Алгоритмы «разделяй и властвуй»

```
1  Function BSearch(A, x, l, r):
2  |  if l==r then
3  |  |  if x == A[l] then
4  |  |  |  return l
5  |  |  else
6  |  |  |  return Nil
7  |  |  end
8  |  end
9  |  if x > A[⌊(l+r)/2⌋] then
10 |  |  BSearch(A, x, ⌈(l+r)/2⌉, r)
11 |  else
12 |  |  BSearch(A, x, l, ⌊(l+r)/2⌋)
13 |  end
14 end
```

```
1  Function HelloWorld(n):
2  |  if n < 2020 then
3  |  |  for i = 1 to n do
4  |  |  |  print("Hello, World!");
5  |  |  end
6  |  else
7  |  |  HelloWorld(⌊n/3⌋);
8  |  |  print("Hello, World!");
9  |  |  HelloWorld(⌊n/3⌋);
10 |  |  for i = 1 to 2020 do
11 |  |  |  print("Hello, World!");
12 |  |  end
13 |  end
14 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 «Hello, World!» функцией `HelloWorld`(n). Оцените асимптотику роста $f(n)$.

Если в рекуррентном соотношении $T(n) = aT(\frac{n}{b}) + f(n)$ не указано округление, то можно считать, что n — степень b. При малых n считается, что $T(n)$ — константа.

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. Постройте алгоритм, который находит в массиве a отрезок $a[l, \ldots, r]$, $l < r$ с максимальной метрикой $a[l] + a[r] + \sum_{i=l}^{r} a[i]$.

В случае, когда в рекуррентных соотношениях возникают округления, задачу можно решать считая, что их нет, т. е. считать, что на вход $T(n)$ могут подаваться дробные параметры. Это облегчит задачу. Однако для того, чтобы лучше разобраться в теме, рекомендуем решать задачи с учётом округлений.

5. 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)** $T(n) = 4T(\lfloor \frac{n}{2} \rfloor) + cn^2 \log n$.

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

7. 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 $A[i] == A[j]$.