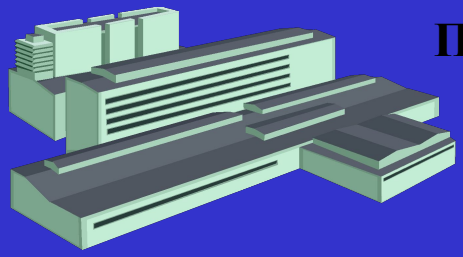
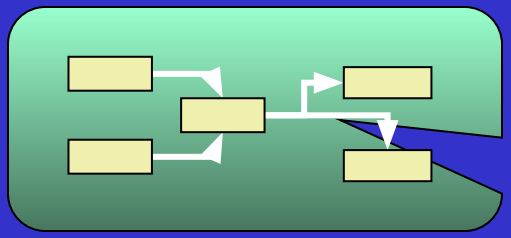
Тема 7. Алгоритмы и структуры данных. Сортировка.

Роль алгоритмов и структур данных

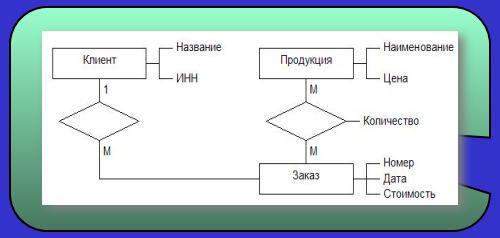


Предметная область

Модель процессов



Модель объектов



Алгоритмы

Структуры данных

Программы

Понятие алгоритма

Алгоритм — это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает пятью важными чертами: конечность, определённость, ввод, вывод, эффективность.

Дональд Эрвин Кнут

Свойства алгоритма

Дискретность — алгоритм должен представлять процесс решения задачи как последовательное выполнение некоторых простых шагов. При этом для выполнения каждого шага алгоритма требуется конечный отрезок времени.

Детерминированность — определённость. В каждый момент времени следующий шаг работы однозначно определяется состоянием системы. Таким образом, алгоритм должен выдавать один и тот же результат для одних и тех же исходных данных.

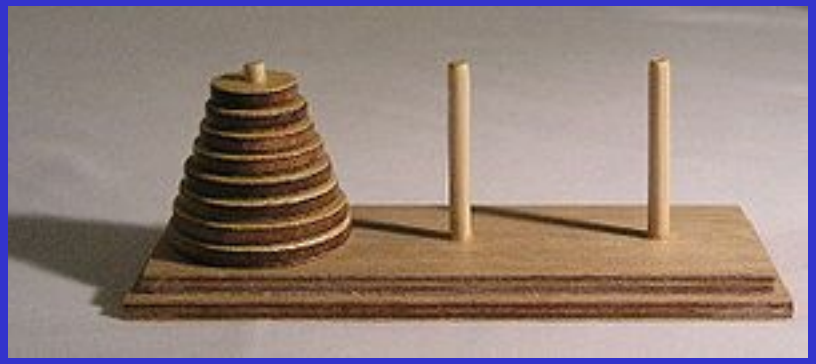
Конечность — при корректно заданных исходных данных алгоритм должен завершать работу и выдавать результат за конечное число шагов.

Масштабируемость — алгоритм должен быть применим к разным наборам исходных данных.

Результативность — завершение алгоритма определенными результатами.

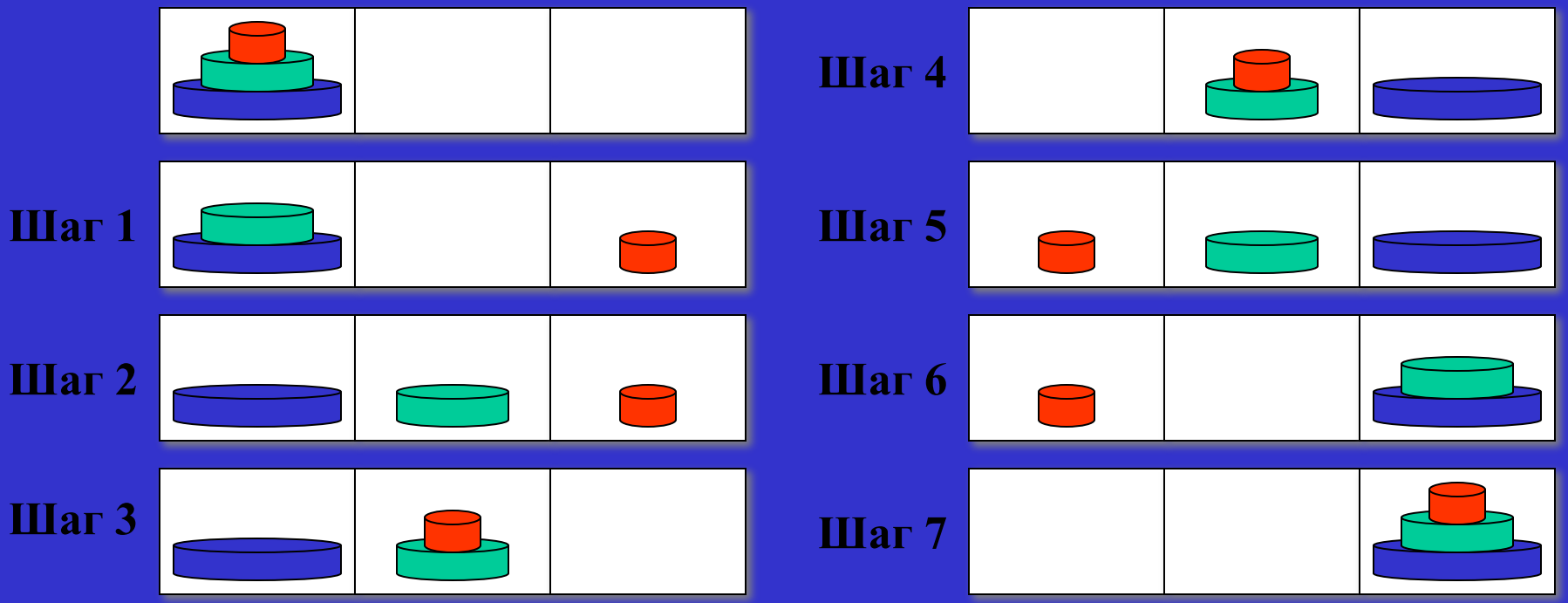
Эффективность — завершение алгоритма определенными результатами за определенное число шагов (время).

Пример алгоритма - задача о Ханойских башнях



Легенда. В одном из буддийских монастырей монахи уже тысячу лет занимаются перекладыванием колец. Они располагают тремя пирамидами, на которых надеты кольца разных размеров. В начальном состоянии 64 кольца были надеты на первую пирамиду и упорядочены по размеру. Монахи должны переложить все кольца с первой пирамиды на вторую, выполняя единственное условие — кольцо нельзя положить на кольцо меньшего размера. При перекладывании можно использовать все три пирамиды. Монахи перекладывают одно кольцо за одну секунду. Как только они закончат свою работу, наступит конец света...

Решение задачи о Ханойских башнях



Число шагов алгоритма вычисляется по формуле $\mathbf{2^N - 1}$, где N – число колец .

Для перекладывания 64-х колец потребуется **18 446 744 073 709 551 615** шагов.

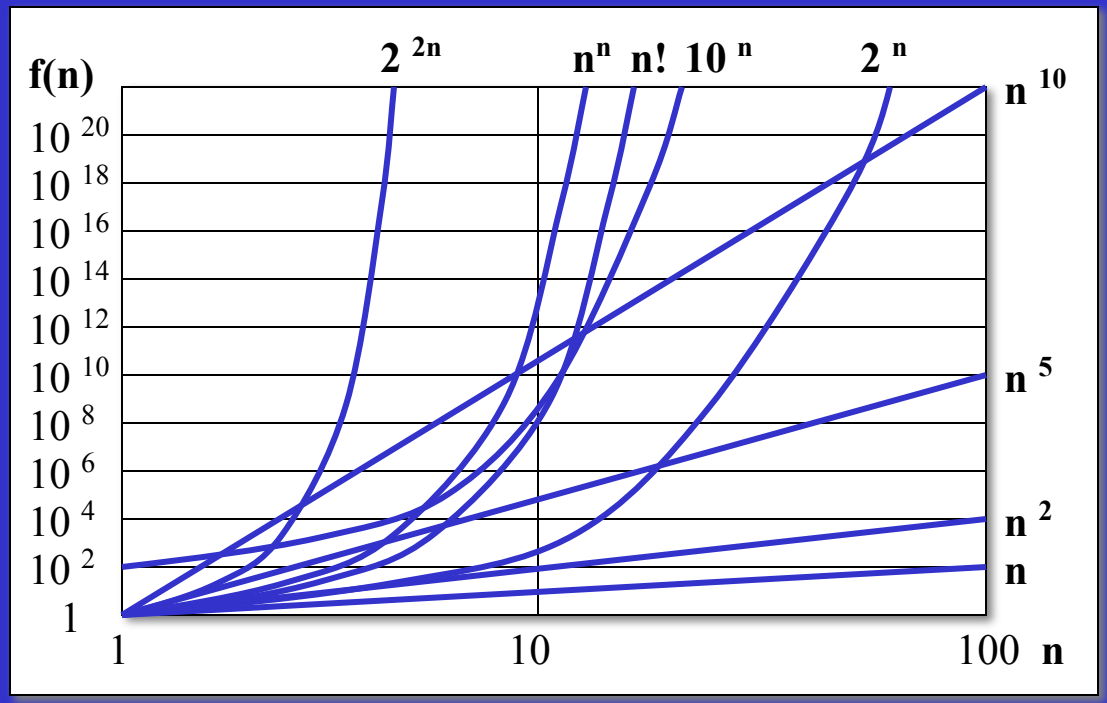
При скорости в одно перекладывание в секунду, потребуется около **584 542 046 091** лет.

Эффективность алгоритмов

Временные фукции сложности

Полиномиальные (P-задачи)

Экспоненциальные (NP-задачи)



Трансвычислительные задачи

Не существует системы обработки данных, искусственной или естественной, которая могла бы обрабатывать более ***2*10^{47}*** *бит в секунду на грамм своей массы.*

Ханс Бреммерман

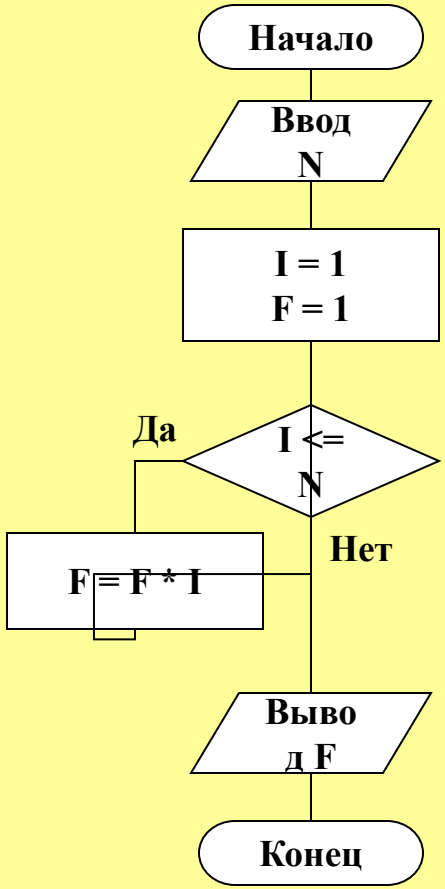
Предел Бреммермана

10^{93} бит

Трансвычислительные задачи

Представления алгоритмов

Блок-схема



Псевдокод

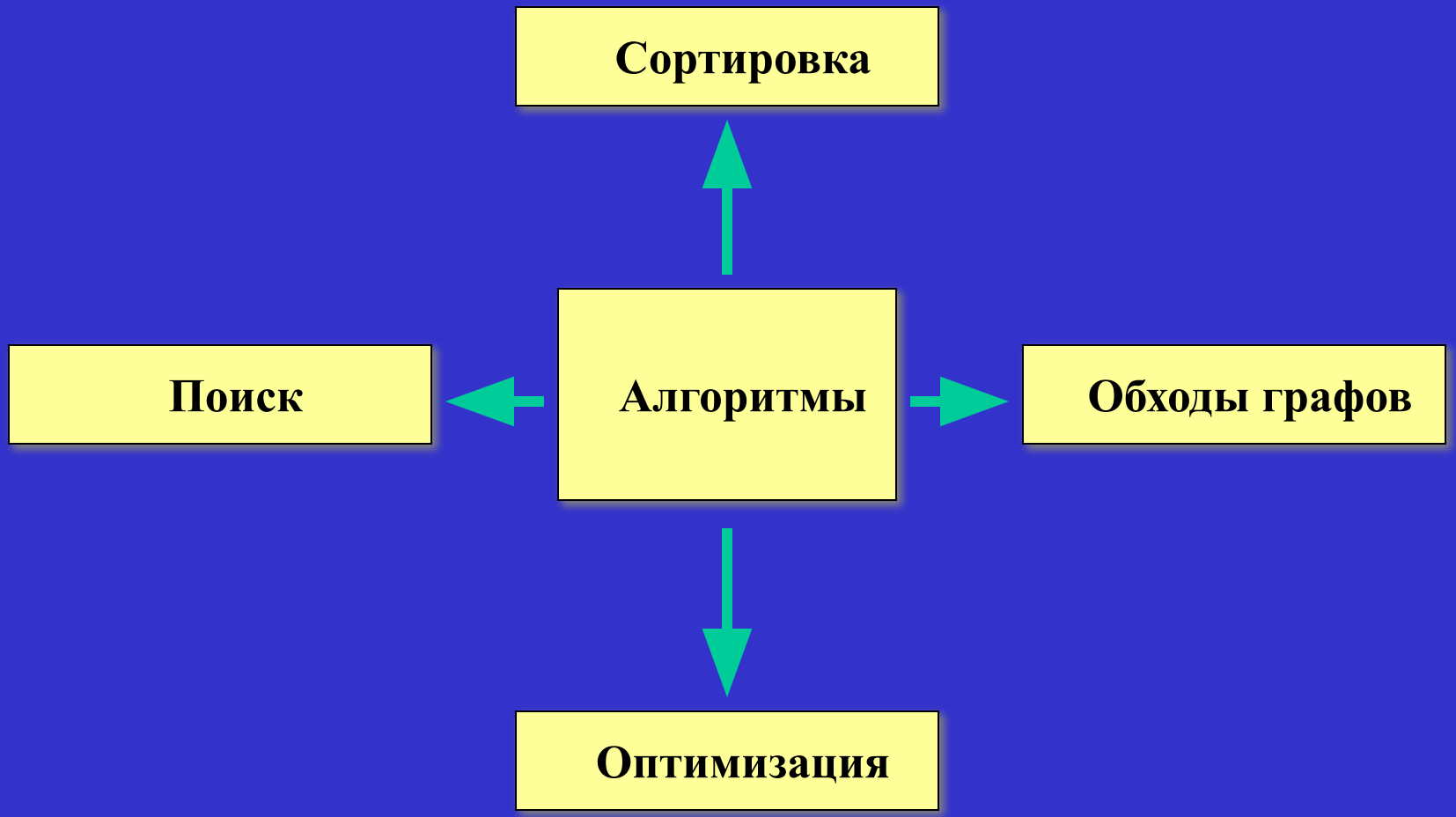
```
Ввод N
I = 1
F = 1
ЦИКЛ ПОКА  I <= N
    F = F * I
ВСЁ-ЦИКЛ
Вывод F
```

Блок-схемы алгоритмов

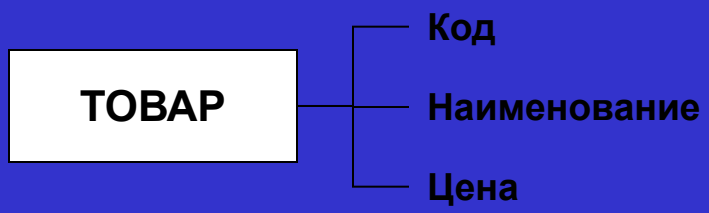
ГОСТ 19.701-90 (ИСО 5807-85). Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.

Наименование	Обозначение	Функция
Терминатор		Элемент отображает вход из внешней среды или выход из нее (наиболее частое применение − начало и конец программы).
Процесс		Выполнение одной или нескольких операций, обработка данных любого вида
Решение		Отображает решение с одним входом и двумя или более альтернативными выходами, из которых только один может быть выбран.
Предопределенный процесс		Символ отображает выполнение процесса, который определен в другом месте программы
Данные		Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод).
Соединитель		Символ отображает выход в часть схемы и вход из другой части этой схемы.
Комментарий		Используется для более подробного описания шага, процесса или группы процессов.

Классы алгоритмов

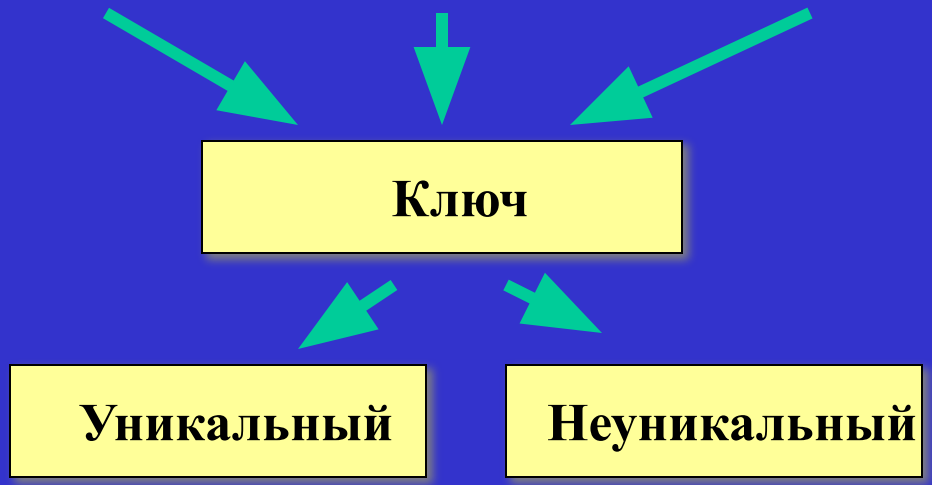


Сортировка массивов



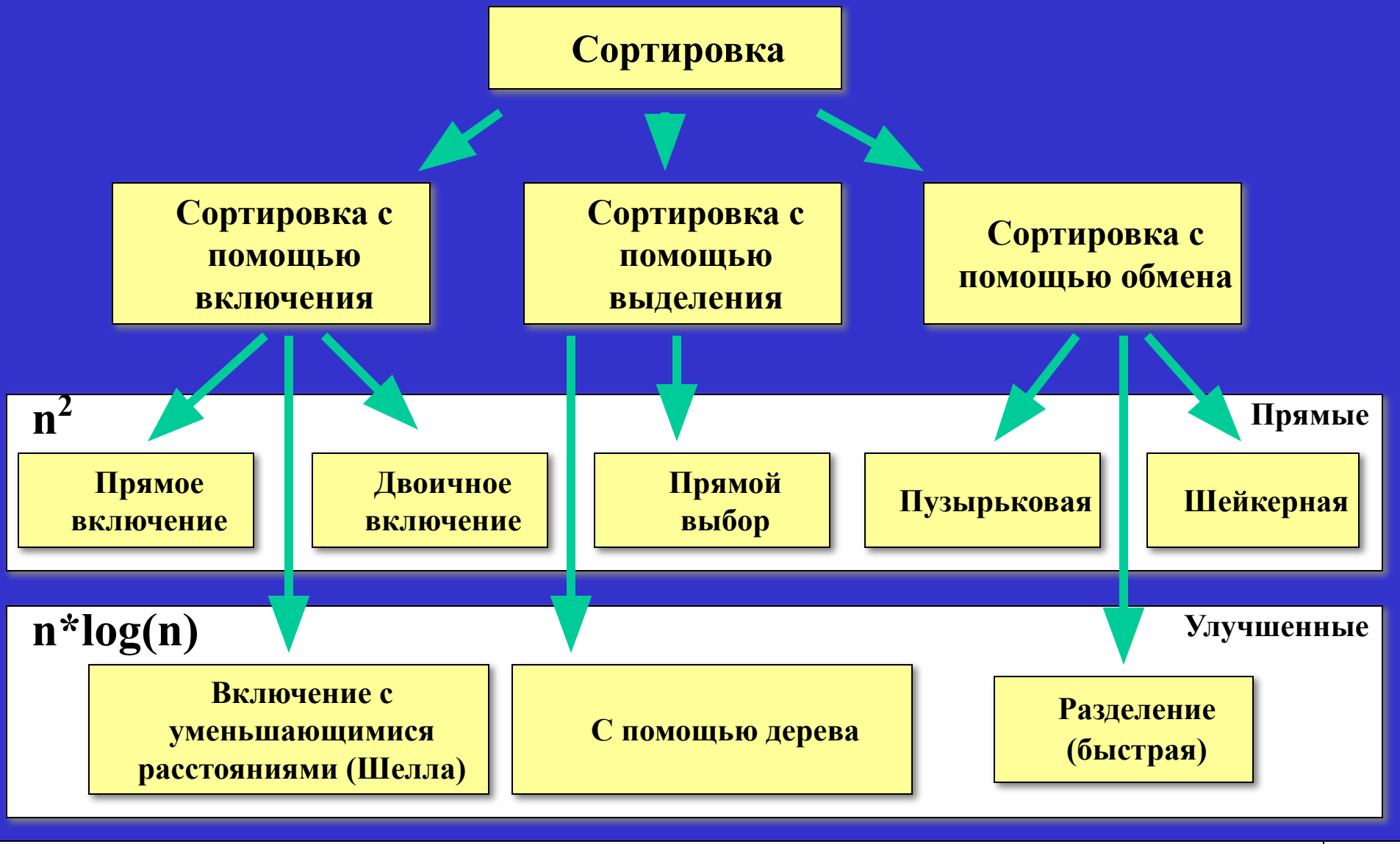
Код	Наименование	Цена
44	Яблоки	35.50
55	Апельсины	29.90
12	Бананы	22.00
...	...	...

```
typedef struct
{
   short code;
   char  name[10];
   float price;
} PROD;

PROD prod[8];
```

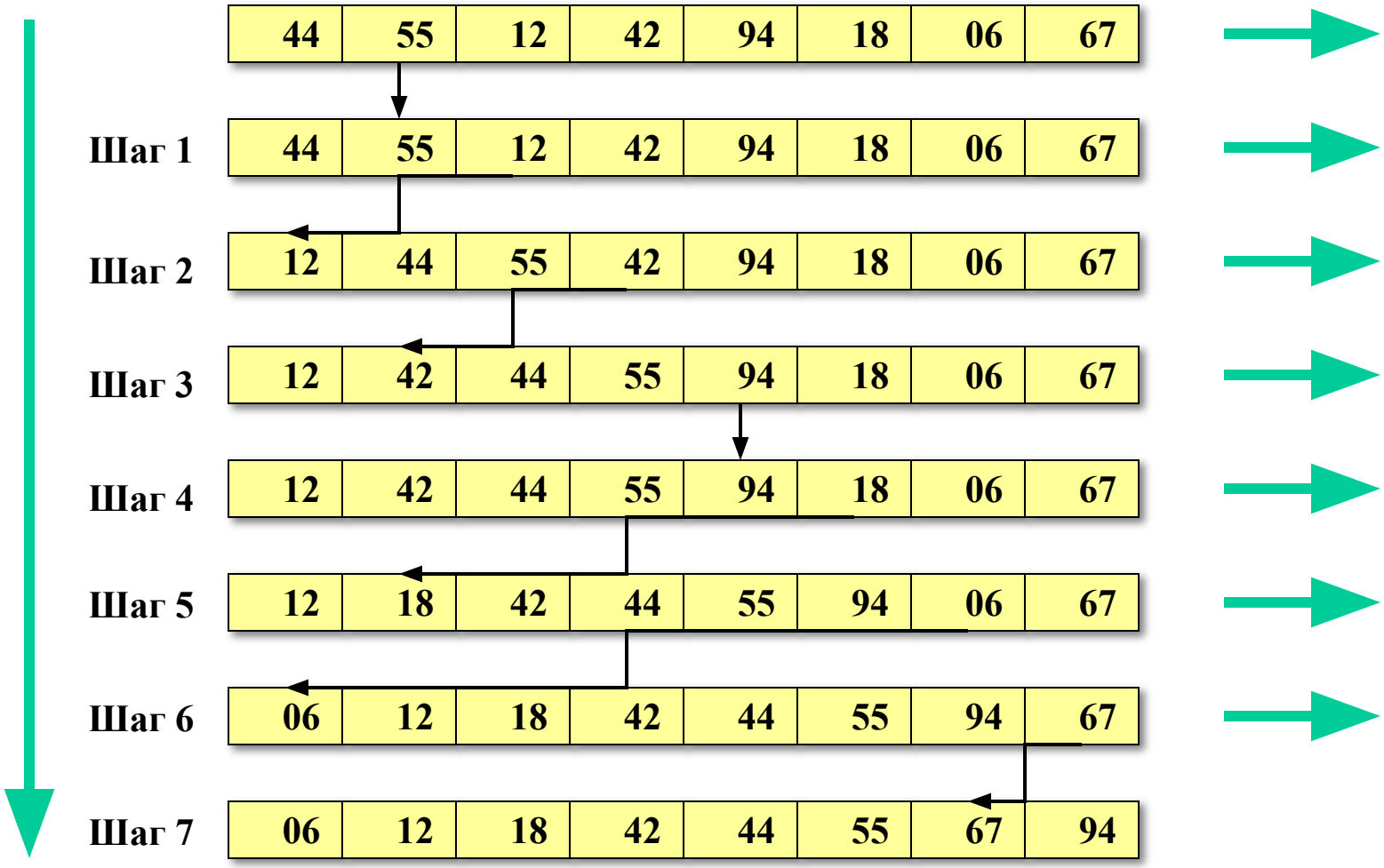


Сортировка *- упорядочение массива в соответствии со значениями ключа*

Алгоритмы сортировки массивов



Сортировка прямым включением (StraightInsertion)

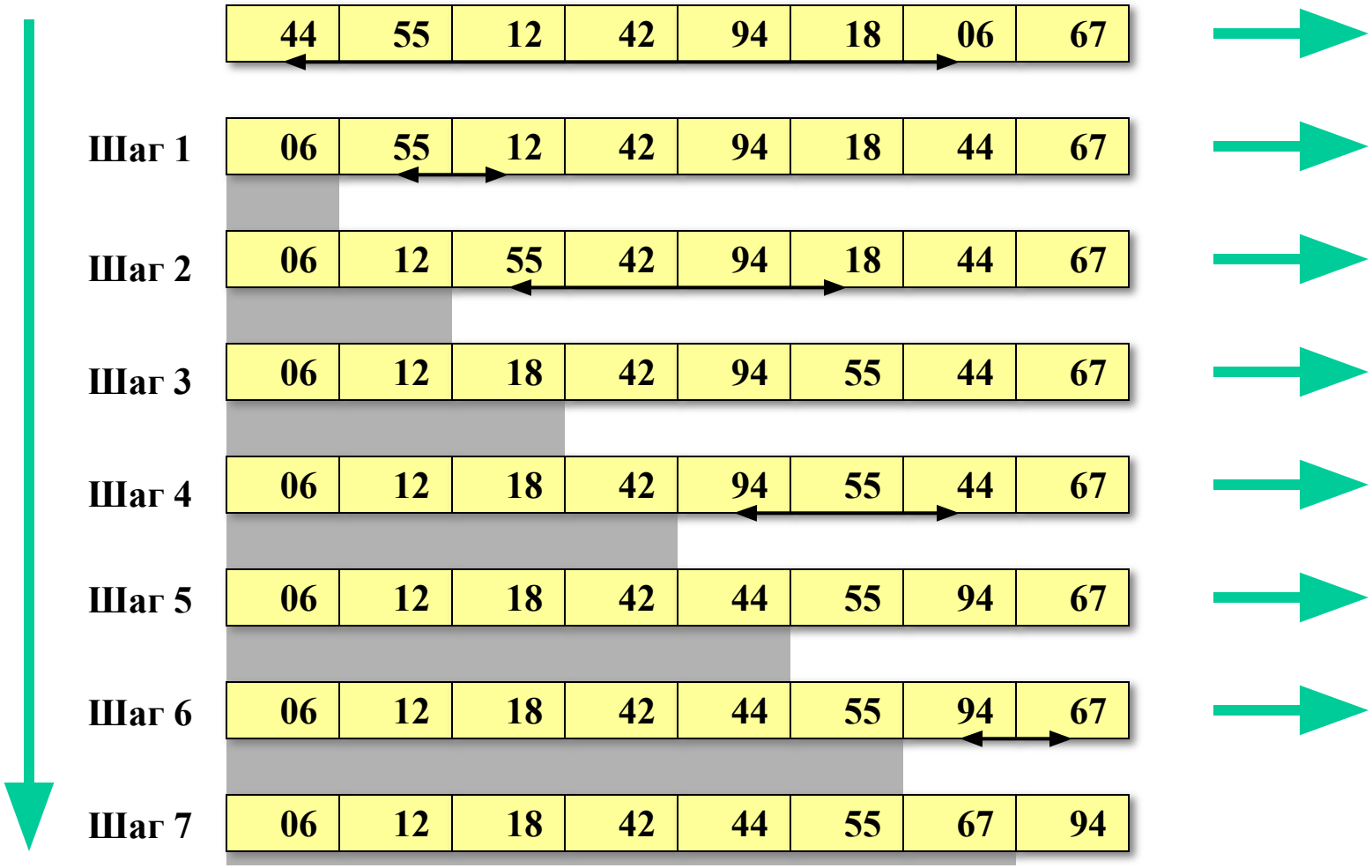


Эффективность $(n^2-n)/2$

Пример сортировки прямым включением

```
void StraightInsertion(PROD* prod, int n)
{
   for(int i = 1; i < n; i++)
   {
      PROD tmp = prod[i];
      int j;

      for(j = i; j > 0 && tmp.code < prod[j-1].code; j--)
         prod[j] = prod[j-1];

      prod[j] = tmp;
   }
}
```

Сортировка прямым выбором (StraightSelection)

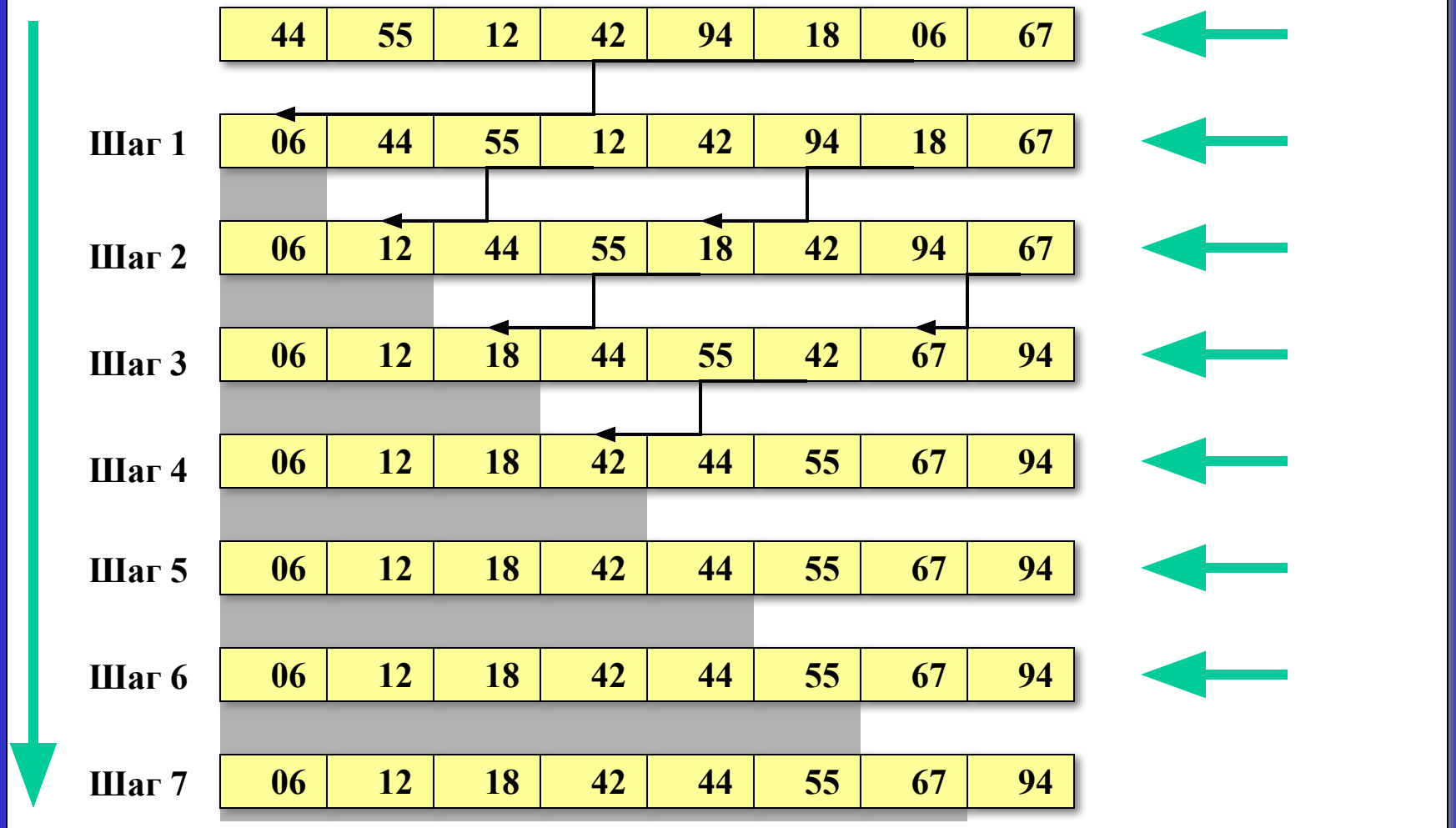


Эффективность $(n^2-n)/2$

Пример сортировки прямым выбором

```
void StraightSelection(PROD* prod, int n)
{
   for(int i = 0; i < n-1; i++)
   {
      PROD tmp = prod[i];
      int k = i;

      for(int j = i+1; j < n; j++)
         if(prod[j].code < tmp.code)
         {
            tmp = prod[j];
            k = j;
         }

      prod[k] = prod[i];
      prod[i] = tmp;
   }
}
```

Сортировка прямым обменом (BubbleSort)

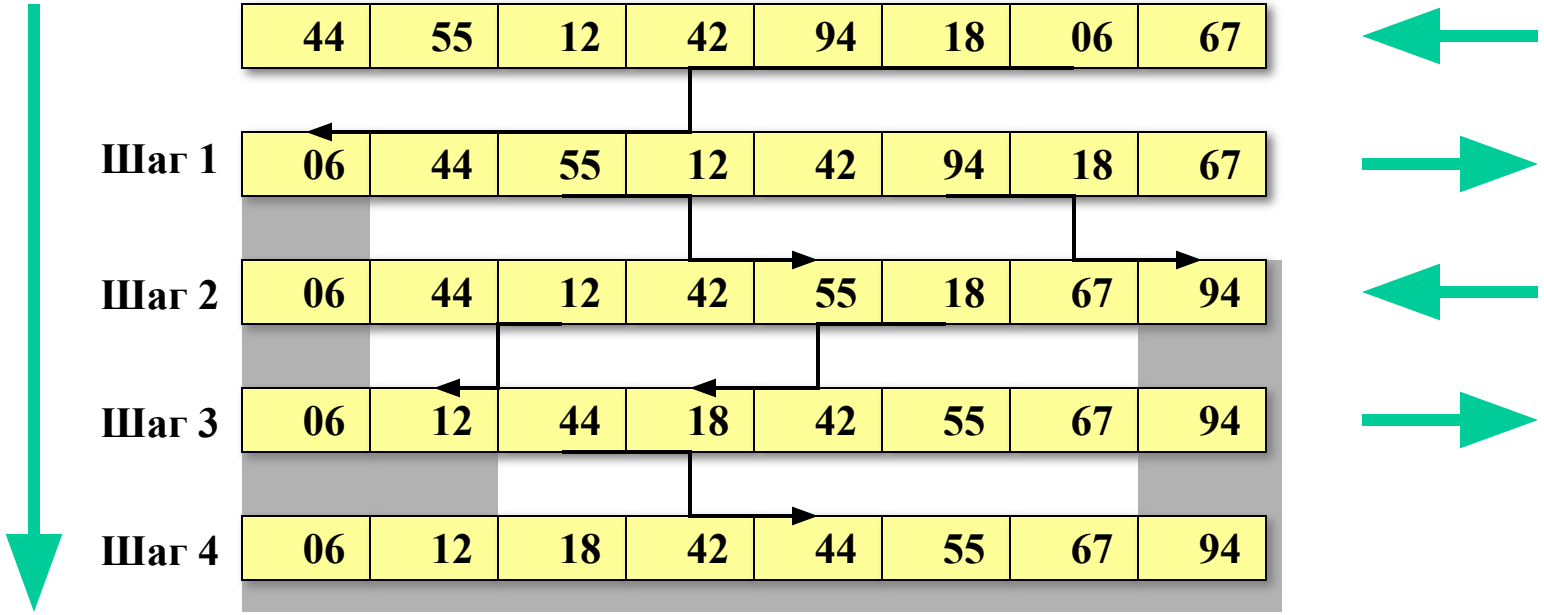


Эффективность $(n^2-n)/2$

Пример сортировки прямым обменом

```
void BubbleSort(PROD* prod, int n)
{
   for(int i = 1; i < n; i++)
      for(int j = n-1; j > i; j--)
         if(prod[j-1].code > prod[j].code)
         {
            PROD tmp = prod[j-1];
            prod[j-1] = prod[j];
            prod[j] = tmp;
         }
}
```

Шейкерная сортировка (ShakerSort)



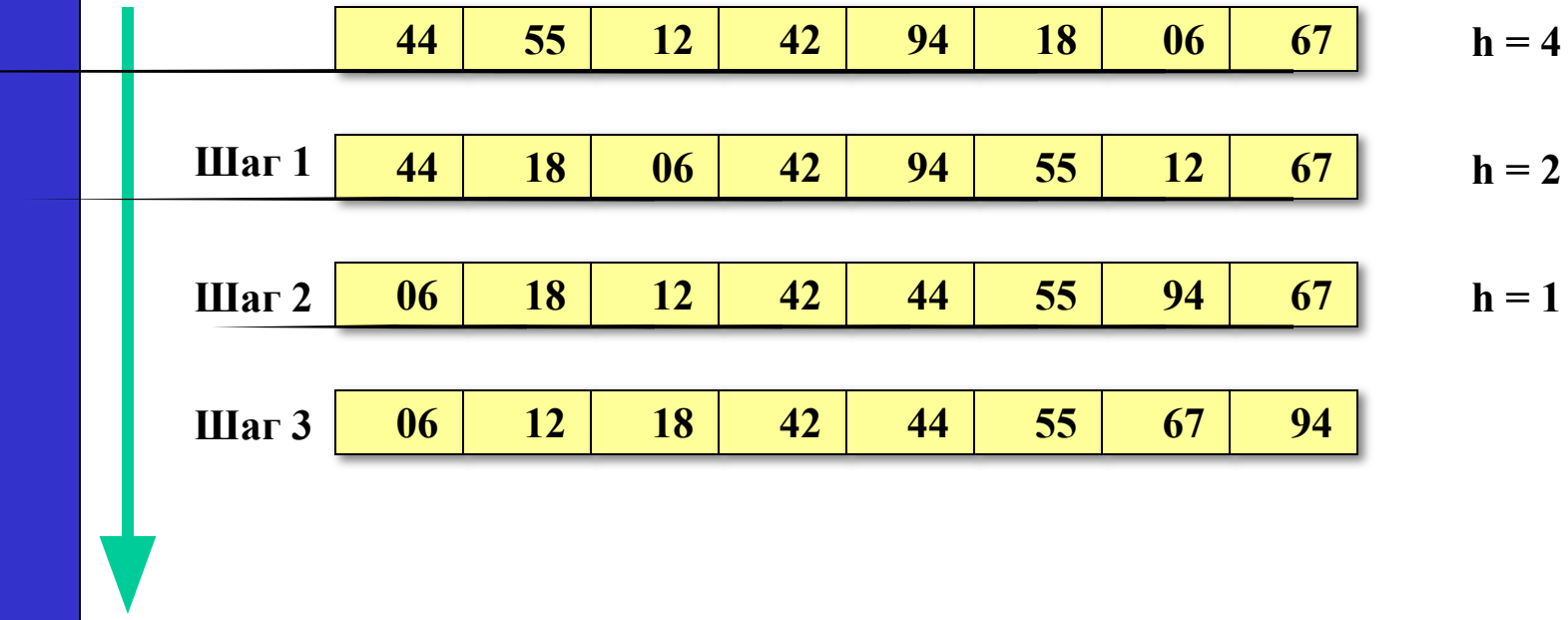
Эффективность $(n^2-n(k+\ln(n)))/2$

Шейкерная сортировка (ShakerSort)

```
void ShakerSort(PROD* prod, int n)
{
   int L = 1, R = n-1, k = n-1;

   do
   {
      for(int j = R; j >= L; j--)
         if(prod[j-1].code > prod[j].code)
         {
            PROD tmp = prod[j-1]; prod[j-1] = prod[j]; prod[j] = tmp;
            k = j;
         }

      L = k+1;

      for(int j = L; j <= R; j++)
         if(prod[j-1].code > prod[j].code)
         {
            PROD tmp = prod[j-1]; prod[j-1] = prod[j]; prod[j] = tmp;
            k = j;
         }

      R = k-1;
   } while(L < R)
}
```

Улучшенный метод сортировки Шелла (ShellSort)

(сортировка включением с уменьшающимися расстояниями)

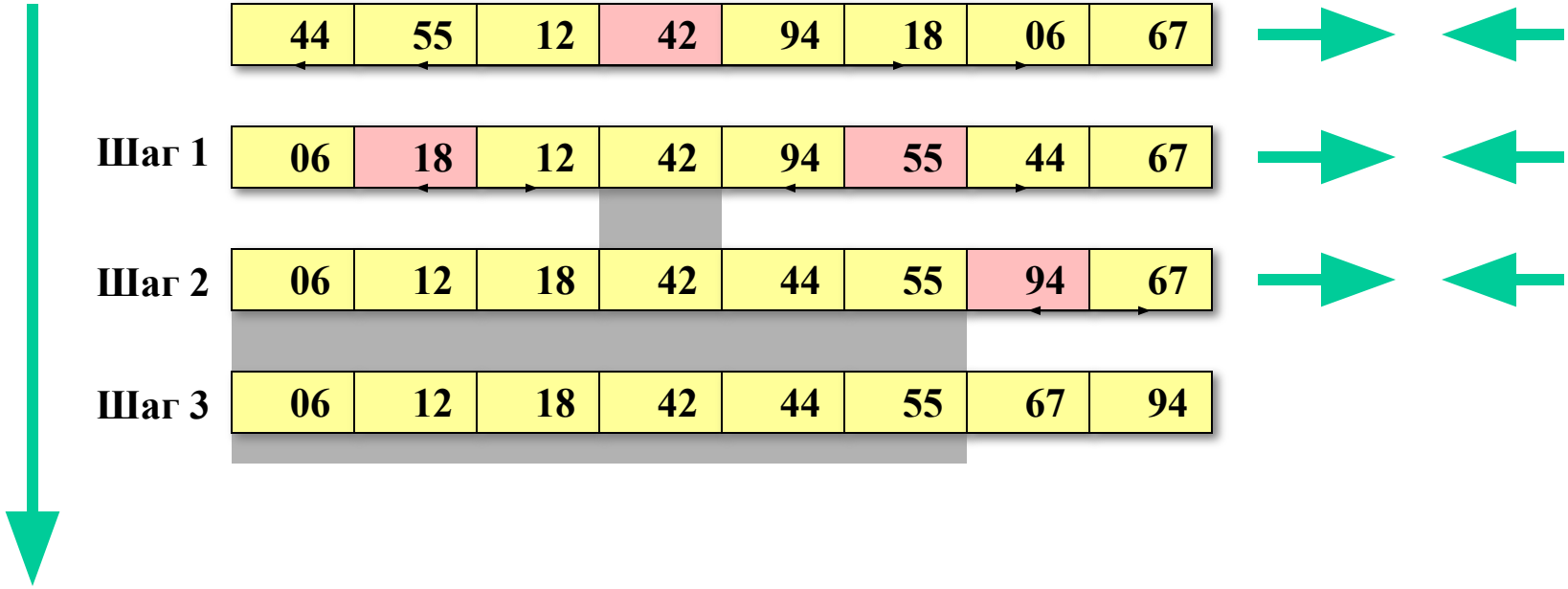


Последовательность расстояний 1, 3, 7, 15, 31... $t = \log_2 n - 1$, $h_t = 1$, $h_{k-1} = 2h_k + 1$

Эффективность $n^{1.2}$

Метод быстрой сортировки Хоара (QuickSort)

(сортировка разделением)



Эффективность n*log(n)

Пример быстрой сортировки

```
void sort(PROD* prod, int L, int R)
{
   int i = L, j = R;
   PROD x = prod[(L+R)/2];
   do
   {
      while(prod[i].code < x.code) i++;
      while(x.code < prod[j].code) j--;
      if(i < j)
      {
         PROD tmp = prod[i]; prod[i] = prod[j]; prod[j] = tmp;
      }
      i++; j--;
   } while(i <= j);
   if(L < j) sort(prod, L, j);
   if(i < R) sort(prod, i, R);
}
```

```
void QuickSort(PROD* prod, int n)
{
   sort(prod, 0, n);
}
```