Разделяй и властвуй (Divide & Conquer)

1. Разбить на подзадачи
2. Решаем подзадачи рекурсивно
3. „Склеиваем" результаты подзадач ⇒ решение исходной

Алгоритм Карацубы

„Трюк Гаусса"

$$(a+bi)(c+di) = \underline{ac} - \underline{bd} + i(\underline{ad+bc})$$

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

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

1 2 3

Пусть X и Y — n-битовые числа

$$X = X_u \cdot 2^{n/2} + X_L$$
$$Y = Y_u \cdot 2^{n/2} + Y_L$$

X_u, Y_u, X_L, Y_L — $n/2$ битовые числа

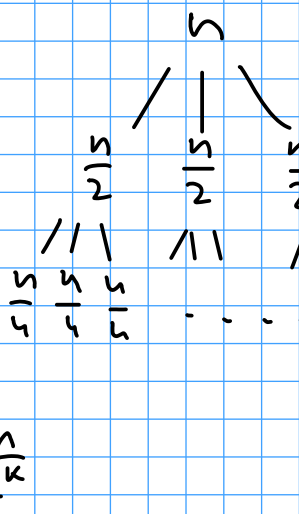
$$X \cdot Y = \underline{X_u \cdot Y_u}^{1} \cdot 2^n + \underline{X_L \cdot Y_L}^{2} + 2^{n/2}(\underline{X_u Y_L + X_L \cdot Y_u}^{3})$$

$$(X_u + X_L)\cdot(Y_u + Y_L) = \underline{X_u \cdot Y_u} + \underline{X_L \cdot Y_L} + (\underline{X_u Y_L + X_L Y_u})$$

$$(X_u \cdot Y_L + X_L \cdot Y_u) = (X_u + X_L)(Y_u + Y_L) - X_u Y_u - X_L Y_L$$

3

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



На уровне k 3^k вершин (подзадач) размера $n/2^k$

Т.е. $O(n/2^k) \cdot 3^k$

Всего: $T(n) = \sum_{k=0}^{\log_2 n} O(n/2^k) \cdot 3^k =$

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

$$= O\left(\cancel{n} \cdot \frac{3^{\log_2 n}}{2^{\log_2 n}}\right) = O(n^{\log_2 3}) \qquad a^{\log_b c} = c^{\log_b a}$$

n

$$\approx O(n^{1.59})$$

Без трика Карацубы:

$$T(n) = 4 \cdot T\left(\frac{n}{2}\right) + \Theta(n)$$

На уровне k: $4^k \cdot \Theta(n/2^k)$

Всего: $T(n) = \sum_{k=0}^{\log_2 n} 4^k \Theta(n/2^k) = \Theta(n) \cdot \sum_{k=0}^{\log_2 n} 2^k =$

$$= \Theta(n^2)$$

Основная теорема (Master theorem) о рекуррентных соотношениях

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

$a, b \geq 1 \qquad d \geq 0$

1. $\log_b a > d \qquad T(n) = O(n^{\log_b a})$
2. $\log_b a < d \qquad T(n) = O(n^d)$
3. $\log_b a = d \qquad T(n) = O(n^d \cdot \log n)$

△ k-ый уровень рекурсии:

a^k подзадач размера n/b^k

операций на ур-не k: $a^k \cdot O\left(\left(\frac{n}{b^k}\right)^d\right)$

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

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

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

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

n^d

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

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

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

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

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

Двоичный поиск (binary search)

```
BinSearch(A, i, j, v) // найти v в A[i:j]
    if j - i ≤ 1:
        ....
    m = i + (j - i)/2
    if A[m] < v:
        return BinSearch(A, i, m, v)
    else
        return BinSearch(A, m, j, v)
```

$$T(n) = T(\lceil \tfrac{n}{2} \rceil) + O(1)$$

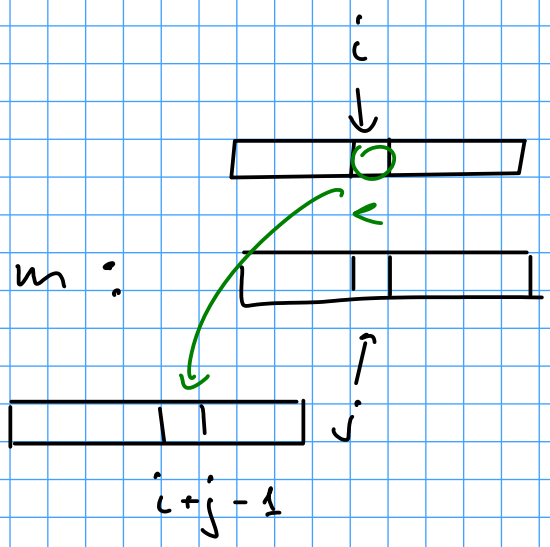
$a = 1 \quad b = 2 \quad d = 0$

$\log_b a = 0 = d \Rightarrow O(n^d \cdot \log n) = O(\log n)$

Сортировка слиянием (Merge Sort)

```
MergeSort(A[n]):
    if n ≤ 1:
        return A
    B ← MergeSort(A[1 : n/2])
    C ← MergeSort(A[n/2+1 : n])
    return Merge(B, C)
```

```
Merge(B[n], C[m]):
    i = j = 1
    D[n+m]
    while i ≤ n or j ≤ m:
        if B[i] < C[j]:
            D[i+j-1] = B[i]
            i = i + 1
        else
            D[i+j-1] = C[j]
            j = j + 1
    while i ≤ n:
        D[i+j-1] = B[i]
        i = i + 1
    while j ≤ m:
        D[i+j-1] = C[j]
        j = j + 1
```



$O(m+n)$

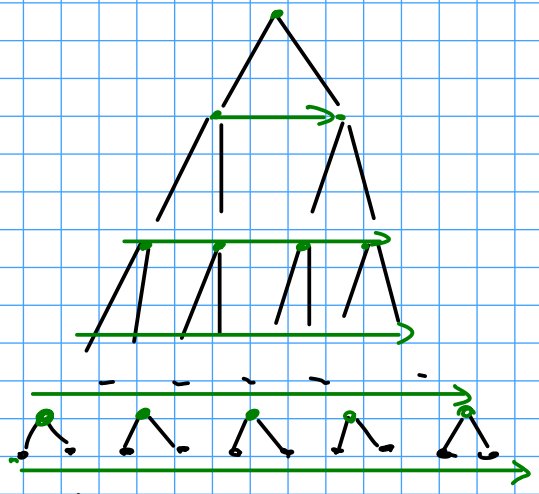
return D

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

$a = 2 \quad b = 2 \quad d = 1 \qquad \log_2 2 = 1 = d$

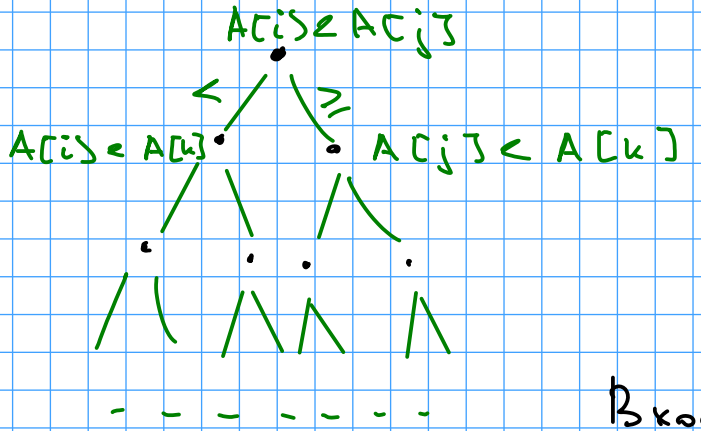
$$O(n^d \cdot \log n) = O(n \log n)$$

```
NR Merge Sort (A)
    Q ← queue
    for i = 1 to n:
        Q.enqueue ( [A[i]] )
    while Q.size() > 1:
        B = Q.dequeue()
        C = Q.dequeue()
        Q.enqueue ( Merge (B, C))
    return Q.dequeue()
```



Сортировки сравнением

Деревья сравнений



Время работы = глубина дерева

Листьев у двоичного дерева глубины $H \le 2^H$

Вход: массив длины n
∃ все эл-ты различны

Для каждой перестановки исходного массива должен быть свой лист

$2^H \geq$ # листьев $\geq n!$

$$H \geq \log_2 n! = \log_2 1 + \log_2 2 + \ldots + \log_2 n \geq$$

$$\geq \frac{n}{2} \cdot \log_2 n/2 = \Omega(n \log n)$$