Занятие 1. Сложность алгоритмов

Упражнение 1. Дан следующий фрагмент кода. Чему равно «О-большое» его времени выполнения?

```
test = 0
for i in range(n):
   for j in range(n):
      test = test + i * j
```

Упражнение 2. Дан следующий фрагмент кода. Чему равно «О-большое» его времени выполнения?

```
test = 0
for i in range(n):
   test = test + 1
for j in range(n):
   test = test - 1
```

Упражнение 3. Дан следующий фрагмент кода. Чему равно «О-большое» его времени выполнения?

```
i = n
while i > 0:
   k = 2 + 2
   i = i // 2
```

Упражнение 4. Иногда константы, которыми пренебрегают в асимптотическом обозначении, играют очень важную роль. Представьте, что два алгоритма выполняют одинаковую задачу. Для работы первого требуется 1500N шагов, второй справляется с тем же заданием за $30N^2$ шагов. При каких значениях N вы отдадите предпочтение тому или иному алгоритму?

Упражнение 5. У вас есть два алгоритма. Один состоит из

$$\frac{N^3}{75} - \frac{N^2}{4} + N + 10 \text{ шагов},$$

другой — из $\frac{N}{2} + 8$ шагов.

Каково должно быть значение N, чтобы вы выбрали первый второй алгоритм?

Время выполнения алгоритма

При подсчете времени работы алгоритма учитывается только время выполнения непосредственно алгоритма и не учитывается ввод данных и вывод результатов.

Для подсчета времени выполнения сортировки в **Lazarus** использовать `GetTickCount (GetTickCount64)`. Функция GetTickCount извлекает число милисекунд, которые истекли с тех пор как система была запущена. Она ограничивается разрешающей способностью системного таймера.

```
Var tm:dword; (or variant)
…..
tm:=GetTickCount;
//алгоритм
tm:=GetTickCount-tm;
```

Возвращаемое значение `tm` - число миллисекунд, которые истекли после того, как система была запущена. Так как `1` сек. = 1000 милисекунд, следовательно, для перевода в секунды делаем следующее

```
tm:=tm div 1000
```

В **PascalABC.NET** использовать функцию Milliseconds:

```
Var tm:integer;
begin
    . . . . .
    tm:=Milliseconds;
    //алгоритм
    tm:=Milliseconds-tm;
    . . . . .
end.
```

Возвращаемое значение `tm` - число миллисекунд.

Дополнительная информация об измерении времени в Windows

http://iproc.ru/programming/windows-timers/

https://habrahabr.ru/post/75234/

В PASCAL ABC.NET необходимо использовать встроенный класс Stopwatch. Пример кода приведен ниже (код взят из примеров интегрированной среды разработки PABCWork.NET\Samples\NETLibraries\Other\).

```
program Time;
var
 ts : System.TimeSpan;
// Stopwatch - класс высокоточного таймера (с точностью до 0.001 с)
begin
  var stopWatch := new System.Diagnostics.Stopwatch;
  stopWatch.Start;
  {Участок кода, для которого будет замеряться время выполнения}
  stopWatch.Stop;
  ts := stopWatch.Elapsed;
  writelnFormat('Время работы: {0:00}:{1:00}:{2:00}.{3:000}', ts.Hours, ts.Minutes, ts.Seconds, ts.Milliseconds);
end.
```

В **Python** можно проделать эту операцию, отметив время начала и время окончания работы алгоритма. В модуле `time` есть функция `time`, которая возвращает текущее системное время в секундах, прошедшее с некоторого произвольно выбранного начального момента. Вызвав эту функцию дважды - в начале и в конце, - и затем посчитав разницу, мы получим точное количество секунд (дробное в большинстве случаев), затраченных на выполнение.

```
import time
start = time.time()
 # алгоритм
end = time.time()
res = end-start
```

Задание 1. Написать в `Python` два варианта программы вычисления суммы чисел

$$1 + 2 + 3 + \cdots + (n - 1) + n.$$

Сделать выводы.

Вариант 1. Вычисление «в лоб», т.е. последовательно в цикле прибавляя очередное значение. Запустить несколько раз, задавая разные значения n (n=10, 1000, 1000000, 1000000000, 1000000000000). Для каждого запуска вычислить и вывести на экран время выполнения в секундах.

Вариант 2. Вычислить эту сумму, используя известную формулу. Запустить несколько раз, задавая разные значения n (n=10, 1000, 1000000, 1000000000, 1000000000000). Для каждого запуска вычислить и вывести на экран время выполнения в секундах.

Задание 2. Написать программу вычисления n-го числа Фибоначчи разными способами (найти самостоятельно в Интернете способы вычисления чисел Фибоначчи – рекурсивный и другие). Вычислить время работы программы на разных исходных данных для каждой реализации и количество вызовов функции.

Задание 3. Рассмотрим алгоритм, который определяет, содержатся ли в массиве повторяющиеся элементы. (Стоит отметить, что это не самый эффективный способ обнаружения дубликатов.)

```
Boolean: ContainsDuplicates(Integer: array[])
    // Цикл по всем элементам массива.
    For i = 0 To <наибольший индекс>

        For j = 0 To <наибольший индекс>
            // Проверяем, являются ли два элемента дубликатами.
            If (i != j) Then
                If (array[i] == array[j]) Then Return True
            End If
        Next j
    Next i

    // Если мы дошли до этой строки, то дубликатов нет.
    Return False
End ContainsDuplicates
```

Алгоритм содержит два цикла, один из которых является вложенным. Внешний цикл перебирает все элементы массива N, выполняя O(N) шагов. Внутри каждого такого шага внутренний цикл повторно пересматривает все N элементов массива, совершая те же O(N) шагов. Следовательно, общая производительность алгоритма составит O(N×N) = O(N^2).

Рассмотрим его улучшенную версию.

```
Boolean: ContainsDuplicates(Integer: array[])
  // Цикл по всем элементам массива, кроме последнего.
  For i = 0 To <наибольший индекс> - 1
          // Цикл по элементам после элемента i.
          For j = i + 1 To <наибольший индекс>
                  // Проверяем, являются ли два элемента дубликатами.
                  If (array[i] == array[j]) Then Return True
          Next j
  Next i

  // Если вы дошли до этой строки, то дубликатов нет.
  Return False
End ContainsDuplicates
```

Определите время работы для новой версии алгоритма.

Задание 4. В таблице 1.1 показана взаимосвязь между сложностью задачи N и производительностью алгоритмов, описанных разными функциями. Эту же зависимость можно продемонстрировать еще одним способом — определить максимальную сложность задачи, которую способен выполнить компьютер с определенной скоростью в течение конкретного времени.

Таблица 1.1. Величины функций для различных входных данных

N	$\log_2 N$	sqrt (N)	N	N^2	2^N	$N!$
1	0	1	1	1	2	1
5	2,32	2,23	5	25	32	625
10	3,32	3,16	10	100	1 024	1×10^9
15	3,90	3,87	15	225	$3,3 \times 10^4$	$2,9 \times 10^{16}$
20	4,32	4,47	20	400	1×10^6	$5,24 \times 10^{24}$
50	5,64	7,07	50	2500	$1,1 \times 10^{15}$	$1,8 \times 10^{83}$
100	6,64	10,00	100	1×10^4	$1,3 \times 1\ 0^{30}$	1×10^{198}
1 000	9,96	31,62	1 000	1×10^6	$1,1 \times 10^{301}$	–
10 000	13,28	100,00	1×10^4	1×10^8	–	–
100 000	16,60	316,22	1×10^5	1×10^{10}	–	–

Предположим, компьютер проделывает 1 млн шагов алгоритма за 1 с и алгоритм выполняется в течение времени O(N^2). В таком случае через 1 ч машина решит задачу, у которой N = 60 000 (поскольку 60 000^2 = 3 600 000 000 — это количество произведенных шагов за указанный промежуток).

Создайте таблицу, где будет представлен размер наибольшей задачи N, которую сможет выполнить компьютер для каждой из функций, приведенных в таблице 1.1, за секунду, минуту, час, день, неделю, год.

Задание 5. Программа берет в качестве входных параметров N букв и генерирует из них все возможные пары. Например, из букв ABCD выстраиваются следующие комбинации: AB, AC, AD, BC, BD и CD (подразумевается, что AB и BA — одна и та же пара). Каково время работы алгоритма?

Задание 6. Рассмотрим два алгоритма покраски забора.

```
Algorithm1()
  For i = 0 To <количество досок в заборе> - 1
          <Красим доску под номером i.>
  Next i
End Algorithm1

Algorithm2(Integer: first_board, Integer: last_board)
  If (first_board == last_board) Then
          // Имеется только одна доска. Красим только ее.
          <Красим доску под номером first_board.>
  Else
          // Досок больше одной, делим их на две группы и красим рекурсивно.
          Integer: middle_board = (first_board + last_board) / 2
          Algorithm2(first_board, middle_board)
          Algorithm2(middle_board, last_board)
  End If
End Algorithm2
```

Каково время работы этих алгоритмов (N — количество досок в заборе)?

Какой вариант лучше?