

Алгоритмы сортировки

Задача

На вход алгоритма подаётся последовательность n элементов

$$a_1, a_2, \dots, a_n.$$

На выходе алгоритм должен вернуть перестановку исходной последовательности

$$a'_1, a'_2, \dots, a'_n,$$

чтобы выполнялось следующее соотношение

$$a'_1 \leq a'_2 \leq \dots \leq a'_n.$$

Сортировка пузырьком (bubble sort)

Пример

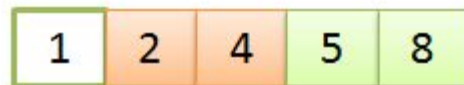
$i = 1$



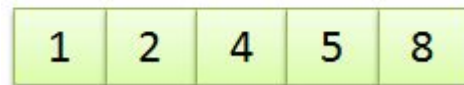
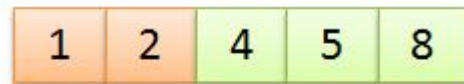
$i = 2$



$i = 3$



$i = 4$



Алгоритм

for i = 1 to n-1

 for j = 1 to n-i

 if $A[j] > A[j+1]$ then

 temp = A[j]

 A[j] = A[j+1]

 A[j+1] = temp

Сложность

for i = 1 to n-1

 for j = 1 to n-i

 if A[j] > A[j+1] then

 temp = A[j]

 A[j] = A[j+1]

 A[j+1] = temp

Количество операций

n

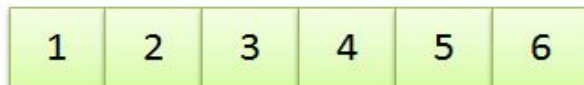
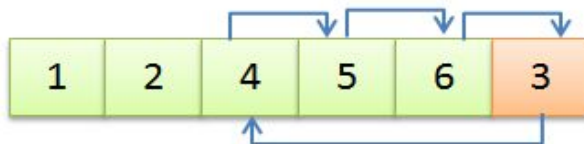
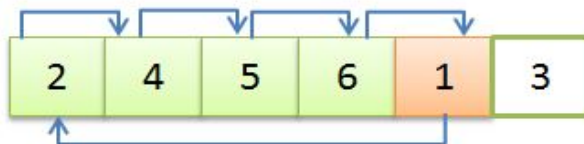
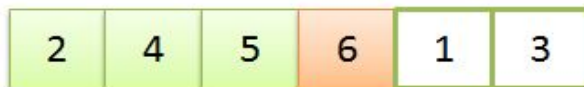
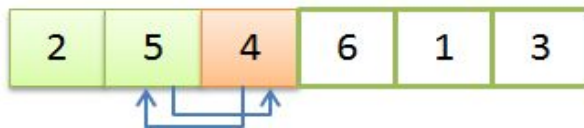
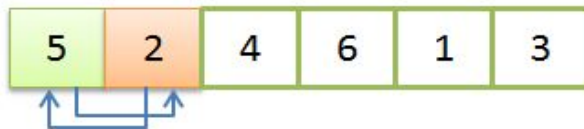
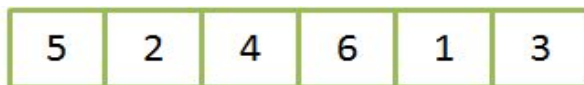
$$\sum_1^{n-1} (n-i+1) = \frac{(n+2)(n-1)}{2} = \frac{n^2 + n - 2}{2}$$

$$\sum_1^{n-1} (n-i) = \frac{(n-1+1)(n-1)}{2} = \frac{n^2 - n}{2}$$

$$\Rightarrow T(n) = O(n^2).$$

Сортировка вставками (insertion sort)

Пример



Алгоритм

for $j = 2$ to n

key = $A[j]$

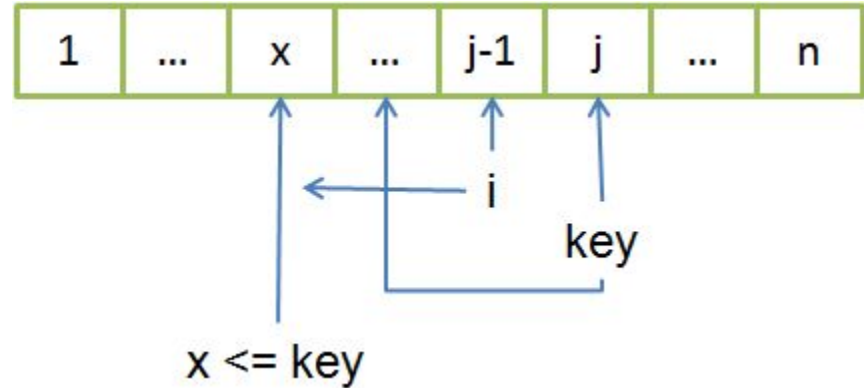
$i = j - 1$

while $i > 0$ and $A[i] > \text{key}$

$A[i+1] = A[i]$

$i = i - 1$

$A[i+1] = \text{key}$



СЛОЖНОСТЬ

for j = 2 to n

key = A[j]

i = j - 1

while i > 0 and A[i] > key

A[i+1] = A[i]

i = i - 1

A[i+1] = key

Количество операций

n

n-1

n-1

$\sum_2^n t_j$

$\sum_2^n (t_j - 1)$

$\sum_2^n (t_j - 1)$
n-1

Сложность

Лучший случай: массив отсортирован по возрастанию, тогда $t_j = 1$,

$$\sum_2^n t_j = n - 1, \sum_2^n (t_j - 1) = 0.$$

$$T(n) = O(n).$$

Худший случай: массив отсортирован по убыванию, тогда $t_j = j$,

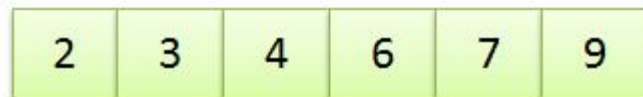
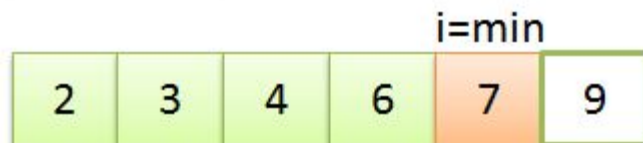
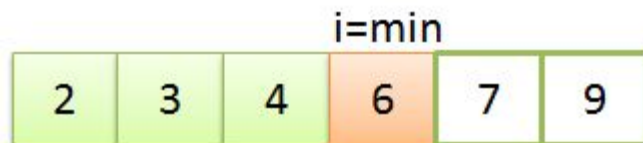
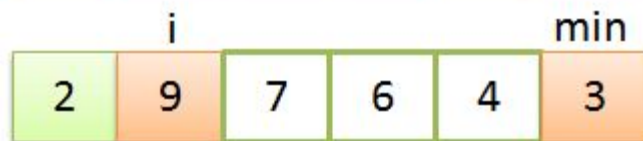
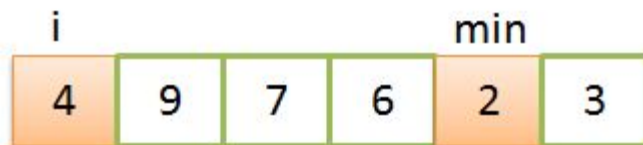
$$\sum_2^n t_j = \sum_2^n j = \frac{(2+n)(n-1)}{2} = \frac{n^2 + n - 2}{2},$$

$$\sum_2^n (t_j - 1) = \sum_2^n (j - 1) = \frac{(1+n-1)(n-1)}{2} = \frac{n^2 - n}{2}.$$

$$T(n) = O(n^2).$$

Сортировка выбором (selection sort)

Пример



Алгоритм

for $i = 1$ to $n-1$ do

$min = i$

 for $j = i+1$ to n do

 if $A[min] > A[j]$ then

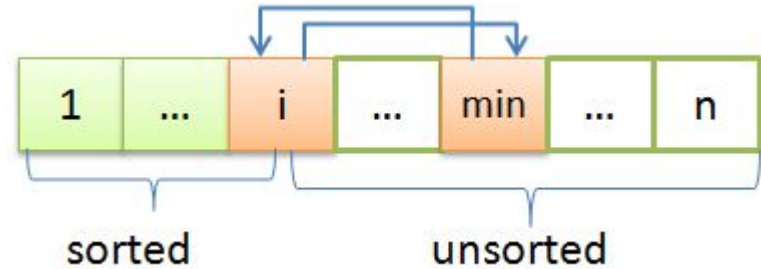
$min = j$

 if $min \neq i$ then

$temp = a[i]$

$a[i] = a[min]$

$a[min] = temp$



СЛОЖНОСТЬ

```
for i = 1 to n-1 do
  min = i
  for j = i+1 to n do
    if A[min] > A[j] then
      min = j
  if min <> i then
    temp = a[i]
    a[i] = a[min]
    a[min] = temp
```

Количество операций

n

n-1

$$\sum_{i=1}^{n-1} (n-i+1) = \frac{(n+2)(n-1)}{2} = \frac{n^2 + n - 2}{2}$$

$$\sum_{i=1}^{n-1} (n-i) = \frac{(n-1+1)(n-1)}{2} = \frac{n^2 - n}{2}$$

n-1

$$\Rightarrow T(n) = O(n^2).$$

Быстрая сортировка (Хоара) (*QuickSort*)

Идея

Вход: массив A , индексы l и r , которые определяют подмассив для сортировки $A[l...r]$.

Выход: отсортированный подмассив $A[l...r]$.

- 1) Разбить массив A на 2 части: $A[l...q-1]$ (где все элементы $\leq A[q]$) и $A[q+1...r]$ (где все элементы $> A[q]$). Элемент $X=A[q]$ - опорный.
- 2) Рекурсивное решение задачи для $A[l...q-1]$ и $A[q+1...r]$.



Алгоритм

QuickSort(A,l,r)

If $l < r$ then

$q = \text{Partition}(A,l,r)$

$\text{QuickSort}(A,l,q-1)$

$\text{QuickSort}(A, q+1,r)$

Partition(A,l,r)

$X = A[r]$

$i = l - 1$

for $j = l$ to $r-1$

 if $A[j] \leq X$ then

$i = i + 1$

$A[i] \leftrightarrow A[j]$

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

return $i+1$

Алгоритм (случайный выбор опорного элемента)

RandomQuickSort(A,l,r)

If $l < r$ then

$q = \text{RandomPartition}(A,l,r)$

$\text{RandomQuickSort}(A,l,q-1)$

$\text{RandomQuickSort}(A, q+1,r)$

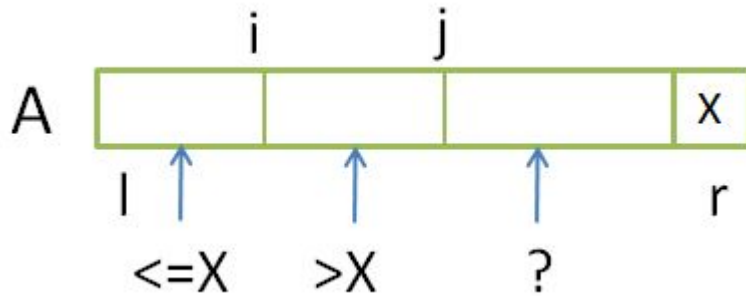
RandomPartition(A,l,r)

$i = \text{random}(l,r)$

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

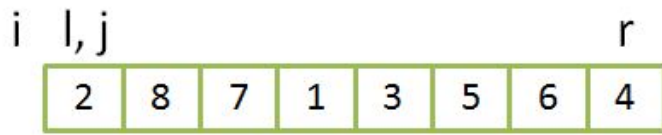
$\text{Partition}(A,l,r)$

Процедура разбиения

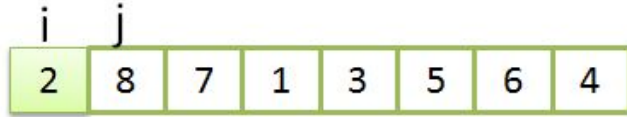


- 1) $X = A[r]$ - опорный элемент (разделитель)
- 2) $A[l \dots i] \leq X$
- 3) $A[i+1 \dots j-1] > X$
- 4) $A[j \dots r-1]$ - элементы, которые еще не рассмотрены

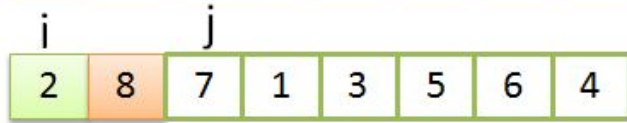
Сложность $T(n) =$
 $O(n)$



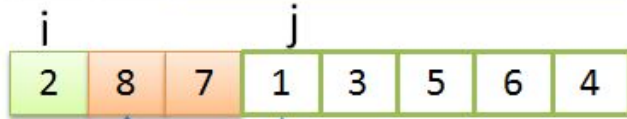
$A[j]=2 < X=4 \Rightarrow i=i+1, j=j+1$



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



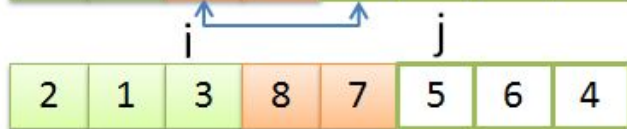
$A[j]=7 > X=4 \Rightarrow j=j+1$



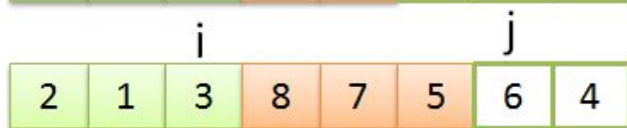
$A[j]=1 < X=4 \Rightarrow i=i+1, j=j+1$



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



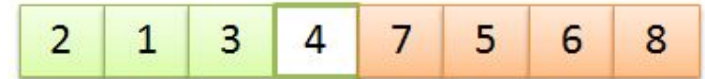
$A[j]=5 > X=4 \Rightarrow j=j+1$



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



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



СЛОЖНОСТЬ

Лучший случай. В наиболее сбалансированном варианте при каждой операции разделения массив делится на две почти одинаковые части. В результате общая сложность алгоритма $O(n \cdot \log_2 n)$.

Средний случай. В среднем глубина рекурсии не превысит $2 \log_{3/4} n$, что равно $O(\log n)$. А поскольку на каждом уровне рекурсии по-прежнему выполняется не более $O(n)$ операций, средняя сложность составит $O(n \cdot \log n)$.

Худший случай. В самом несбалансированном варианте (в качестве опорного выбран максимальный или минимальный элемент) каждое разделение даёт два подмассива размерами 1 и $n-1$. Т.о. сложность $O(n^2)$.

Сортировка слиянием (*merge sort*)

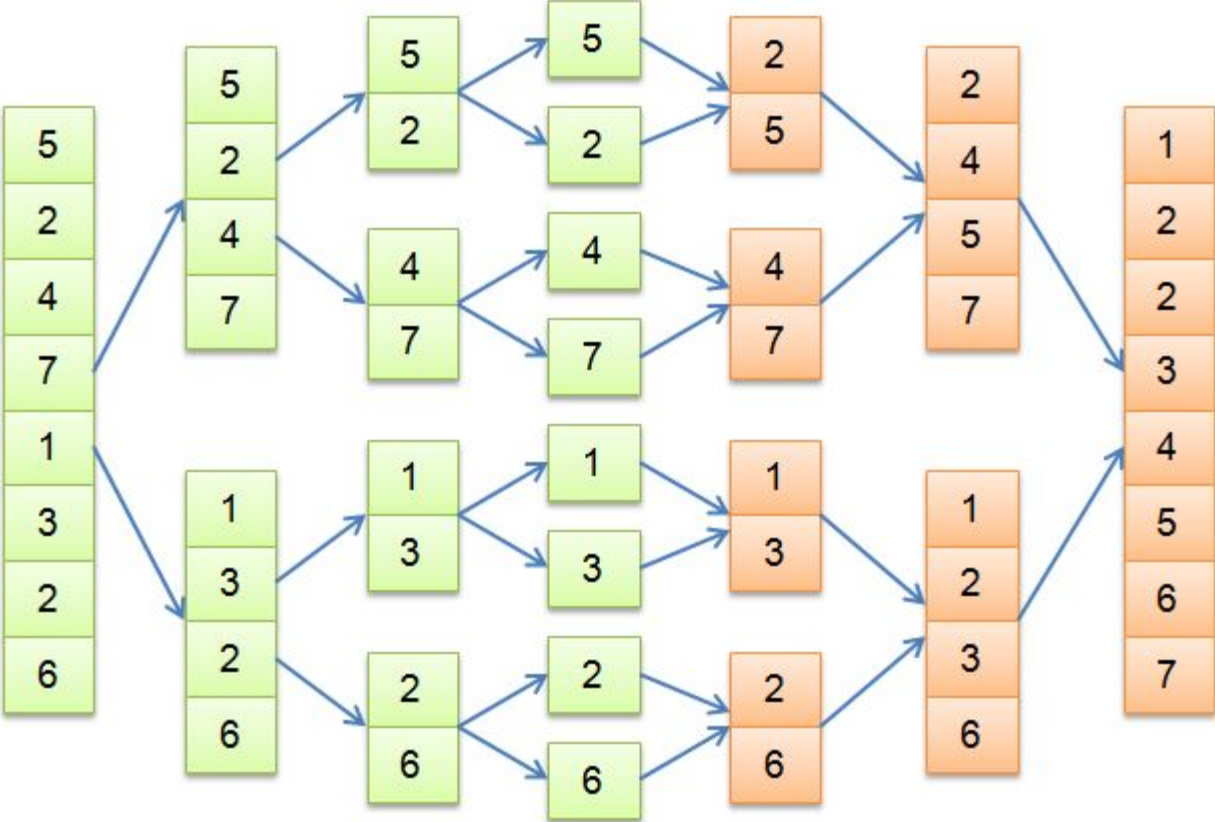
Идея

Вход: массив A , индексы l и r , которые определяют подмассив для сортировки $A[l\dots r]$.

Выход: отсортированный подмассив $A[l\dots r]$.

- 1) Сортируемый массив разбивается на две части примерно одинакового размера
- 2) Каждая из получившихся частей сортируется отдельно, например — тем же самым алгоритмом
- 3) Два упорядочённых массива половинного размера сливаются в один.

Пример



Алгоритм MergeSort

MergeSort(A,l,r)

If $l < r$ then

$q = \lfloor (l + r) / 2 \rfloor$

MergeSort(A,l,q)

MergeSort(A,q+1,r)

Merge(A,l,q,r)

Сложность алгоритма

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

$V(n) = O(n)$

Алгоритм Merge

Merge(A,l,q,r)

n1= q-l+1

n2= r-q

Создать массивы

L[1...n1+1] и R[1...n2+1]

for i = 1 to n1

L[i]=A[l+i-1]

for j = 1 to n2

R[j]=A[q+j]

L[n1+1]= ∞

R[n2+1]= ∞

i= 1

j= 1

for k = l to r

if L[i]<=R[j] then

A[k]=L[i]

i=i+1

else

A[k]=R[j]

j=j+1