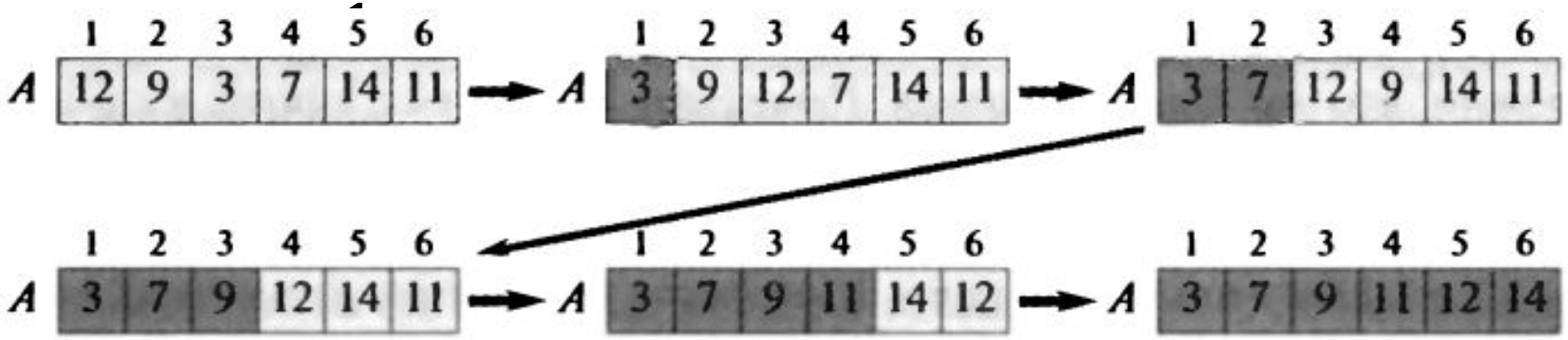
Лабораторная №2.
Алгоритмы сортировки

Сортировка выбором

- Проходим по всему массиву, находим наименьший элемент и меняем этот элемент местами с первым элементом.
- Вновь проходим по массиву, начиная со второго элемента, находим наименьший элемент среди оставшихся и меняем этот элемент со вторым элементом массива.
- То же самое выполняется для третьего элемента и т.д.
- После того, как нужный элемент поставлен в по



Процедура Selection-Sort(A,n).

Вход:

- A – сортируемый массив.
- n – количество сортируемых элементов в массиве A.

Результат: элементы массива A отсортированы в неубывающем порядке.

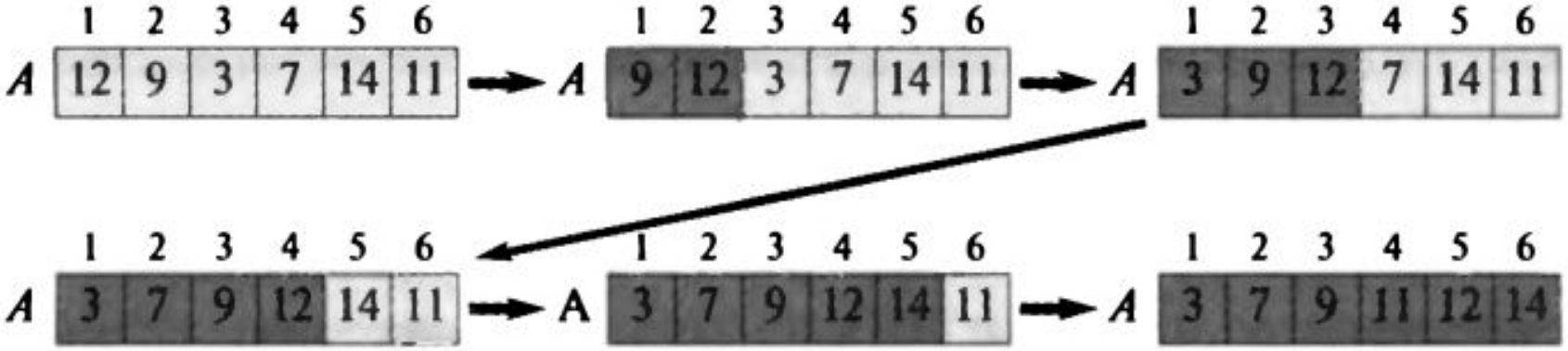
Шаги процедуры:

1. Для i = 1 до n-1:
    A. Установить значение переменной smallest равным i.
    B. Для j = i+1 до n:
        i. Если A[j] < A[smallest] присваиваем переменной smallest значение j.
    C. Обменять A[i] ↔ A[smallest].

Сортировка вставкой

- Сортировка ведется так, что элементы в первых i позициях — это те же элементы, которые были изначально в первых i позициях, но теперь отсортированные в правильном порядке (по возрастанию).
- Чтобы определить, куда надо вставить элемент, первоначально находившийся в A[i], сортировка вставкой проходит по подмассиву A[1..i-1] справа налево, начиная с элемента A[i-1], и переносит каждый элемент, больший, чем A[i] на одну позицию вправо.

- При обнаружении элемента, который не превышает A[i], или перемещения до левого конца массива, элемент, изначально находившийся в A[i], переносится в его новую позицию в массиве.



Алгоритм сортировки вставкой

Процедура Insertion-Sort(A, n).
Вход и результат: те же, что и в Selection-Sort.
Шаги процедуры:
1. Для i = 0 до n-2:
   A. Установить переменную key равной A[i], а переменной j присвоить значение i-1.
   B. Пока j > 0 и A[j] > key, выполнять следующее:
      i. Присвоить A[j+1] значение A[j].
      ii. Уменьшить j на единицу.
   C. Присвоить A[j + 1] значение key.

Сортировка слиянием

Парадигма *«разделяй и властвуй»*

1) *Разделение*. Задача разбивается на несколько подзадач, которые представляют собой меньшие экземпляры той же самой задачи.

2) *Властвование*. Рекурсивно решаются подзадачи. Если они достаточно малы, они решаются как базовый случай.

3) *Объединение*. Решения подзадач объединяются в решение исходной задачи.

Разделяем сортируемый подмассив путем нахождения значения q посредине между p и r:

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

Рекурсивно сортируем элементы в каждой половине подмассива, созданной на шаге разделения (от p до q и от q+1 до r).

Объединение отсортированных элементов в промежутках от p до q и от q+1 до r так, чтобы элементы в промежутке от p-го до r-го были отсортированы.

Процедура Merge-Sort(A,p,r).

Вход: A – массив, p, r – начальный и конечный индексы подмассива A.

Результат: элементы подмассива A[p..r] отсортированы в неубывающем порядке.

Шаги процедуры:

1. Если p≥r, подмассив A[p..r] содержит не более одного элемента, так что он автоматически является отсортированным. Выполняем возврат из процедуры без каких-либо действий.

2. В противном случае выполняем следующие действия: $q = \lfloor (p+r)/2 \rfloor$

   A. Установить

   B. Рекурсивно вызвать Merge-Sort(A,p,q).

   C. Рекурсивно вызвать Merge-Sort(A,q+1,r).

   D. Вызвать Merge(A,p,q,r).

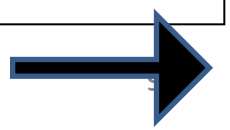
Процедура Merge(A,p,q,r).

Вход: A – массив, p, q, r – индексы в массиве A. Подмассивы A[p..q] и A[q+1..r] считаются уже отсортированными.

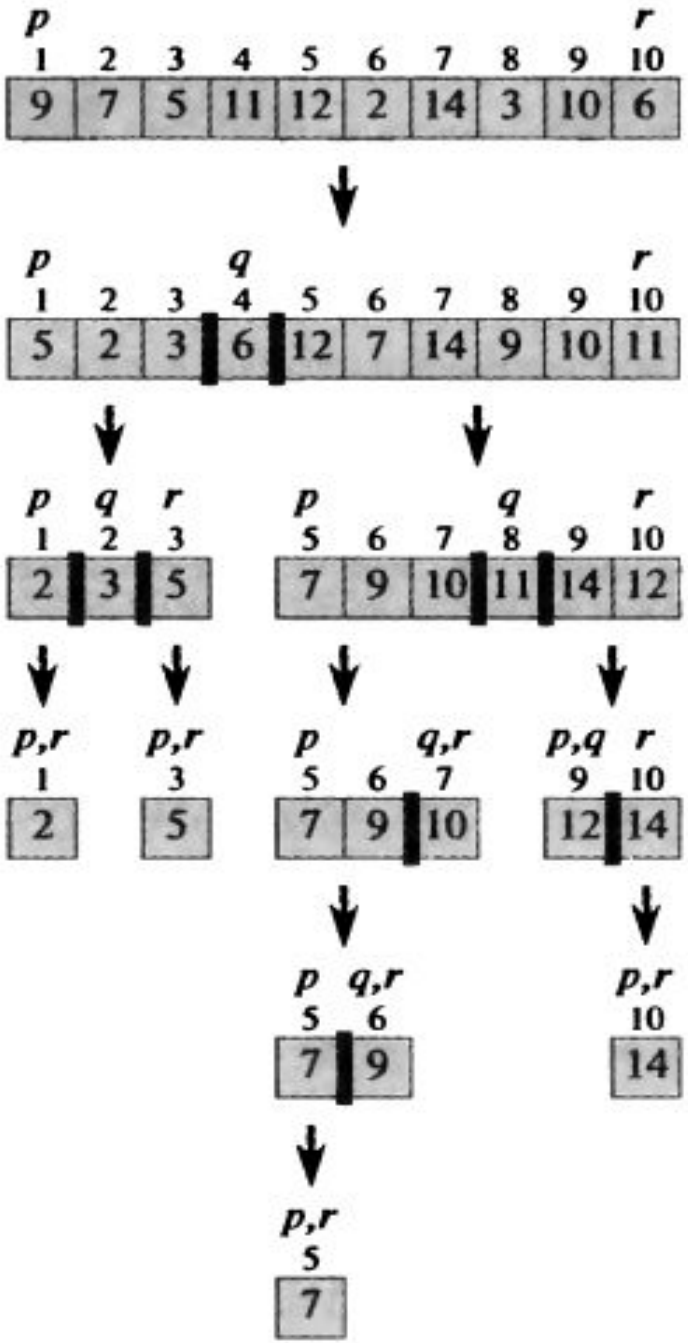
Результат: отсортированный подмассив A[p..r], содержащий все элементы, изначально находившиеся в подмассивах A[p..q] и A[q+1..r].

Шаги процедуры:

1. Установить n_1 равным q-p+1, а n_2 — равным r-q.
2. B[1..n_1 +1] и C[1..n_2+1] представляют собой новые массивы.
3. Скопировать A[p..q] в B[1..n_1], а A[q+1..r] — в C [1..n_2].



4. Установить B[n_1 +1] и C[n_2+1] равными ∞.
5. Установить i и j равными 1.
6. Для k = p до r:
   A. Если B[i] ≤ C[j], установить A[k] равным
      B[i] и увеличить i на 1.
   B. В противном случае (B[i] > C[j])
      установить A[k] равным C[j] и
      увеличить j на 1.



Быстрая сортировка

Выберем один элемент и назовем его опорным. Поместим все элементы, меньшие опорного, слева, а элементы, большие опорного, справа от этого элемента. Если опорный элемент находится в позиции q, то далее рекурсивно сортируются элементы в положениях с p до q-1 и с q+1 до r. Эта рекурсия и дает полностью отсортированный массив.

На примере в качестве опорного элемента используется последний элемент каждого подмассива.

Процедура Quicksort(A,p,r).

Вход и результат: те же, что и у процедуры Merge-Sort.

Шаги процедуры:

1. Если p > r, просто выйти из процедуры, не выполняя никаких действий.
2. В противном случае выполнить следующее:
   A. Вызвать Partition(A,p,r) и установить значение q равным результату вызова.
   B. Рекурсивно вызвать Quicksort(A,p,q-1).
   C. Рекурсивно вызвать Quicksort(A,q+1,r).

Процедура Partition(A,p,r).

Вход: тот же, что и для Merge-Sort.

Результат: перестановка элементов A[p..r], такая, что каждый элемент в A[p..q-1] не превышает A[q], а каждый элемент в A[q+1..r] больше A[q]. Возвращает значение индекса q.

Шаги процедуры:

1. Установить q равным p.
2. Для u = p до r-1:
   A. Если A[u] ≤ A[r], обменять A[q] с A[u], а затем увеличить q на 1.
3. Обменять A[q] и A[r], а затем вернуть q.

Задание

- Сделать сортировку выбором или вставкой
- Сделать одну из рекурсивных сортировок – слиянием или быструю