Разделяй и властвуй

(Divide & Conquer)

1. Задача разбивается на подзадачи меньшего размера
2. Подзадачи решаются рекурсивно
3. Ответ задачи строится из ответов на подзадачи.

- Умножение n-битовых чисел.

- Трюк Гаусса

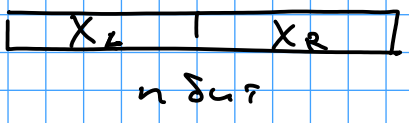
$$(a+ib)\cdot(c+id) = \underline{ac} - \underline{bd} + i(\underline{ad+cb})$$

$$(a+b)\cdot(c+d) = ac + (bc+ad) + bd$$

$$\underline{bc+ad} = (a+b)\cdot(c+d) - \underline{ac} - \underline{bd}$$

- $X \cdot Y$, x, y - n-битовые числа

$$X = X_L \cdot 2^{n/2} + X_R$$

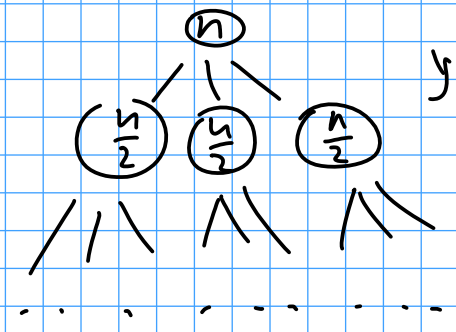


$$Y = Y_L \cdot 2^{n/2} + Y_R$$

$$(X_L \cdot 2^{n/2} + X_R)\cdot(Y_L \cdot 2^{n/2} + Y_R) =$$
$$\underline{X_L Y_L} \cdot 2^n + \underline{X_R \cdot Y_R} + 2^{n/2}(\underline{X_L \cdot Y_R + X_R \cdot Y_L})$$

$$(X_L + X_R)\cdot(Y_L + Y_R) = X_L Y_L + X_R \cdot Y_R + (X_R \cdot Y_L + X_L \cdot Y_R)$$
$$X_R \cdot Y_L + X_L \cdot Y_R = \underline{(X_L + X_R)(Y_L + Y_R)} - \underline{X_L Y_L} - \underline{X_R Y_R}$$

$$T(n) = 3 \cdot T(\lceil \tfrac{n}{2} \rceil) + \underline{O(n)}$$



уровень k: $3^k \cdot O\left(\left\lceil \frac{n}{2^k} \right\rceil\right) =$

$\left(\frac{3}{2}\right)^k \cdot O(n)$

$T(n) = \sum_{k=0}^{\log_2 n} \left(\frac{3}{2}\right)^k \cdot O(n) =$

Алгоритм Карацубы

$= O(n) \cdot \sum_{k=0}^{\log_3 n} \left(\frac{3}{2}\right)^k \leq$

$\leq O(n) \cdot \left(\frac{3}{2}\right)^{\log_2 n} =$

$= O\left(n \cdot \frac{3^{\log_2 n}}{n}\right) =$

$= O\left(3^{\log_2 n}\right) = O\left(n^{\log_2 3}\right)$

$a^{\log_b c} = c^{\log_b a}$

$\Rightarrow O(n^{1.59})$ алгоритм

Без трюка Гаусса:

$T(n) = 4\, T\left(\left\lceil \frac{n}{2} \right\rceil\right) + O(n)$

k уровень: $2^k \cdot O(n)$

$k = \log_2 n$: $2^{\log_2 n} O(n) = O(n^2)$

Основная теорема о рекуррентных соотношениях

(Master theorem)

$$T(n) = a \cdot T\left(\left\lceil \frac{n}{b} \right\rceil\right) + O(n^d)$$

$$T(n) \begin{cases} 1.\ O(n^d), & d > \log_b a \quad b^d > a \\ 2.\ O(n^{\log_b a}), & d < \log_b a \quad b^d < a \\ 3.\ O(n^d \cdot \log n), & d = \log_b a \quad b^d = a \end{cases}$$

▷

◁ k уровень:

$$a^k \cdot O\left(\left|\frac{n}{b^k}\right|^d\right) \quad | \; n - \text{степень } b$$

$$\left(\frac{a}{b^d}\right)^k \cdot O(n^d)$$

$$T(n) = \sum_{k=0}^{\log_b n} \left(\frac{a}{b^d}\right)^k \cdot O(n^d) = O(n^d) \cdot \sum_{k=0}^{\log_b n} \left(\frac{a}{b^d}\right)^k$$

1. $\frac{a}{b^d} < 1 \qquad a < b^d \qquad \log_b a < d$

$$T(n) = O(n^d) \cdot C = O(n^d)$$

2. $\frac{a}{b^d} > 1 \qquad \log_b a > d$

$$T(n) = O(n^d) \cdot \left(\frac{a}{b^d}\right)^{\log_b n} = O(n^d) \cdot O\left(\frac{n^{\log_b a}}{n^d}\right) =$$

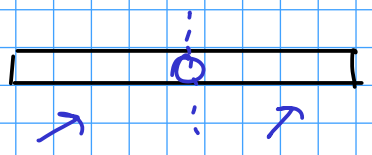
$$= O(n^{\log_b a})$$

3. $\frac{a}{b^d} = 1 \qquad \log_b a = d$

$$T(n) = O(n^d)(\log_b n + 1) = O(n^d \cdot \log n)$$

◁

Двоичный поиск



```
BinarySearch(A[1..n], i, j, x):
    m = (j+i)/2 = i + (j-i)/2

    if j - i < 2:
        .....
    if A[m] > x:
        return BinarySearch(A, i, m-1, x)
    else
        return BinarySearch(A, m, j, x)
```

$T(n) = T(\lceil \frac{n}{2} \rceil) + O(1)$ $\quad a = 1,\ b = 2,\ d = 0$

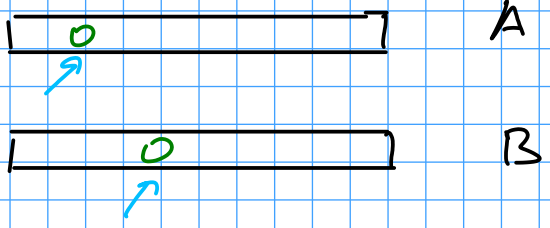
$\log_2 1 = 0 = d$

$T(n) = O(\log n)$

Сортировка слиянием

Merge Sort

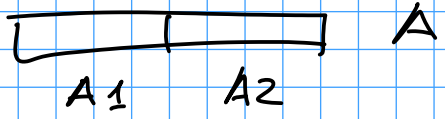
```
Merge(A, B):
    C ← Array(A.size() + B.size())
    i = 1, j = 1
    while i ≤ A.size() and j ≤ B.size():
        if A[i] < B[j]:
            C[i + j - 1] = A[i]
            i = i + 1
        else
            C[i + j - 1] = B[j]
            j = j + 1
```



```
while i ≤ A.size():
    C[i+j-1] = A[i]
    i = i+1
while j ≤ B.size():
    C[i+j-1] = B[j]
    j = j+1
```

$O(|A|+|B|)$

```
MergeSort(A)
    if A.size() < 2:
        return A
    A1, A2 = Split(A)
    A1 = MergeSort(A1)
    A2 = MergeSort(A2)
    return Merge(A1, A2)
```



$$T(n) = 2T(\lceil \tfrac{n}{2} \rceil) + \underbrace{O(n)}_{\text{Merge}}$$

$a = 2$, $b = 2$, $d = 1$

$\log_b a = 1 = d$

$O(n \cdot \log n)$

```
NRMergeSort(A)
    Q = Queue
    for i = 1 to A.size():
        Q.enqueue(Array(A[i]))
    while Q.size() > 1:
        A1 = Q.dequeue()
```

A_2 = Q.dequeue()

Q.enqueue(Merge(A_1, A_2))

return Q.dequeue()