

Сложность алгоритмов

1. Расположите следующие функции в порядке увеличения скорости роста при больших n :

- | | | |
|----------------|---------------------------|----------------|
| (a) $\log n$ | (f) $\log \log n$ | (k) n^n |
| (b) 1 | (g) $\sqrt{2}^{\log n}$ | (l) $n \log n$ |
| (c) \sqrt{n} | (h) $(\log(n))^{\log(n)}$ | (m) n^2 |
| (d) n | (i) 2^{2^n} | (n) 2^n |
| (e) 1.01^n | (j) $n!$ | |

2. Отметьте все функции, равные $\Theta(n^2)$ и все функции, равные $O(n^2)$

- | | |
|---------------------|--------------------------------|
| • $1000n^2$ | • $\frac{n^3}{1000} + 5000n^2$ |
| • e^n | • $\log(n^9 + n^5)$ |
| • $4n^2 + 10n + 50$ | • $n \log n$ |
| • $\log n$ | • $n^3/(1+n)$ |

3. Чему равна алгоритмическая сложность следующих операций?

- Поиск элемента в массиве размера N
- Добавление элемента в начало массива размера N
- Сортировка пузырьком массива размера N
- Быстрая сортировка массива размера N
- Добавление элемента в стек размера N
- Сложение матриц размера $N \times N$
- Простой алгоритм умножения матриц размера $N \times N$
- Следующий участок кода:

```
int sum = 0;
for (int i = 0; i < N; i++)
    for (int j = i+1; j < N; j++)
        for (int k = j+1; k < N; k++)
            sum++;
```

4. Алиса и Боб любят игры и соревнования. И сейчас они готовы приступить к новой игре. Всего у них есть X плиток шоколада. По правилам игры они могут есть этот шоколад по очереди (первой начинает Алиса). Известно, что Алиса съедает 7 плиток шоколада за ход, а Боб – 5 плиток шоколада. Выигрывает тот, кто съест последнюю плитку. При заданном X , определить победителя. Предложено 2 алгоритма решения этой задачи:

- Плохой: Вычитаем сначала 7, затем 5 и так до тех пор пока не дойдём до 0 (или отрицательного числа). Чему равна сложность данного решения?
- Хороший: Сначала находим остаток от деления X на 12. В зависимости от остатка определяем победителя. Чему равна сложность данного решения?

Сортировки

1. Создайте массив со следующими элементами: {163, 624, 7345, 545, 41, 78, 5, 536, 962, 1579}
2. **Печать массива:** Написать функцию `print_array(int n, int arr[])`
3. **Сортировка выбором:** Написать функцию сортировки выбором `void selection_sort(int n, int arr[])`. `arr` – массив чисел, которые нужно отсортировать, `n` – количество чисел в этом массиве. Будем обозначать подмассивы так: `arr[k:m]` – подмассив массива `arr` с элементами под номерами от `k` до `m`. Таким образом, весь массив можно обозначить как `arr[0:n-1]`.

Алгоритм сортировки выбором:

- Найти минимальный элемент в массиве.
 - Поменять местами минимальный элемент и первый элемент массива.
 - Повторить эти операции для подмассива `arr[1:n-1]`, затем для подмассива `arr[2:n-1]` и т.д.
4. **Рекурсивная сортировка выбором:** Написать рекурсивную функцию сортировки выбором `void rec_selection_sort(int start, int n, int arr[])`. `arr` – массив чисел, которые нужно отсортировать, `n` – количество чисел в массиве `arr`, `start` – начальный индекс подмассива в массиве `arr`.
 5. **Быстрая сортировка:** Написать рекурсивную функцию быстрой сортировки `void quick_sort(int arr[], int lo, int hi)`, которая будет сортировать подмассив `arr[lo:hi]` массива `arr`.

Алгоритм быстрой сортировки:

- Если `lo < hi`:
 - (a) Выбираем последний элемент подмассива в качестве опорного. Сохраняем это значение в переменной `pivot`.
 - (b) Вводим переменные-счётчики `i = lo`
 - (c) Бежим по массиву, используя новую переменную счётчик `j`, и, для каждого элемента, который не больше опорного, меняем `i`-й и `j`-й элементы. После каждого обмена увеличиваем `i` на 1.
 - (d) Меняем `i`-й элемент и опорный, так как мы хотим, чтобы опорный элемент разделял 2 подмассива.
 - (e) Рекурсивно вызываем функцию `quick_sort()` для каждого из подмассивов.

Стандартная функция `qsort()`

Пример использования функции `qsort()`:

```
#include <stdio.h>      /* printf */
#include <stdlib.h>     /* qsort */

int cmp(const void* a, const void* b)
{
    int* pa = (int*)a;
    int* pb = (int*)b;
    return ( *pa - *pb );
}

int main ()
{
    int arr[] = {163, 624, 7345, 545, 41, 78, 5, 536, 962, 1579};
    qsort(arr, 10, sizeof(int), cmp);

    print_array(10, arr);
}
```

Задачи:

1. Отсортировать числа массива `arr` по убыванию.
2. Отсортировать числа массива `arr` по возрастанию последней цифры.
3. Отсортировать числа массива `arr` по первой цифре числа.
4. Описать структуру `Movie` с полями `title`(название – строка), `year`(год выхода – целое число), `rating`(рейтинг на кинопоиске – вещественное число). Создать массив из 6-ти таких структур. Написать отдельную функцию для печати такого массива на экран.
5. Отсортировать массив фильмов по убыванию рейтинга.
6. Отсортировать массив фильмов по возрастанию года выхода.
7. Интерпретируем значения массива `arr`, как значения некоторых углов в градусах. Отсортировать числа по возрастанию косинусов соответствующих углов.
8. Отсортировать числа массива `arr` следующим образом: сначала четные числа по возрастанию, затем нечётные по убыванию.