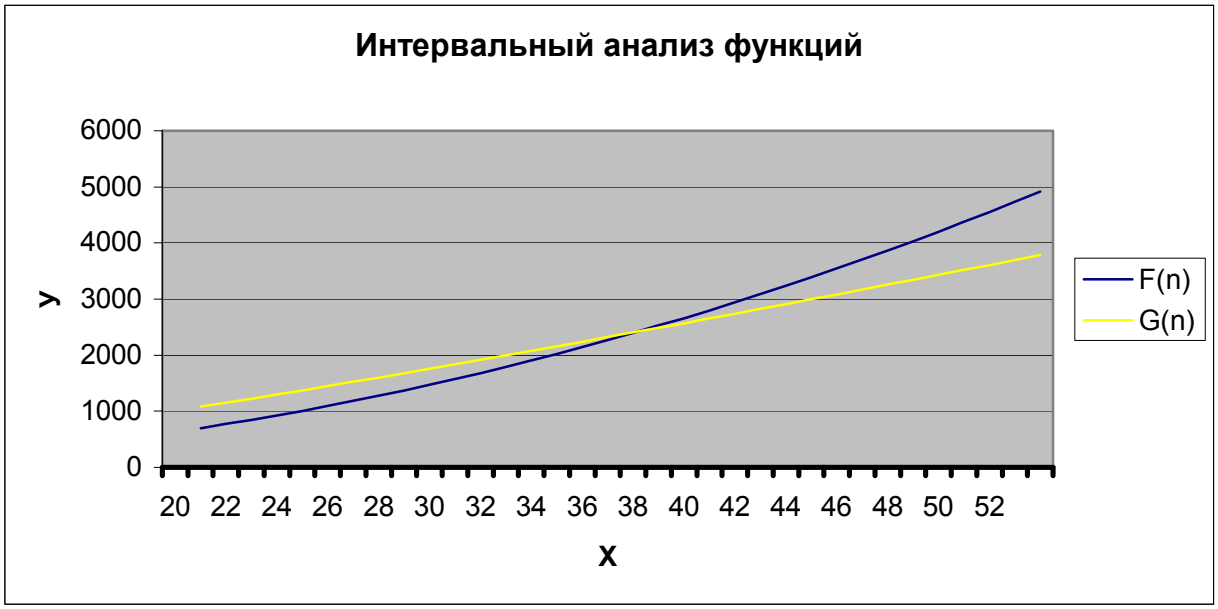
18	$29\sqrt{n}\ln n$	$7\log_2^7 n$
19	$\frac{59n}{\ln n}$	$14\sqrt{n}^{\log_{10} 11}$
20	$11\ln\left(2^{n-1}+3^{n-2}\right)$	$19\left(n-\ln n\right)$
21	$n^{\frac{\sqrt{n}}{8}}$	$e^{\ln^3 n}$
22	$9\frac{\sqrt{n}}{\ln n}$	$31\frac{\ln^3 n}{\sqrt{n}}$
23	$154\ln\left(2^{\sqrt{n}}+128\right)$	$9\sqrt{n}\ln n$
24	$1{,}618^{\sqrt{n}}$	$1{,}5^{\sqrt{n}\ln\ln n}$
25	$23n^2\log_2^2 n$	$4n^3-21n^2$
26	$3\ln n^{\ln n}$	$17n^4+39n^3$
27	$14n^{\log_2 14}$	$45n^{\log_3 45}$
28	$512\ln\ln n$	$7n^{\log_{10} 2}$
29	$68n^{\log_2 7}$	$6n^3+28n^2$
30	$86n^{\log_2 3}$	$4n^2+18n$
31	$2n^3+24n^2+196$	$48n^{\log_3 24}$
32	$4n^4+18n^3$	$76n^{\log_2 14}$

5. Приложение 1 - Таблица интервального анализа функций

Интервальный анализ функций						Порог в долях Pi K=32	
						Pi=3,14159	Fi = Pi/K
							0,09817
F(n)=1,75*n*n						G(n)=18*n*Ln(n)	

n	F(n)	G(n)	ArcTg (F/G)	ArcTg (G/F)	Pi(f,g)	f=o(g)	f=Teta(g)	f=DELTA(g)
20	**700,000**	**1078,464**	0,575722	0,995074	-0,4194	-0,3212	0,3212	0,5175
21	**771,750**	**1150,829**	0,590723	0,980073	-0,3894	-0,2912	0,2912	0,4875
22	**847,000**	**1224,053**	0,605312	0,965484	-0,3602	-0,2620	0,2620	0,4583
23	**925,750**	**1298,095**	0,619504	0,951292	-0,3318	-0,2336	0,2336	0,4300
24	**1008,000**	**1372,919**	0,633313	0,937483	-0,3042	-0,2060	0,2060	0,4023
25	**1093,750**	**1448,494**	0,646754	0,924043	-0,2773	-0,1791	0,1791	0,3755
26	**1183,000**	**1524,789**	0,659838	0,910959	-0,2511	-0,1529	0,1529	0,3493
27	**1275,750**	**1601,777**	0,672578	0,898218	-0,2256	-0,1275	0,1275	0,3238
28	**1372,000**	**1679,431**	0,684987	0,885809	-0,2008	-0,1026	0,1026	0,2990
29	**1471,750**	**1757,728**	0,697076	0,873720	-0,1766	-0,0785	0,0785	0,2748
30	**1575,000**	**1836,647**	0,708856	0,861941	-0,1531	-0,0549	0,0549	0,2513
31	**1681,750**	**1916,165**	0,720337	0,850459	-0,1301	-0,0319	0,0319	0,2283
32	**1792,000**	**1996,264**	0,731530	0,839266	-0,1077	-0,0096	0,0096	0,2059
33	**1905,750**	**2076,925**	0,742445	0,828352	-0,0859	0,0123	-0,0123	0,1841
34	**2023,000**	**2158,133**	0,753090	0,817707	-0,0646	0,0336	-0,0336	0,1628
35	**2143,750**	**2239,869**	0,763475	0,807322	-0,0438	0,0543	-0,0543	0,1420
36	**2268,000**	**2322,120**	0,773608	0,797188	-0,0236	0,0746	-0,0746	0,1218
37	**2395,750**	**2404,871**	0,783498	0,787298	-0,0038	0,0944	-0,0944	0,1020
38	**2527,000**	**2488,109**	0,793153	0,777644	0,0155	0,1137	-0,0827	0,0827
39	**2661,750**	**2571,820**	0,802580	0,768217	0,0344	0,1325	-0,0638	0,0638
40	**2800,000**	**2655,993**	0,811786	0,759010	0,0528	0,1510	-0,0454	0,0454

41	**2941,750**	**2740,616**	0,820780	0,750017	0,0708	0,1689	-0,0274	0,0274
42	**3087,000**	**2825,678**	0,829566	0,741230	0,0883	0,1865	-0,0098	0,0098
43	**3235,750**	**2911,169**	0,838153	0,732643	0,1055	0,2037	0,0073	-0,0073
44	**3388,000**	**2997,078**	0,846546	0,724250	0,1223	0,2205	0,0241	-0,0241
45	**3543,750**	**3083,397**	0,854752	0,716045	0,1387	0,2369	0,0405	-0,0405
46	**3703,000**	**3170,115**	0,862775	0,708021	0,1548	0,2529	0,0566	-0,0566
47	**3865,750**	**3257,225**	0,870623	0,700174	0,1704	0,2686	0,0723	-0,0723
48	**4032,000**	**3344,718**	0,878299	0,692497	0,1858	0,2840	0,0876	-0,0876
49	**4201,750**	**3432,586**	0,885810	0,684986	0,2008	0,2990	0,1026	-0,1026
50	**4375,000**	**3520,821**	0,893160	0,677636	0,2155	0,3137	0,1173	-0,1173
51	**4551,750**	**3609,416**	0,900355	0,670442	0,2299	0,3281	0,1317	-0,1317
52	**4732,000**	**3698,364**	0,907398	0,663398	0,2440	0,3422	0,1458	-0,1458
53	**4915,750**	**3787,658**	0,914294	0,656502	0,2578	0,3560	0,1596	-0,1596



МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ №2

«ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ СРЕДНЕЙ ТРУДОЕМКОСТИ АЛГОРИТМА ДЛЯ ЗАДАЧИ ПОИСКА МАКСИМУМА В МАССИВЕ»

1. Введение

Лабораторная работа посвящена экспериментальной проверке теоретической оценки трудоемкости алгоритма поиска минимума и включает ознакомление с принципами использования генератора случайных чисел для генерации наборов исходных данных. Обработка результатов эксперимента предполагает так же определение зависимости стандартного отклонения от количества экспериментов.

В приложениях к описанию лабораторных работ приведены исходные тексты для соответствующих экспериментальных программ и таблицы оформления результатов.

2. Теоретическая оценка среднего количества операций присваивания в алгоритме поиска максимума

Алгоритм поиска максимума последовательно перебирает элементы массива, сравнивая текущий элемент массива с текущим значением максимума. На очередном шаге, когда просматривается **к-**ый элемент массива, переприсваивание максимума произойдет, если в подмассиве из первых **к** элементов максимальным элементом является последний. Очевидно, что в случае равномерного распределения исходных данных, вероятность того, что максимальный из **к** элементов расположен в определенной (последней) позиции равна **1/к**.

Тогда в массиве из **N** элементов общее количество операций переприсваивания максимума определяется как [1]:

$$\sum_{i=1}^{N} 1/i = Hn \approx Ln(N) + \gamma, \_\gamma = 0{,}57$$

Величина **Hn** называется **n**-ым гармоническим числом. Таким образом точное значение (математическое ожидание) среднего количества операций присваивания в алгоритме поиска максимума в массиве из **N** элементов определяется величиной **Hn** (на бесконечности количества испытаний).

3. Зависимость стандартного отклонения средних от количества экспериментов

Экспериментальные данные о количестве операций присваивания максимума при обработке массива из **N** элементов, получаемые на основе генератора случайных чисел имеют определенную погрешность, носящую случайный характер. Эта погрешность, одной из возможных мер которой является стандартное отклонение **σ**, убывает с ростом количества испытаний.

Если мы проведем серию из **к** испытаний при фиксированном значении количества экспериментов в испытании и определим дисперсию и стандартное отклонение случайных величин, которыми являются средние значения количества операций присваивания в каждом из **к** испытаний и повторим такие серии для других значений **Ne**, то стандартное отклонение будет убывать с ростом **Ne.** Заметим, что в данном случае мы имеем дело со стандартным отклонением средних.

Из теории вероятности известно, что стандартное отклонение средних убывает с ростом количества испытаний обратно пропорционально квадратному корню из количества испытаний [2,3]. Таким образом, если мы будем рассматривать **σ** как функцию от **Ne,** то мы можем прогнозировать следующую зависимость :

$$\sigma = \frac{b}{\sqrt{Ne}}$$

Более точная интерпретация стандартного отклонения средних **σ** состоит в том, что с вероятностью 0,68 среднее значение, полученное в некотором испытании, отличается от истинного не более, чем на **σ.**

4. Задание на лабораторную работу.

1. Ознакомится с работой генератора случайных чисел для данного языка программирования.

2. Составить и отладить программу, формирующую массив из случайных чисел.

3. Составить программу, вычисляющую точное значение среднего количества операций присваивания и определяющую экспериментальное среднее количество операций присваивания в алгоритме поиска максимума в массиве случайных чисел.

4. Составить программу, определяющую среднее количество операций присваивания в 10 испытаниях для данного количества экспериментов **Ne**, и для фиксированной длины массива **n.** Пример программы приведен в приложении 3.

Значения **Ne - (1 000,10 000,100 000, 1 000 000, 10 000 000)**

5. На основании проведенных экспериментов заполнить соответствующие ячейки таблицы (EXCEL), вычислить дисперсию и стандартное отклонение средних для каждого значения **Ne -** Лист 1 таблицы - приложение 1.

6. По вычисленным стандартным отклонениям средних построить линейную регрессию, определить значения **k** и **b** уравнения регрессии:

$$\mathbf{Ln\ (\sigma) = k\ *Ln(Ne) + b}$$

Значение **k** должно быть порядка **-0.5** (обратно пропорционально квадратному корню из количества испытаний) - Лист 2 таблицы - приложение 2.

7. На основе полученного уравнения регрессии получить прогноз **σ** для значения **Ne - 10 000 000** и определить ошибку прогноза в процентах.

5. Приложение 1 - Таблица для экспериментальных данных

Экспериментальные данные по испытаниям для N=10

Теоретическое значение $th = \sum_{i=1}^{N} \frac{1}{i}$ **2,9289683**

Ne **1 000**

Студенты: Ф.И.О.

k	*delta*	*delta*delta*
1	0,06696825	0,00448475
2	0,02603175	0,00067765
3	0,03996875	0,00159750
4	0,06296875	0,00396506
5	0,05803175	0,00336768
6	0,05903175	0,00348475
7	0,05679825	0,00322604
8	0,00796875	0,00006350
9	0,00203175	0,00000413
10	0,06796875	0,00461975

D **0,00254908**

Sigma **0,0504884**

6. Приложение 2 - Таблица обработки экспериментальных данных

Количество элементов массива **N=10**

Исходные экспериментальные данные

Ne	Sigma
1,00E+03	0,05048843
1,00E+04	0,01435727
1,00E+05	0,00372455
1,00E+06	0,00138375
1,00E+07	**0,00037789**

Для контроля прогноза

Прогнозируемые значения для Sigma по полиномиальной функции

Xp	Sigma e	Sigma p	
1,00E+07	**0,00037789**	**0,00037584**	
Ln(Xp)	**Ошибка прогноза**	**0,543360%**	**Yp=K*Xp + B**
16,118096			*-7,886360*

Аппроксимация полиномиальной функцией

X=Ln(Ne)	Y=Ln(sigma)		Y=K*X + B	Анализ
6,907755	-2,986011		-3,030296	**K**
9,210340	-4,243499		-4,244312	**-0,527240**
11,512925	-5,592809		-5,458328	**B**
13,815511	-6,582961		-6,672344	**0,611752**

7. Приложение 3 - Программа анализа среднего количества операций присваивания в алгоритме поиска максимума

```
Program testmax;
 { Lab rab  2 }

 Uses crt;
 Const
   np = 10;
 Type
   fix = Longint;
   fp  = Extended;
   Vect= Array [1..100] of fp;
 Var
   n,ne,k         : fix;
   r              : Vect;
   fe,ft,delta    : fp;

   Function Hn(n:fix):fp;
    Var
     i:fix;
     s:fp;
    Begin
       s:=0.0;
       for i:=1 to n do
        begin
           s:=s+(1.0/(1.0*i));
        end;
       Hn:=s;
    End; { Hn }

    Function Max(n,ne:fix):fp;
    Var
       i,j   : fix;
       sc,mx : fp;

    Begin
      sc:=0.0;
      randomize;

      for i:=1 to ne do
      begin

         for j:=1 to n do
         begin
            r[j]:=random;
         end;

         sc:= sc + 1.0;     { count max }
         mx:= r[1];

          for j:=2 to n do
           begin
              if mx < r[j]
              then
               begin
                  mx := r[j];
                  sc:=sc+1.0;
               end;
           end;
       end; { for Ne }

       Max:= sc/(1.0*ne);
    End; {Max}

    Procedure Count;
    Begin
       Clrscr;
       Write('n=');
       Readln(n);
       Write('ne=');
       Readln(ne);

       For k:=1 to np do
        begin
          ft := Hn(n);
          fe := Max(n,ne);
          delta := abs(ft-fe);

          Writeln;
          Writeln('k=',k:12);
          Writeln('ft=',ft:10:8);
          Writeln('fe=',fe:10:8);
          Writeln('delta=',delta:10:8);

          Readln;
        end;
    End; { Count }

Begin
   Count;
End.
```

8. Литература

1. Т. Кормен, Ч. Лейзерсон, Р. Ривест Алгоритмы: построение и анализ. М.: МЦИМО, 2001.-960 с.,263 ил.

2. Гулд Х., Тобочник Я. Компьютерное моделирование в физике: в 2-х частях. Часть 2: Пер.с англ. - М.: Мир, 1990. -400 с., ил.

3. Теория вероятностей (Основные понятия. Предельные теоремы. Случайные процессы). Ю.В. Прохоров, Ю.А. Розанов. Главная редакция физико-математической литературы из-ва «Наука», М., 1973. - 494 с.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ №3

«ЭКСПЕРИМЕНТАЛЬНОЕ ОПРЕДЕЛЕНИЕ КОЛИЧЕСТВА ЭЛЕМЕНТАРНЫХ ОПЕРАЦИЙ ЯЗЫКА ВЫСОКОГО УРОВНЯ В ПРОГРАММНОЙ РЕАЛИЗАЦИИ АЛГОРИТМА»

1. Введение

Лабораторная работа посвящена экспериментальной проверке теоретически полученной функции трудоемкости для алгоритма точного решения задачи о сумме методом полного перебора. Основная задача лабораторной работы - получение практических навыков расстановки счетчика операций в программе на языке высокого уровня.

В приложениях к описанию лабораторной работы приведены исходные тексты для соответствующих экспериментальных программ и таблицы оформления результатов.

2. Понятие элементарных операций в формальной системе языка высокого уровня

Будем понимать под элементарными операциями языка высокого уровня (на примере языка Паскаль) те, которые могут быть представлены в виде элементарных конструкций данного языка (но не обязательно в виде одной машинной команды), а именно за одну элементарную операцию будем считать следующие:

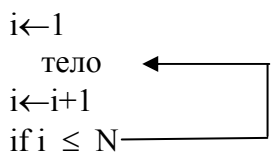
1) операцию присваивания a←b;
2) операцию индексации массива a[i];
3) арифметические операции *,/,-,+;
4) операции сравнения a < b;
5) логические операции or,and,not.

Отметим, что неявно в операцию сравнения входит машинная команда перехода в конструкции if then else.

Цикл не является элементарной операцией, т.к. может быть представлен в виде;

```
for i←1 to N
тело
end;
```

⇔



Таким образом конструкция цикла требует 1+3*N элементарных операций.

В качестве примера построения функции трудоемкости рассмотрим задачу суммирования двумерного массива:

SumM(A,N;Sum);	элементарных операций в строке
Sum←0	1
for i←1 to N	1 + 3*N
for j←1 to N	1+3*N
Sum←Sum+A[i,j];	
1(←)+1(+)+1(i)+1(j)	
end for j	
end for i	
Return (Sum);	
End;	

Таким образом $f_A(N) = 1+1+N*(3+1+N*(3+4)) = 7N^2+4*N+2 = \Theta(N^2)$;

3. Формулировка задачи о сумме

Словесно задача о сумме формулируется как задача нахождения таких чисел из данной совокупности, которые в сумме дают заданное число.

В терминах языка высокого уровня задача формулируется, как задача определения таких элементов исходного массива из N чисел, которые в сумме дают число V (отметим, что задача относится к классу NPC).

Формулировка:

Дано: Массив S[i], i=1, N и число V.

Требуется: определить S_j: $\sum S_j = V$

Получим асимптотическую оценку сложности решения данной задачи при использовании прямого перебора вариантов:

Поскольку исходный массив содержит N чисел, то проверке на V подлежат варианты:

- V содержит 1 слагаемое $\Rightarrow C_N^1 = N$;
- V содержит 2 слагаемых $\Rightarrow C_N^2 = (N*(N-1))/(1*2)$;
- V содержит 3 слагаемых $\Rightarrow C_N^3 = (N*(N-1)*(N-2))/(1*2*3)$;
- и т.д. до N слагаемых.

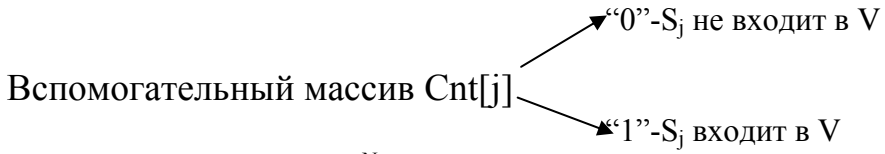
Поскольку сумма биномиальных коэффициентов равна - $(1+1)^n = \sum c_n^k = 2^N$

и для каждого варианта необходимо выполнить суммирование, то оценка сложности имеет вид:

$$f_A(N,V) = O(N*2^N) \qquad (1)$$

4.Точное решение задачи о сумме

Определим вспомогательный массив, хранящий текущее сочетание исходных чисел, подлежащее проверке на V



Решение получено, если $V=\sum_{j=1}^{N} S[j]*Cnt[j]$

Условия существования решения имеют вид:

$$\text{Min}\,(S_i) \le V \le \sum S_i$$

Могут быть предложены следующие две реализации полного перебора:

1.Перебор по C_N^K $\quad\sum_{k=1}^{N} C_N^K = 2^N - 1$

2.Перебор по двоичному счётчику реализованному в массиве Cnt:

00...01

00...10

Вторая идея алгоритмически более проста и сводится к задаче о увеличении счётчика в массиве Cnt на «1».

при 00...0111 увеличение на «1» приводит к сбросу всех правых «1»

при 00...1000 увеличение на «1» приводит к переустановке «0» в «1»

Таким образом процедура перебора имеет вид:

```
TASKSUM(S,N,V;CNT,FL)
FL←false
i←1
repeat
      Cnt[i] ←0
      i←i+1
Until i>N
Cnt[N] ←1
Repeat
 Sum←0
 i←1
 Repeat
  Sum←Sum+S[i]*Cnt[i]
   i←i+1
 Until i>N
if Sum=V
      FL←true
      Return (Cnt,FL)
j←N
While Cnt[j]=1
            Cnt[j]=0
            j←j-1
 Cnt[j] ←1
Until Cnt[0]=1
Return(Cnt,FL)
```

5.Функция трудоемкости процедуры TaskSum

а) В лучшем случае $f\check{}(S,N,V)=\Theta(N)$, если V=S[N]

б) В худшем случае $f\hat{}(S,N,V)= \Theta (N*2^N)$, если решения вообще нет

Получим детальную оценку для худшего случая, используя принятую методику подсчета элементарных операций:

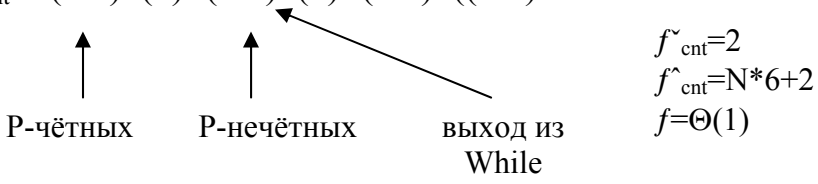
$$f\hat{}=2+N*(3+2)+2+(2^N-1)*\{2+N(3+5)+1+1+f_{cnt}+2+2\} \quad (2)$$

Для получения значения f_{cnt} - количества операций увеличения счетчика рассмотрим по шагам проходы цикла While :

CNT	Количество проходов в While	операций
001	1	6+2
010	0	2
011	2	2*6+2
100	0	2
101	1	6+2
110	0	2
111	3	3*6+2

таким образом:

$f_{cnt} = (1/2)*(2)+(1/2)*(2)+(1/2)*((1/2)*1*6+1/4*2*6+1/8*3*6+\ldots)=$



$=2+1/2*6(1/2^1+2/2^2+3/2^3+\ldots)=2+3*(\sum_{k=1}^{\infty} k/2^k);$

Так как $\sum k*x^k = x/(1-x)^2$, [1]

то $\sum k*(1/2)^k = (1/2)/(1-(1/2))^2 = 2$, и следовательно

$f_{Cnt} = 8$ (! и не зависит от длины счетчика)

Подстановка f_{Cnt} в (2) дает :

$f\hat{}_A(N) = 4+5*N+(2^N-1)*(8*N+16)$, и окончательно:

$f\hat{}_A(N) =8*N*2^N+16*2^N-3*N-12,$

что согласуется с асимптотической оценкой - формула (1).

6. Задание на лабораторную работу.

1. Ознакомится с программой на языке Паскаль, реализующей алгоритм точного решения задачи о сумме - Приложение 2.

2. Включить в процедуру FindSum строки счетчика элементарных операций в соответствии с принятой методикой.

3. Отладить полученную программу в среде Borland Paskal до совпадения экспериментального значения счетчика с теоретическим значением.

4. Для заданных значений **n = (16,17,..,23,24)** заполнить в MS Excel итоговую таблицу - Приложение 1

7. Приложение 1 - Таблица для экспериментальных данных

Лабораторная работа № 3

Экспериментальный анализ алгоритма прямого перебора для решения задачи о сумме

f(n)= 8*n*2^n + 16*2^n - 3*n - 12

n	2^n		f(n) - теория	f(n) - эксперимент
16	65 536		9 437 124	
17	131 072		19 922 881	
18	262 144		41 942 974	
19	524 288		88 080 315	
20	1 048 576		184 549 304	
21	2 097 152		385 875 893	
22	4 194 304		805 306 290	
23	8 388 608		1 677 721 519	
24	16 777 216		3 489 660 844	

8. Приложение 2 - Программа точного решения задачи о сумме

```
PROGRAM TaskSum;
Uses Dos,Crt;
Type
   fp  = Extended;
   fix = Longint;
   Vector = Array [0..100] of fix;

Var
  Number,Count     : Vector;
  N                : fix;
  V                : fix;
  Flag             : Boolean;

                c : fix; { счетчик операций }

  Procedure SetNumber;
  Var
     i : fix;
  Begin
    ClrScr;
    Write('N=');
    Readln(N);

    Write('V=');
    Readln(V);

    For i:=1 to N do
     begin
        Number[i]:=Random(100)+50;
     end;
  End;

  Procedure FindSum(N,V : fix; Var S,Cnt : Vector; Var Flag: Boolean);
  Var
    i,j : fix;
    Sum : fix;

  Begin

    Flag:=False;
                          c:=c+1;  { пример вставки строки счетчика для Flag:=False}
    i:=1;
                          c:=c+1;  { пример вставки строки счетчика i:=1}
    Repeat
     begin
        Cnt[i]:=0;
        i:=i+1;
     end;
    Until i>N;
    Cnt[N]:=1;
```

```
    Repeat
     begin
       sum:=0;
       i:=1;

       Repeat
        begin
          sum:=sum + Cnt[i]*S[i];
          i:=i+1;
        end;
       Until i>N;

       If sum = V
         then
          begin
            Flag:=True;
            Write('OK');
            Readln;
            Halt;
           end;

        j:=n;
        While Cnt[j]=1 do
         begin
           Cnt[j]:=0;
           j:=j-1;
         end;

        Cnt[j]:=1;
      end;
   Until Cnt[0] = 1;
  End;

BEGIN
  Randomize;
  SetNumber;
                                c:=0; { обнуление счетчика операций }
  FindSum(N,V,Number,Count,Flag);
  Writeln('f(',N:2,')=',c:12);
  Readln;
END.
```

9. Литература

1. Т. Кормен, Ч. Лейзерсон, Р. Ривест Алгоритмы: построение и анализ. М.: МЦИМО, 2001.-960 с.,263 ил.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ №4

«ОПРЕДЕЛЕНИЕ СРЕДНЕГО ВРЕМЕНИ ВЫПОЛНЕНИЯ ОБОБЩЕННОЙ ЭЛЕМЕНТАРНОЙ ОПЕРАЦИИ МЕТОДОМ ТЕСТОВЫХ ПРОГОНОВ»

1. Введение

Лабораторная работа посвящена экспериментальному определению среднего времени выполнения обобщенной элементарной операции в языке высокого уровня и получению на этой основе прогноза времени выполнения программы для больших размерностей множества исходных данных в зависимости от типов данных (целые, длинные целые, действительные).

В приложениях к описанию лабораторных работ приведены исходные тексты для соответствующих экспериментальных программ и таблицы оформления результатов.

2. Экспериментальное определение среднего времени выполнения обобщенной элементарной операции

Анализ трудоемкости алгоритмов позволяет получить функциональную зависимость между параметрами исходной задачи (мощностью множества исходных данных) и количеством выполненных элементарных операций в заданной среде реализации.

При этом переход от функции трудоемкости к времени выполнения программы для алгоритма с известной трудоемкостью зависит от:

1. времени выполнения элементарных операций в среде языка реализации алгоритма.

2. специфики реализации алгоритма (квалификация программиста, структуры данных и др.) на данном языке программирования.

Таким образом, имея аналитическую оценку трудоемкости данного алгоритма - **fa(N)** и ее экспериментальное подтверждение и зная среднее время выполнения обобщенной элементарной операции для данного языка и данного процессора **t ср** можно получить оценку времени выполнения программы:

$$ta(N) = t\,ср * fa(N); \quad (1)$$

Построение (определение) оценки **t ср** методом Гиббсона связано с решением следующих трех задач:

1. Определение времени выполнения элементарных операций в среде языка реализации для различных типов данных (типы операций - целые, действительные).

2. Построение средневзвешенной оценки для типов операций, на основе предположений или экспериментальных оценок по частотной встречаемости операции в рамках данного типа.

3. Построение на этой основе оценки **t ср** с использованием весов, характеризующих особенности данного класса задач

При решении задачи 1 можно воспользоваться техническими данными процессора, только в том случае, когда языком реализации является Ассемблер.

Для языков высокого уровня время выполнения элементарных операций по сравнению с временем выполнения команды процессора корректируется следующими факторами:

1. Особенностями среды языка высокого уровня.

2. Особенностями построения машинного кода, связанными собственно с процессором и данной версией компилятора.

3. Принятой в языке системой адресации, структурой хранения данных и массивов, размещением текущих переменных в регистрах.

4. Местоположением операндов операции в иерархической структуре памяти ЭВМ - кэш 1-ого уровня, кэш 2-ого уровня, основная оперативная память.

Прямой учет всех особенностей достаточно затруднителен, поэтому альтернативой метода Гиббсона может выступать определение времени выполнения обобщенной элементарной операции программным путем с использованием процедур службы времени данного языка программирования. Получение достаточно точных значений требует усреднения относительно реального времени выполнения порядка нескольких секунд, что определяется точностью часов.

Таким образом или программа, реализующая данный алгоритм, работает достаточно долго при небольших размерностях множества исходных данных (например реализация алгоритма для задаче о сумме - лабораторная работа №3), или необходимо организовать циклическое повторение вызова программы для псевдослучайных наборов исходных данных для обеспечения совокупного времени выполнения порядка секунд.

Для получения средневзвешенной оценки **t ср** необходимо получить значения **ta(N)** для различных значений N, что определяет следующую методику определения среднего времени выполнения элементарной операции:

1. Измерение времени **ta(N)** для различных значений N;

2. Экспериментальное или теоретическое получение функции трудоемкости алгоритма **fa(N);**

2. Определение **t ср = ta(N) / f(N)** для каждого значения N**;**

3. Усреднение полученных значений **t ср** по количеству испытаний.

3. Интерпретация результатов и построение прогноза

Описанная выше методика получения средневзвешенной оценки среднего времени выполнения элементарной операции в языке высокого уровня позволяет провести ряд дополнительных исследований.

а) Зависимость **t ср** от типов данных

Полученные по вышеуказанной методике значения **t ср** для различных типов данных (целые, длинные целые, действительные и т.д.) позволяют при сравнительном анализе определить качественные различия процессоров при обработке данных различного типа, что может являться основой одного из процессорных тестов.

б) Пересчет **t ср** в такты процессора

Пересчет полученных значений **t ср** в такты процессора (в единицы тактовой частоты) позволяет провести сравнительный анализ относительной скорости выполнения элементарных операций для различных типов данных у процессоров с различной тактовой частой и различной внутренней архитектурой. Отметим, что таким образом можно обнаружить преимущества той или иной архитектуры по отношению к данной задаче - т.е. решить проблему выбора процессора как наиболее рационального по отношению к данной задаче.

в) Прогноз времени выполнения программы

Если тестовый алгоритм обладает достаточной устойчивостью по **t ср -** т.е. значения **t ср** полученные для различных N имеют малую дисперсию, то возникает возможность прогнозировать поведение программы для больших размерностей исходных данных, что позволяет определить границы применимости данного алгоритма на данном процессоре.

4. Задание на лабораторную работу.

1. Ознакомится с программой, определяющей временя выполнения процедуры решения задачи о сумме - Приложение 1. Обратить внимание на точность возвращаемого функцией GetTime значения.

2. Подобрать такое значение N, при котором время выполнения составляет порядка 0,5 секунды на данном процессоре для целого типа данных (Word).

3. Начиная с определенного в пункте 2 значения N выполнить 7 последовательных замеров времени выполнения для возрастающих на единицу значений N.

4. Полученные экспериментальные данные занести в расчетную таблицу для соответствующего типа данных - Приложение 2. Обратить внимание, что значения fa(N) в расчетной таблице взяты по результатам лабораторной работы №3.

5. Изменить тип данных в программе, перекомпилировать ее, и проведя 7 замеров времени для тех же значений N, что и в пункте 3, и заполнить листы 2 и 3 расчетной таблицы для типов LondInt и Extended.

6. По полученным временам выполнения процедуры решения задачи о сумме и известной трудоемкости рассчитать время выполнения обобщенной элементарной операции **t ср** в наносекундах.

7.Зная тактовую частоту процессора рассчитать **t ср** в тактах процессора.

8. На основе полученных данных рассчитать средневзвешенное значение **t ср** по 7 испытаниям.

9. На основе полученных данных построить прогноз времени выполнения для значений N вплоть до 34.

5. Приложение 1 - Программа определения времени выполнения процедуры решения задачи о сумме

```
PROGRAM TaskSum;
Uses Dos,Crt;
Type
   fp      = Extended;
   Value = Word {Extended} {Longint};
   Loop  = Word;
   Vector = Array [0..100] of Value;
Var
  Number,Count    : Vector;
  N               : Loop;
  V               : Value;
  Flag            : Boolean;
  Tbegin,Tend,Tfm :fp;
  Function Time : Fp;
  Var
    t         : fp;
    h,m,s,s100 : Word;
  Begin
    GetTime(h,m,s,s100);
    t:=h*3600.0;
    t:=t+(m*60.0);
    t:=t+(s*1.0);
    t:=t+( (s100*1.0)/100.0 );
    Time:=t;
  End;
  Procedure SetNumber;
  Var
     i : Loop;
  Begin
    ClrScr;
    Write('N=');
    Readln(N);
    Write('V=');
    Readln(V);
    For i:=1 to N do
     begin
       Number[i]:=Random(100)+50;
       Writeln(Number[i]:5);
     end;
  End;
  Procedure FindSum(N: Loop;
V: Value; Var S,Cnt: Vector;
Var Flag: Boolean);
  Var
    i,j : Loop;
    Sum : Value;
  Begin
    Flag:=False;
    i:=1;
    Repeat
     begin
       Cnt[i]:=0;
       i:=i+1;
     end;
    Until i>N;
    Cnt[N]:=1;
    Repeat
     begin
       sum:=0;
       i:=1;
       Repeat
        begin
          sum:=sum + Cnt[i]*S[i];
          i:=i+1;
        end;
       Until i>N;
       If sum = V
         then
          begin
            Flag:=True;
            Write('OK');
            Readln;
            Halt;
          end;
       j:=n;
       While Cnt[j]=1 do
        begin
          Cnt[j]:=0;
          j:=j-1;
        end;
       Cnt[j]:=1;
     end;
   Until Cnt[0] = 1;
 End;
BEGIN
  Randomize;
  SetNumber;
  Tbegin:=Time;
  FindSum(N,V,Number,Count,Flag);
  Tend:= Time;
  Tfm := Tend-Tbegin;
  Writeln('ta(',N:2,')=',Tfm:8:2,' sec.');
  Readln;
END.
```

6. Таблица обработки экспериментальных данных

Лабораторная работа № 4

Анализ времени выполнения для алгоритма решения задачи о сумме

Процессор 200 МГц

f(n)= 8*n*2^n + 16*2^n - 3*n - 12

Экспериментальные данные

TYPE = WORD

		f(n) - теория =				
n	2^n	f(n) - эксперимент		Time -s	top - ns	top - takt
16	65 536	9 437 124		0,22	23,31	4,66
17	131 072	19 922 881		0,44	22,09	4,42
18	262 144	41 942 974		0,93	22,17	4,43
19	524 288	88 080 315		1,98	22,48	4,50
20	1 048 576	184 549 304		4,17	22,60	4,52
21	2 097 152	385 875 893		8,62	22,34	4,47
22	4 194 304	805 306 290		18,02	22,38	4,48

Прогнозируемые результаты

		f(n) - теория =			Средние значения	
n	2^n	f(n) - эксперимент	Прогноз	Прогноз	22,48	4,50
			h:m:s	sec		
					t эксп	delta %
23	8 388 608	1 677 721 519	0:00:38	37,72	37,62	0,25%
24	16 777 216	3 489 660 844	0:01:18	78,45		
25	33 554 432	7 247 757 225	0:02:43	162,93		
26	67 108 864	15 032 385 446	0:05:38	337,93		
27	134 217 728	31 138 512 803	0:11:40	700,00		
28	268 435 456	64 424 509 344	0:24:08	1448,27		
29	536 870 912	133 143 986 077	0:49:53	2993,09		
30	1 073 741 824	274 877 906 842	1:42:59	6179,28		
31	2 147 483 648	566 935 682 967	3:32:25	12744,77		
32	4 294 967 296	1 168 231 104 404	7:17:42	26261,96		
33	8 589 934 592	2 405 181 685 649	15:01:09	54068,74		
34	17 179 869 184	4 947 802 324 878	30:53:47	111227,13		