ЛАБОРАТОРНАЯ РАБОТА № 1.
“АСИМПТОТИЧЕСКИЙ АНАЛИЗ КОЛИЧЕСТВЕННО-ЗАВИСИМЫХ АЛГОРИТМОВ”

1. ЦЕЛЬ РАБОТЫ

Изучить поведение функций трудоемкости количественно-зависимых алгоритмов в реальных интервалах значений мощности множества исходных данных. На основании этого сделать предпочтительный выбор того или иного алгоритма. Для сравнения функций трудоемкости использовать аппарат интервального анализа, реализованный в виде программы на языке C++.

2. ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

2.1 Основные определения асимптотического анализа алгоритмов

При использовании алгоритмов для решения практических задач мы сталкиваемся с проблемой рационального выбора алгоритма решения задачи. Решение проблемы выбора связано с построением системы сравнительных оценок, которая в свою очередь существенно опирается на формальную модель алгоритма.

Под *трудоёмкостью* алгоритма для данного конкретного входа – $F_a(N)$, будем понимать количество «элементарных» операций, совершаемых алгоритмом для решения конкретной проблемы в данной формальной системе. При анализе поведения функции трудоемкости алгоритма часто используют принятые в математике асимптотические обозначения, позволяющие показать *скорость роста функции*, маскируя при этом конкретные коэффициенты.

Такая оценка функции трудоемкости алгоритма называется *сложностью алгоритма* и позволяет определить предпочтения в использовании того или иного алгоритма для больших значений размерности исходных данных.

В асимптотическом анализе приняты следующие обозначения:

1) Оценка Θ (тетта)

Пусть $f(n)$ и $g(n)$ – положительные функции положительного аргумента, $n \geq 1$ (количество объектов на входе и количество операций – положительные числа), тогда: $f(n) = \Theta(g(n))$, если существуют положительные c_1, c_2, n_0, такие, что: $c_1 * g(n) \leq f(n) \leq c_2 * g(n)$, при $n > n_0$

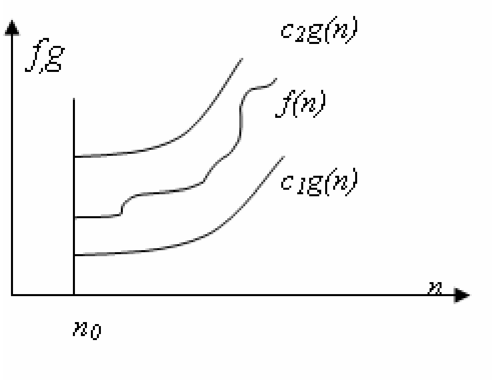


Рисунок 2.1 – Оценка *Θ*

Обычно говорят, что при этом функция *g(n)* является асимптотически точной оценкой функции *f(n),* т.к. по определению функция *f(n)* не отличается от функции *g(n)* с точностью до постоянного множителя.

Отметим, что из $f(n) = \Theta(g(n))$ следует, что $g(n) = \Theta(f(n))$.

Примеры:

1) $f(n)=4n^2+nlnN+174$ – $f(n)= \Theta(n^2)$;

2) $f(n)=\Theta(1)$ – запись означает, что *f(n)* или равна константе, не равной нулю, или *f(n)* ограничена константой на ∞ : $f(n) = 7+1/n = \Theta(1)$.

2) Оценка *O* (*O* большое)

В отличие от оценки *Θ,* оценка *O* требует, чтобы функция *f(n)* не превышала *g(n)* начиная с $n > n0$, с точностью до постоянного множителя:

$$\exists\, c > 0,\ n0 > 0 :$$

$$0 \le f(n) \le c * g(n),\ \forall n > n0$$

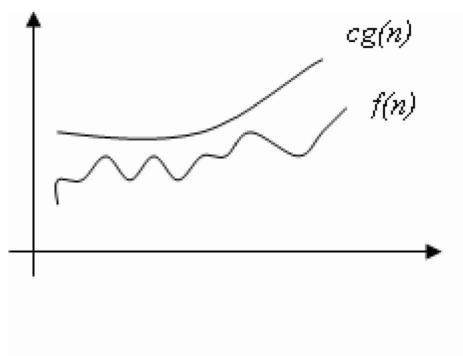


Рисунок 2.2 – Оценка *O* (О большое)

Выбор того или иного вычислительного алгоритма для решения некоторой задачи требует исследования претендующих алгоритмов не только для оценки сложности, но и детального исследования для определения размерности и иных характеристик множества реальных исходных данных задачи. Это существенно влияет на рациональный выбор алгоритма.

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

Это требует практического сравнительного анализа алгоритмов, для чего осуществляется:

- детальный анализ трудоемкости алгоритмов, получение в явном виде функции трудоемкости;
- сравнительный анализ функций трудоемкости претендующих алгоритмов для выбора рационального алгоритма решения данной задачи при реальных ограничениях множества исходных данных.

Реализация второго этапа – сравнительного анализа функций трудоемкости – предполагает определение характеристических значений реального множества исходных данных задачи (Da), при которых предпочтение может быть отдано одному из анализируемых алгоритмов. Можно считать, что таким характеристическим значением является размерность множества исходных данных - n , а трудоемкость алгоритма в основном определяется количеством исходных данных - f(Da) = f(n).

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 предлагаемый ниже метод назван *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 *результат*ами - *предпочтительные интервалы значений множества исходных данных* задачи для рационального применения того или иного алгоритма.

Реально 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 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.

2.2. 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 считаем, что:

а) $$f(n)=\Delta_{\varphi}(g(n)),\quad \textit{если}\ \min_{n\in(a,b)}\{\pi(f(n),g(n))\}\ge\varphi\,;$$

б) $$f(n) = \Theta_{\varphi}(g(n)), \quad \text{если} \max_{n \in (a,b)} |\{\pi(f(n), g(n))\}| < \varphi;$$

в) $$f(n) = O_{\varphi}(g(n)), \quad \text{если} \max_{n \in (a,b)} \{\pi(f(n), g(n))\} \le -\varphi.$$

2.3. Метод интервального анализа

На основании введенной системы обозначений может быть предложен метод интервального анализа функций трудоемкости алгоритмов, включающий этапы:

- 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 подынтервалы, в каждой целочисленной точке которых явно выполняется одно из следующих соотношений для принятой меры $\pi(f(n), g(n))$:
    1) если $\varphi - \pi(f,g) < 0$, то $f(n) = \Delta_{\varphi}(g(n))$ и предпочтительным на данном подынтервале является алгоритм, имеющий функцию трудоемкости $g(n)$.
    2) 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.
    3) если $\pi(f,g) + \varphi < 0$, то $f(n) = O_{\varphi}(g(n))$ и предпочтительным на данном подынтервале является алгоритм, имеющий функцию трудоемкости $f(n)$.

2.4. Мера $\pi(f(n), g(n))$

Мера расхождения значений функций $f(n)$ и $g(n)$ при фиксированном n должна удовлетворять следующим требованиям:

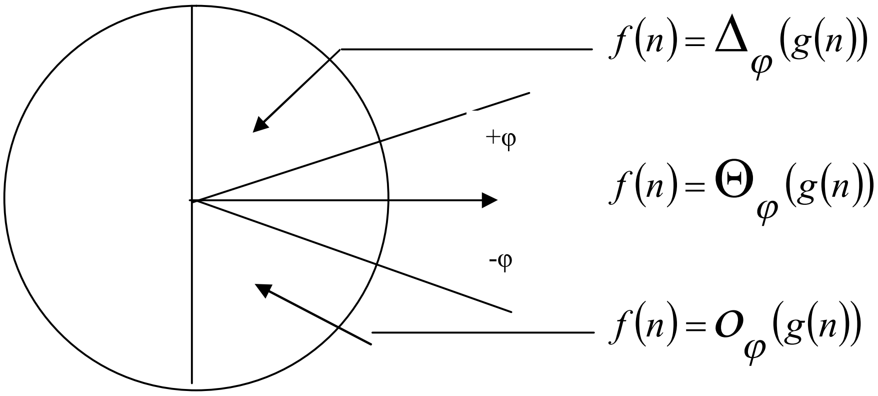
- $\pi(f(n), g(n)) = -\pi(g(n), f(n))$;
- $\pi(f(n), g(n)) = 0, \text{если } f(n) = g(n)$.

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

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

Поскольку значения функций трудоемкости положительны, то при таком определении меры значения $\pi(f,g)$ ограничены между π/2 и - π/2. Пороговое значение φ может быть интерпретировано, как максимальное значение угла расхождения, для которого алгоритмы могут быть признаны равноприменимыми на интервале.

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



3. ХОД РАБОТЫ

1. Ознакомиться с теоретическим разделом настоящих методических указаний и повторить соответствующий лекционный материал.

2. Для указанной в варианте задания пары функций трудоемкости, целочисленных интервалов {(20;50), (100;120), (500, 540)} и значений $\varphi = \{\frac{\pi}{32},\frac{\pi}{24},\frac{\pi}{18}\}$, определить, каково соотношение между функциями трудоемкости на заданном интервале. Для вычисления значений функций и определения соотношений написать программу на языке С. Пример программы приведен в приложении. Результаты работы программы сохранять в текстовые файлы. Сведения об основных алгебраических функциях и основных конструкциях языка С можно получить либо из системы помощи интегрированной среды разработки, либо из методических указаний по АиП, посвященных программированию на языке С.

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

4. Построить графики заданных функций на указанном интервале (вручную или с помощью любого программного обеспечения).

5. Сделать выводы по работе, оформить отчет, подготовить ответы на контрольные вопросы.

4. ВАРИАНТЫ ЗАДАНИЙ

	$f(n)$	$g(n)$
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	$29\sqrt{n}\ln n$	$7\log_2^7 n$

	$f(n)$	$g(n)$
19	$\dfrac{59n}{\ln n}$	$14\sqrt{n}^{\log_{10} 11}$
20	$11\ln\left(2^{n-1}+3^{n-2}\right)$	$19(n-\ln n)$

5. ПРИМЕР ПРОГРАММЫ ИНТЕРВАЛЬНОГО АНАЛИЗА

Пример программы определения интервалов предпочтения алгоритмов для функций трудоемкости $F(n) = 1.75 * n^2$ и $G(n) = 18 * n * \ln(n)$

```
/* подключение основных используемых в программе библиотек*/
 #include <conio.h>
 #include <math.h>
 #include <stdio.h>
 #include <stdlib.h>
 #include <iomanip.h>

 main()
 {
 double
        Fn,            //F(n)
        Gn,            //G(n)
        ATg_FG,
        ATg_GF,
        pi,
        Nbegin,        //Левая граница интервала
        Nend,                   //Правая граница интервала
        step,          //Шаг изменения аргумента
        phi,           //Значение угла
        k,             //Коэффициент кратности
        Delta,         //Оценка «Дельта»
        Theta,         //Оценка «Тетта»
        O_large,       //Оценка «О-большое»
        ii;            //Значение аргумента функций трудоемкости

        FILE *stream;
              //Указатель на файл, в который осуществляется ввод-вывод
              расчетов
```

```
        stream = fopen("Example_TA.TXT", "w+"); // открытие файла для
записи
                /*Ввод значений границ интервалов, шага изменения
аргумента внутри интервала, коэффициента кратности*/

        cout<<"Input Nbegin  ";     cin>>Nbegin; //Левая граница, ввод
значения
        cout<<"Input Nend    ";     cin>>Nend;     //Правая граница, ввод
значения
        cout<<"Input step    ";     cin>>step;     //Шаг изменения аргумента
        cout<<"Input koefficient";  cin>>k;        //Коэффициент кратности
```

phi = M_PI/k;　　　　/*Определили угол изменения как $\frac{\pi}{k}$, M_Pi – встроенная константа яз.С, число $\pi = 3.1415...$*/

```
     ii=Nbegin;                  //Аргумент функций равен левой границе
интервала
      while (ii<=Nend)
        {       Fn     = 1.75*ii*ii; //Расчет значения функции F(n)
                Gn     = 18*ii*log(ii);      //Расчет значения функции G(n)
                ATg_FG     = atan(Fn/Gn);
                ATg_GF     = atan(Gn/Fn);
                pi      = ATg_FG - ATg_GF;
                Delta   = phi - pi;
                Theta= fabs(pi) - phi;
                O_large = pi + phi;
                fprintf(stream, "%f %f %f %f %f %f %f %f %f\n", ii, Fn, Gn,
ATg_FG, ATg_GF, pi, Delta, Theta, O_large);
                //Запись расчетов в файл
                ii=ii+step;           //Получение следующего значения
аргумента
        }//end while
        fclose(stream);            //Закрыли файл
}// end main
```

6. СОДЕРЖАНИЕ ОТЧЁТА

1. Цель работы.
2. Вариант задания.
3. Текст программы, реализующей расчеты по соответствующему варианту.
4. Анализ результатов работы программы в виде таблицы результатов и графиков.
5. Развернутый вывод по работе.

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Определение алгоритма по Маркову, Колмогорову.
2. Основные требования, предъявляемые к алгоритму.
3. Понятие о трудоемкости алгоритма.
4. Асимптотические обозначения в анализе функции трудоемкости алгоритма.
5. Определение сложности алгоритма.
6. Основные оценки в асимптотическом анализе алгоритмов.
7. Метод интервального анализа функций трудоемкости алгоритмов, его графическая интерпретация.