Розробка та аналіз алгоритмів

5. Швидке сортування

- Алгоритм швидкого сортування
- Ефективність алгоритму
- Випадкове швидке сортування
- Аналіз алгоритму

Властивості швидкого сортування

- Час роботи варіюється
  - В найгіршому випадку: $\Theta(n^2)$
  - В середньому випадку: $\Theta(n \lg n)$
- Не потребує додаткової пам'яті для сортування
- Елегантна структура алгоритму

Метод декомпозиції для алгоритму

- Розділення
- Рекурсивний розв'язок
- Комбінування

Основна робота алгоритмів

	Швидке сортування
Сортування злиттям	

Вхідний масив: $A = [2, 8, 7, 1, 3, 5, 6, 4]$

$x = 4$

Після процедури розбиття: $\underbrace{|2|1|3|}_{\le 4}\,4\,\underbrace{|7|5|6|8|}_{>4}$

Після сортування: $|1|2|3|4|5|6|7|8|$

Властивості процедури розбиття:

1) не використовує додаткової пам'яті
2) працює за час $O(n)$

Алгоритм швидкого сортування

// На основі методу декомпозиції

Вхід: масив A, індекси p, r, які визначають підмасив для сортування $A[p...r]$

Вихід: відсортований підмасив $A[p...r]$

1) Розбити масив $A[p..r]$ на дві частини: $A[p..q-1]$ (де всі елементи $\le A[q]$) та $A[q+1..r]$ (де всі елементи $> A[q]$). Елемент $A[q]$ – опорний

2) Рекурсивний розв'язок задачі для $A[p..q-1]$ та $A[q+1..r]$

3) Комбінування – не потрібно

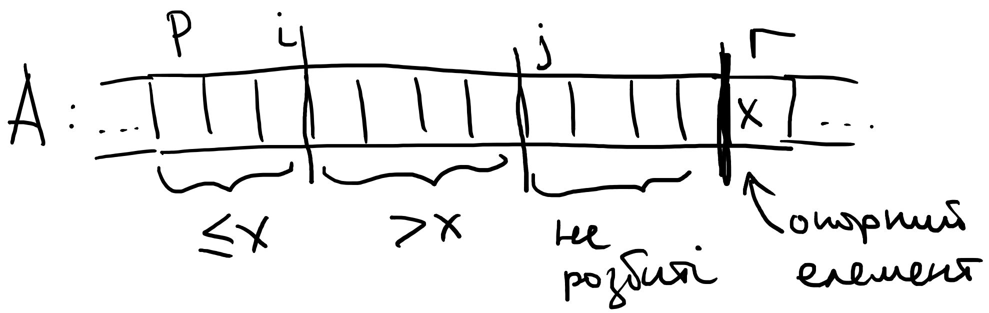
QuickSort(A, p, r)

1. if p<r then:
2. q = Partition(A, p, r)
3. QuickSort(A, p, q-1)
4. QuickSort(A, q+1, r)

початковий запуск:

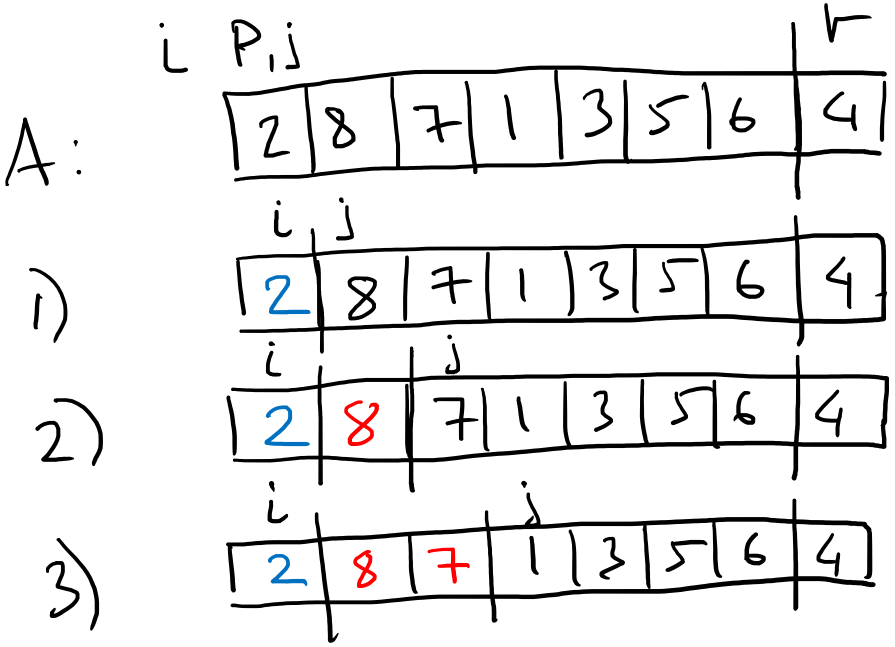
QuickSort(A, 1, n)

Приклад процедури розбиття



1) $A[r] = x$ – опорний елемент
2) $A[p \dots i] \le x$
3) $\underbrace{A[i+1 \dots j-1]} > x$
4) $A[j \dots r-1]$ – ще не розглянуті елементи

$i = p-1$, $j = p$, $x = A[r] = 4$

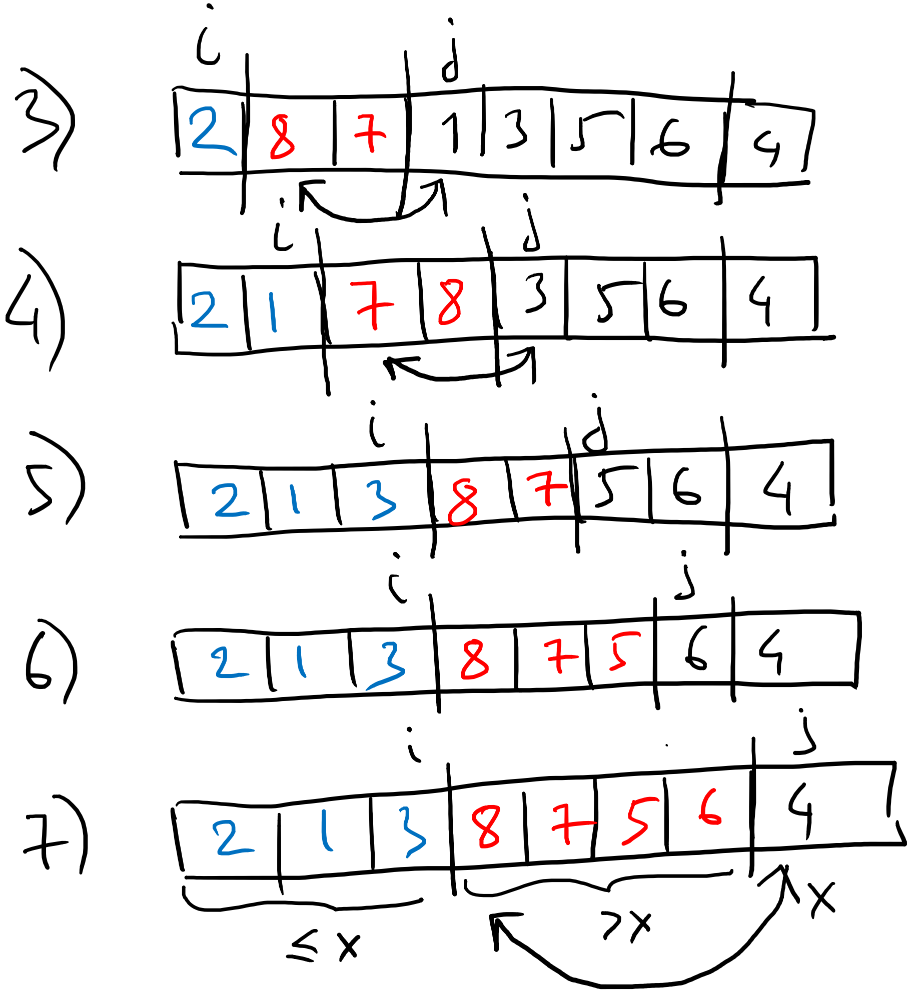


$A[j] = 2 < 4 = x \Rightarrow i = i+1, \; j = j+1$

$A[j] = 8 > 4 = x \Rightarrow j = j+1$

$A[j] = 7 > 4 = x \Rightarrow j = j+1$

Приклад процедури розбиття

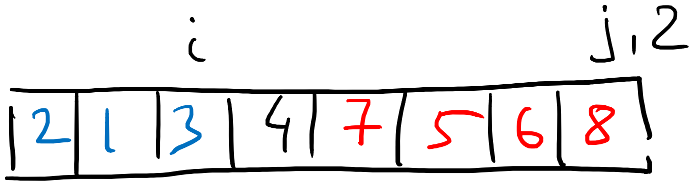


$A[j]=1<4=x \Rightarrow i=i+1,\ j=j+1$

$A[j]=3<4=x \Rightarrow i=i+1,\ j=j+1$

$A[j]=5>4 \Rightarrow j=j+1$

$A[j]=6>4 \Rightarrow j=j+1$



$A[r] \leftrightarrow A[i+1]$

Процедура розбиття

```
Partition(A, p, r)
x = A[r]
i = p - 1
for j = p to r-1:
    if A[j] ≤ x then:
        i = i + 1
        A[i] <-> A[j]
A[r] <-> A[i+1]
return i+1    // повернути індекс опорного елементу
```

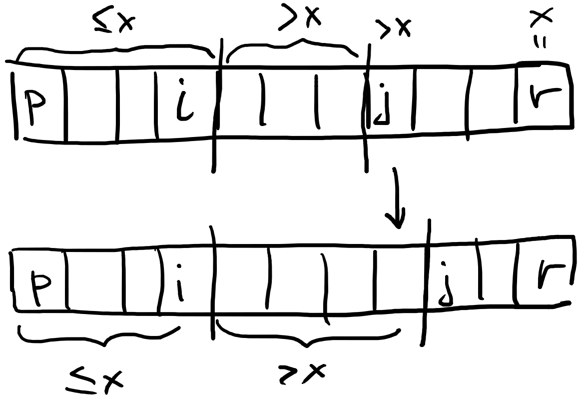
Час роботи: $O(n)$

Пам'ять: $\Theta(1)$

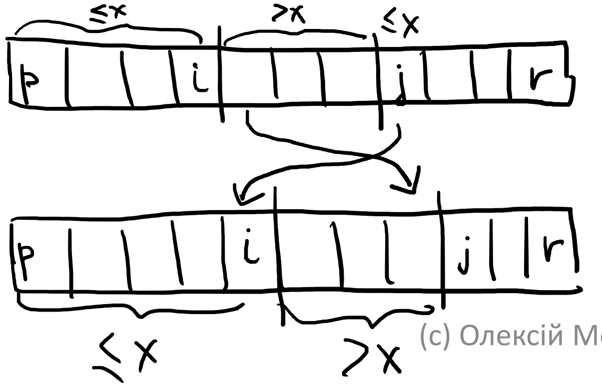
Коректність процедури розбиття

1) $A[p \dots i] \le x$ 2) $A[i+1 \dots j-1] > x$

<u>Випадок №1</u> Поточний елемент $A[j] > x$



<u>Випадок №2</u> Поточний елемент $A[j] \le x$

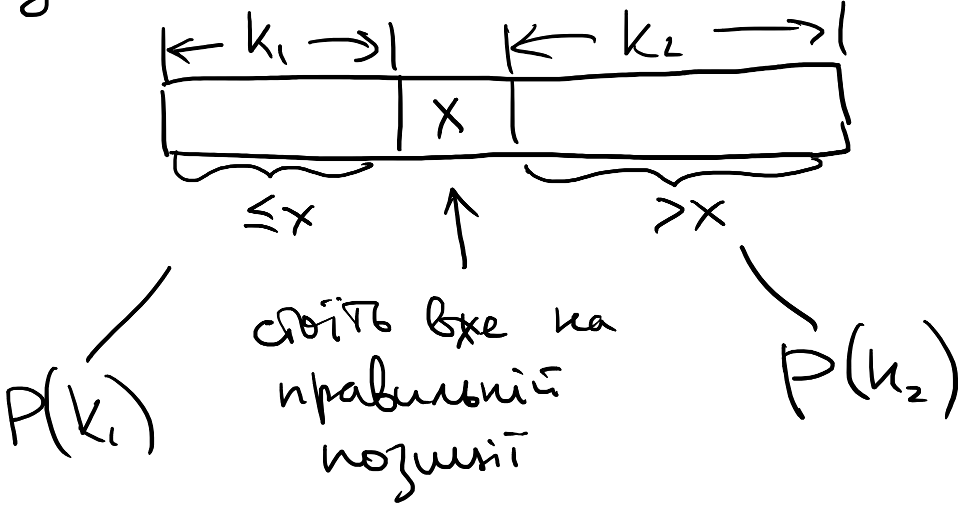


Коректність алгоритму (на основі індукції)

Маємо твердження $P(n)$ = "алгоритм швидкого сортування коректно сортує масив довжиною n"

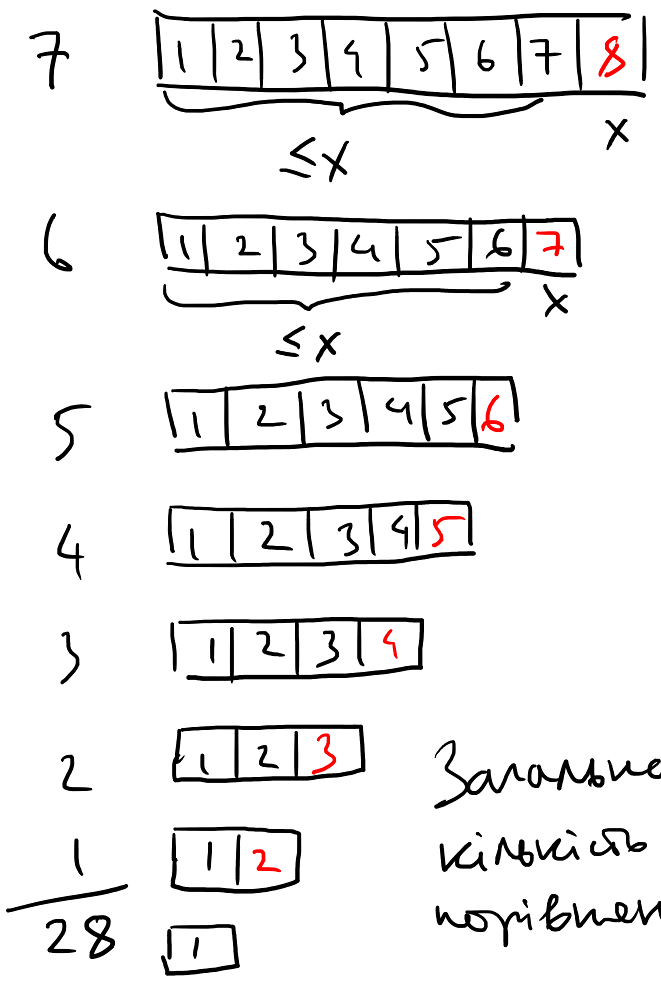
1) Базис індукції. $P(1)$ – виконується

2) Крок індукції. Якщо для всіх $k<n$: $P(k)$, то також виконується $P(n)$ (при $n \geq 2$)



$k_1, k_2 < n$

Ефективність алгоритму. Найгірший випадок



Загальна кількість порівнянь: 28

$$T(n) = T(n-1) + T(0) + \Theta(n)$$

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

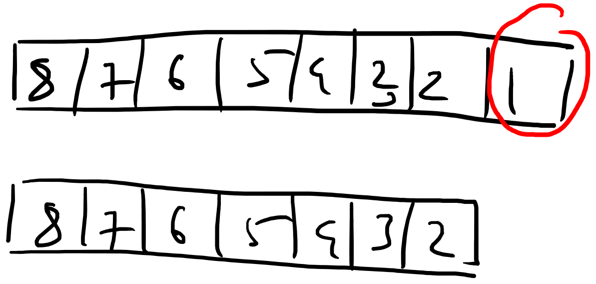
$$c \cdot n^2 = c \cdot (n-1)^2 + c_1 \cdot n$$

$$\cancel{cn^2} = \cancel{cn^2} - 2cn + c + c_1 n$$

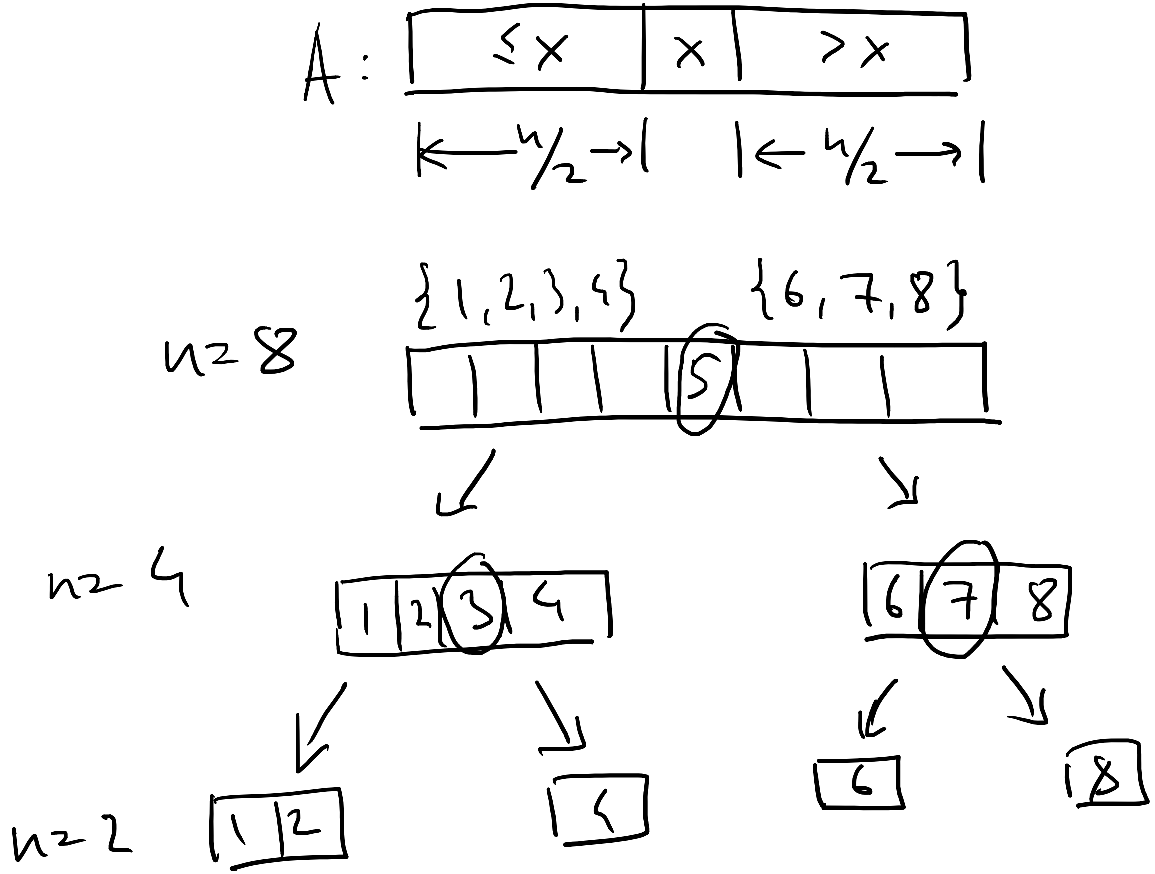
$$c(2n-1) = c_1 \cdot n$$

$$c = \frac{c_1 n}{2n-1}$$

$$c \approx \frac{c_1}{2}$$



Ефективність алгоритму. Найкращий випадок



$T(n)=2\cdot T(n/2)+\Theta(n)$

$T(n)=\Theta(n \lg n)$

Кількість порівнянь

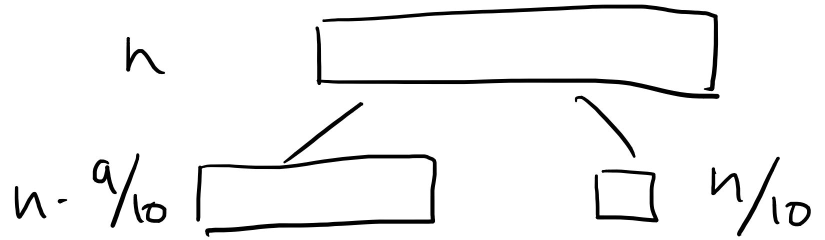
7

$3+2=5$

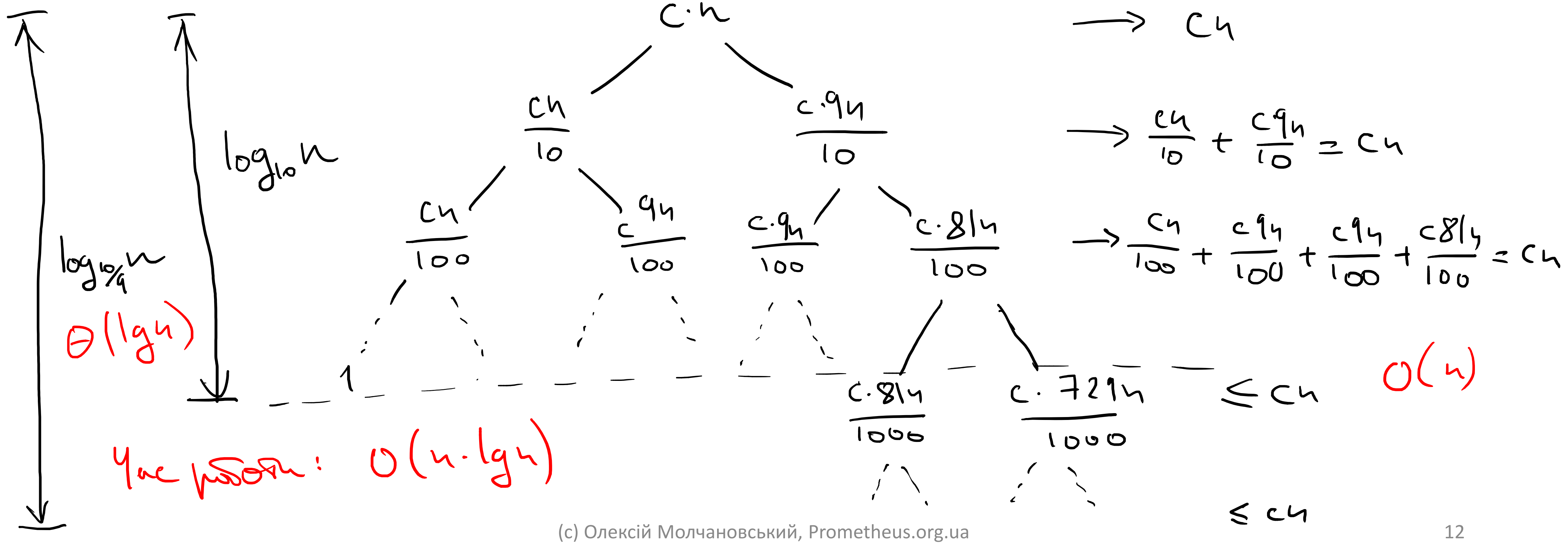
1

13 – кількість порівнянь

Ефективність алгоритму. Розбалансований випадок



$$T(n) = T(9n/10) + T(n/10) + \Theta(n) \leq$$
$$\leq T(9n/10) + T(n/10) + cn$$



Випадкове швидке сортування

RandomizedPartition(A, p, r)

1. i = random(p, r)
2. A[i] ⟷ A[r]
3. return Partition(A, p, r)

Середній час роботи:

$\Theta(n \cdot \lg n)$

RandomizedQuickSort(A, p, r)

1. if p<r then:
2. q = RandomizedPartition(A, p, r)
3. RandomizedQuickSort(A, p, q-1)
4. RandomizedQuickSort(A, q+1, r)

Аналіз випадкової версії алгоритму

X – загальна кількість порівнянь елементів під час роботи алгоритму.

$E[X]$ – ?

$A: \boxed{2 \mid 5 \mid 1 \mid 3}$

$z_2 \; z_4 \; z_1 \; z_3$

z_i – i-й найменший елемент масиву A

X_{ij} – кількість порівнянь елементів z_i та z_j

Питання: чому може дорівнювати X_{ij}? Відповідь: 0 або 1.

z_i або z_j були опорними.

$$X_{ij} = \begin{cases} 1, & \text{якщо } z_i \text{ та } z_j \text{ порівнювались} \\ 0, & \text{в іншому випадку} \end{cases}$$

Аналіз випадкової версії алгоритму

$$E[X] = E\left[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E[X_{ij}] = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \underbrace{P\{z_i \text{ порівнювалось з } z_j\}}$$

лінійність мат. сподівання

$$E[X_{ij}] = 0 \cdot P\{X_{ij}=0\} + 1 \cdot P\{X_{ij}=1\} = P\{X_{ij}=1\}$$

Аналіз випадкової версії алгоритму

$A = \{1, \dots, 10\}$ $x = 7$

$L = \{1, .., 6\}$ $R = \{8, .., 10\}$

$z_i < x < z_j$

$Z_{ij} = \{z_i, z_{i+1}, .., z_{j-1}, z_j\}$

$Z_{1,6} = \{1, 2, 3, 4, 5, 6\}$ $Z_{8,10} = \{8, 9, 10\}$

Питання: за якої умови будуть порівнюватись елементи множини Z_{ij}?

$Z_{1,10}$

$x = 7$

$Z_{1,6}$ $Z_{8,0}$

... ...

Відповідь: z_i та z_j порівнюються між собою, коли один з них обраний опорним серед елементів Z_{ij}.

$P\{z_i$ порівнюється з $z_j\} = P\{$опорним в Z_{ij} обрано або z_i, або $z_j\} =$

$= P\{$опорним в Z_{ij} обрано $z_i\} + P\{$опорним в Z_{ij} обрано $z_j\}$

$= \left[|Z_{ij}| = j-i+1\right] = \frac{1}{j-i+1} + \frac{1}{j-i+1} = \frac{2}{j-i+1}.$

Аналіз випадкової версії алгоритму

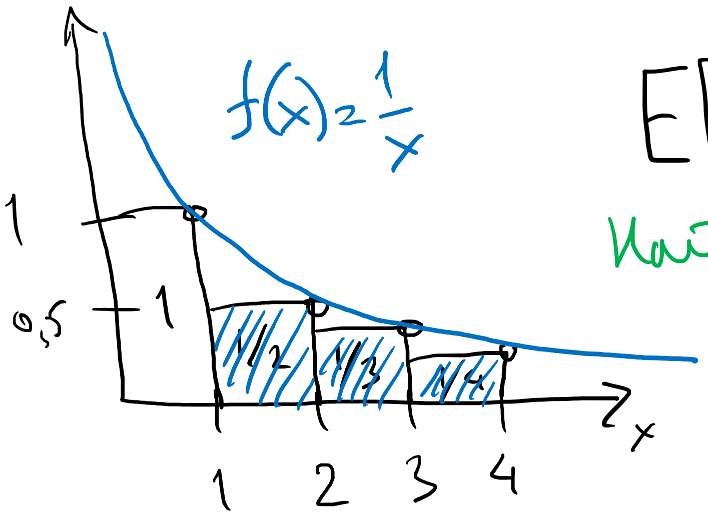
$$E[X] = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} P\{z_i \text{ порівнюється з } z_j\} = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{2}{j-i+1} = 2\cdot\sum_{i=1}^{n-1}\left(\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n-i+1}\right) =$$

$$\parallel$$

$$\frac{2}{j-i+1}$$

$$= 2\cdot\sum_{i=1}^{n-1}\sum_{k=2}^{n-i+1}\frac{1}{k} < 2n\cdot\sum_{k=2}^{n}\frac{1}{k} = 2n\cdot\left(\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n}\right)$$

гармонічний ряд



$$E[X] < 2n\cdot\lg n = O(n\cdot\lg n) \rightarrow \boxed{T(n) = \Theta(n\cdot\lg n)}$$

Найкращий випадок: $\Theta(n\cdot\lg n)$

$$\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n} = \sum_{k=2}^{n}\frac{1}{k} < \int_{1}^{n}\frac{1}{x}\,dx = \ln x\Big|_{1}^{n} = \ln n - \cancel{\ln 1}^{\,0} = \ln n$$