Алгоритмы «разделяй и властвуй»

```
Function BSearch(A, x, l, r):
    if l==r then
        if x == A[l] then
            return l
        else
            return Nil
        end
    end
    if x > A[⌊(l+r)/2⌋] then
        BSearch(A, x, ⌈(l+r)/2⌉, r)
    else
        BSearch(A, x, l, ⌊(l+r)/2⌋)
    end
end
```

```
Function HW(n):
    if n < 2020 then
        for i = 1 to n do
            print("HW!");
        end
    else
        HW(⌊n/3⌋);
        print("HW!");
        HW(⌊n/3⌋);
        for i = 1 to 2020
         do
            print("HW!");
        end
    end
end
```

1. Постройте дерево рекурсии для алгоритма `BSearch`, который ищет вхождение числа x в отсортированный массив A, и оцените его временную сложность. Считайте, что арифметические операции и операции сравнения стоят $O(1)$.

2. Определим $f(n)$ как количество выводов «HW!» функцией `HW`(n). Оцените асимптотику роста $f(n)$.

Если не оговоренно противного, то в рамках курса считается, что $T(n) = \Theta(1)$ при малых n.

3. Найти асимптотическую оценку функции $T(n)$, воспользовавшись основной теоремой о рекурсии:

а) $T(n) = 3T(\frac{n}{3}) + cn$; **б)** $T(n) = 8T(\frac{n}{2}) + cn^2$;

в) $T(n) = 8T(\frac{n}{2}) + cn^4$.

4. Найти асимптотическую оценку $T(n)$, используя деревья рекурсии:

а) $T(n) = T(\lfloor \frac{n}{3} \rfloor) + T(\lceil \frac{2n}{3} \rceil) + cn$; **б)** $T(n) = 4T(\lfloor \frac{n}{2} \rfloor) + cn^2 \log n$.

5. Элемент массива $A[1, .., n]$ называется *majority element*, если встречается в массиве A не меньше $\lceil \frac{n}{2} \rceil$ раз. Постройте алгоритм «разделяй и властвуй», который находит majority element в массиве за $O(n \log n)$, если он есть. Операции сравнения элементов как чисел запрещены (представьте, что вы имеете дело с массивом картинок); вы можете только проверять условия вида $A[i] == A[j]$.

6. Постройте итеративную версию алгоритма сортировки слиянием (MergeSort).

7. Предположим, удалось установить, что любое число можно возвести в квадрат за $O(n)$, где n – длина числа в двоичной записи. Докажите, что тогда любые два числа можно перемножать за $O(n)$, где n – длина максимального из чисел в двоичной записи.