**БИОАЛГОРИТМИКА**

**9.08 Занятие 1.**

1. **Введение.**
   1. При постановке задачи, которую выполняет алгоритм, будем явно описывать «**дано**» (входные данные) и «**надо**» (выходные данные).
   2. Псевдокод для записи алгоритмов. Соглашения:
      * Похож на Паскаль, но со вставками на естественном языке.
      * Опускаем (иногда) «инженерные» детали: объявление переменных, типизацию, обработку ошибок, etc.
      * Блоки кода структурируются отступами.
      * Элементы массива могут обозначаться нижним индексом или квадратными скобками. Индексы (как правило) нумеруются с нуля.

Пример: Алгоритм поиска максимума

max = A[0]

maxi = 0

for i in 1..n-1

if A[i] > max

max = A[i]

maxi = i

* 1. Компьютер = вычислительное устройство + память RAM (по любому адресу обращаемся за фиксированное время)  
     Допустимые операции:

- операции над числами (арифметические операции, сравнения);

- операции с памятью (присваивание).

*Это – не полное описание модели вычислительного устройства. Будем уточнять по потребности.*

Упрощая, считаем: время работы алгоритма – суммарное количество выполненных операций.

* 1. Время работы алгоритма ***T*** зависит от размера входных данных ***N*** и от их конкретных значений.  
     Среди входных данных одного размера ***N*** рассматриваем (пока) худший случай (worst case scenario).   
     Зависимость T(N) оцениваем асимптотически, то есть нас интересует ее *порядок роста*: пренебрегаем константами и членами нестаршего порядка (пояснения ниже).

**2. «Наивный» алгоритм сортировки.**

Идея:

- найдем максимальный элемент и поместим его в конец (поменяем местами с последним элементом);

- в оставшемся массиве сделаем то же самое;

- будем повторять, пока все не отсортируем (не дойдем до массива длины 1)

**alg NaiveSort**

for n in N..2

**утв** Последние N-n элементов массива упорядочены по возрастанию

**утв** Каждый из первых n элементов меньше (не больше) каждого из N-n последних

**assert** Last N-n elements are incrementally ordered

**assert** Each of first n elements is less (not greater) than each of last N-n elements

max = A[0]

maxi = 0

for i in 1..n-1

if A[i] > max

max = A[i]

maxi = i

A[maxi] = A[n-1]

A[n-1] = max

Замечание 1. «утв» (англ. “assert”) – ключевое слово языка записи алгоритмов. После него пишется утверждение о данных при прохождении соответствующей строки.

Если утверждение о состоянии данных стоит в начале цикла, его обычно называют *инвариантом цикла.*

Утверждения о состоянии данных нужны для *доказательства корректности алгоритма.*

Их можно использовать при отладке программ.

Конец замечания.

Подсчитаем количество операций в худшем случае:

Замечание 2. Напомним: нас интересует *порядок роста* функции T(N). Это (в данном случае) записывают так: ***T(N) = O(N 2)***

Это означает (примерно), что *T(N)/(N2)* стремится к какой-то константе, когда Nстремится к бесконечности.

Говорят: «*T(N)* растет, как *N2*»

1. **Слияние упорядоченных массивов.**
   1. Можно ли сортировать быстрее? Вспомогательная задача: слияние упорядоченных массивов.

Дано: A[1..N], B[1..M] – массивы, упорядоченные по возрастанию.

Получить: С[1..N+M] – массив, содержащий все элементы массивов A и B, упорядоченные по возрастанию.

Идея 1. Сравниваем первый элемент массива B с последним элементом массива A

**array**  A[1..M], B[1..N]

**array**  C[1..M+N]

C[1..M] := A[1..M]

for k from 1 to N

**утв** Первые N+k-1 элементов массива C упорядочены по возрастанию

**утв** Они содержат все элементы массива А и k-1 первых элементов

массива B

**assert** First N+k-1 elements of the array C are incrementally

ordered.

**assert** They contain all elements of the array A and first k-1

elements of the array B

// find place p of B[k] in C

p := N+k

while B[k]<C[p] AND p > 0

p:=p-1

end\_while

DO: insert B[k] after C[p]

end\_for

Посмотрим на время.

Для начала будем считать только количество сравнений TComp

В худшем случае TComp(M, N) ~ N\*M

Если повезет: TComp(M, N) ~ N+M.

Замечание. Полное время включает не только сравнения, но и вставки элементов массива B в массив C. Вставка в середину массива – долгое дело. Нужно использовать не массивы, а другую *структуру данных –* списки. Отложим разговор о списках и других структурах данных и будем пока считать только сравнения.

* 1. **Что будет, если использовать схему «первый – первый», т.е. искать место для элементов массива B, начиная с начала массива A?**

Напишите алгоритм по этой идее.