Тема 8. Алгоритмы и структуры данных. Поиск.

Алгоритмы поиска

Поиск

Поиск в массивах

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

Поиск в строках

А. С. Пушкин

"ЕВГЕНИЙ ОНЕГИН"

ГЛАВА ПЕРВАЯ

«Мой дядя самых честных правил,
Когда не в шутку занемог,
Он уважать себя заставил
И лучше выдумать не мог.
Его пример другим наука;
Но, боже мой, какая скука
С больным сидеть и день и ночь,
Не отходя ни шагу прочь!
Какое низкое коварство
Полуживого забавлять,
Ему подушки поправлять,
Печально подносить лекарство,
Вздыхать и думать про себя:
Когда же черт возьмет тебя!»

Поиск в файлах

Поиск в массивах

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

Ключ

Уникальный

Неуникальный

Поиск в массивах

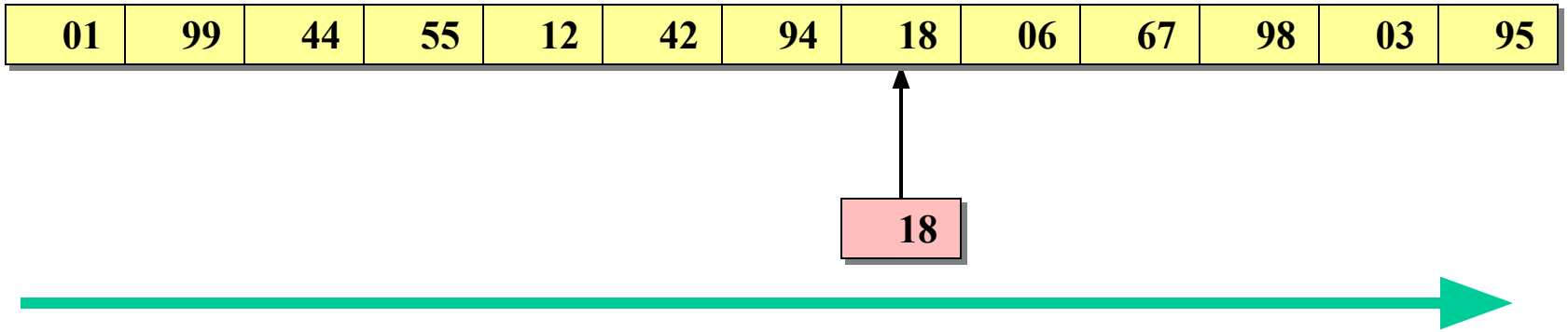
Линейный поиск

Двоичный поиск

Интерполяционный поиск

Поиск *- получение записи по значению ключа*

Линейный поиск в массиве



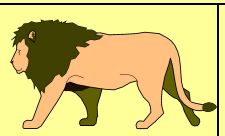
Линейный поиск *- единственно возможный способ поиска в неупорядоченном массиве*

Эффективность n/2

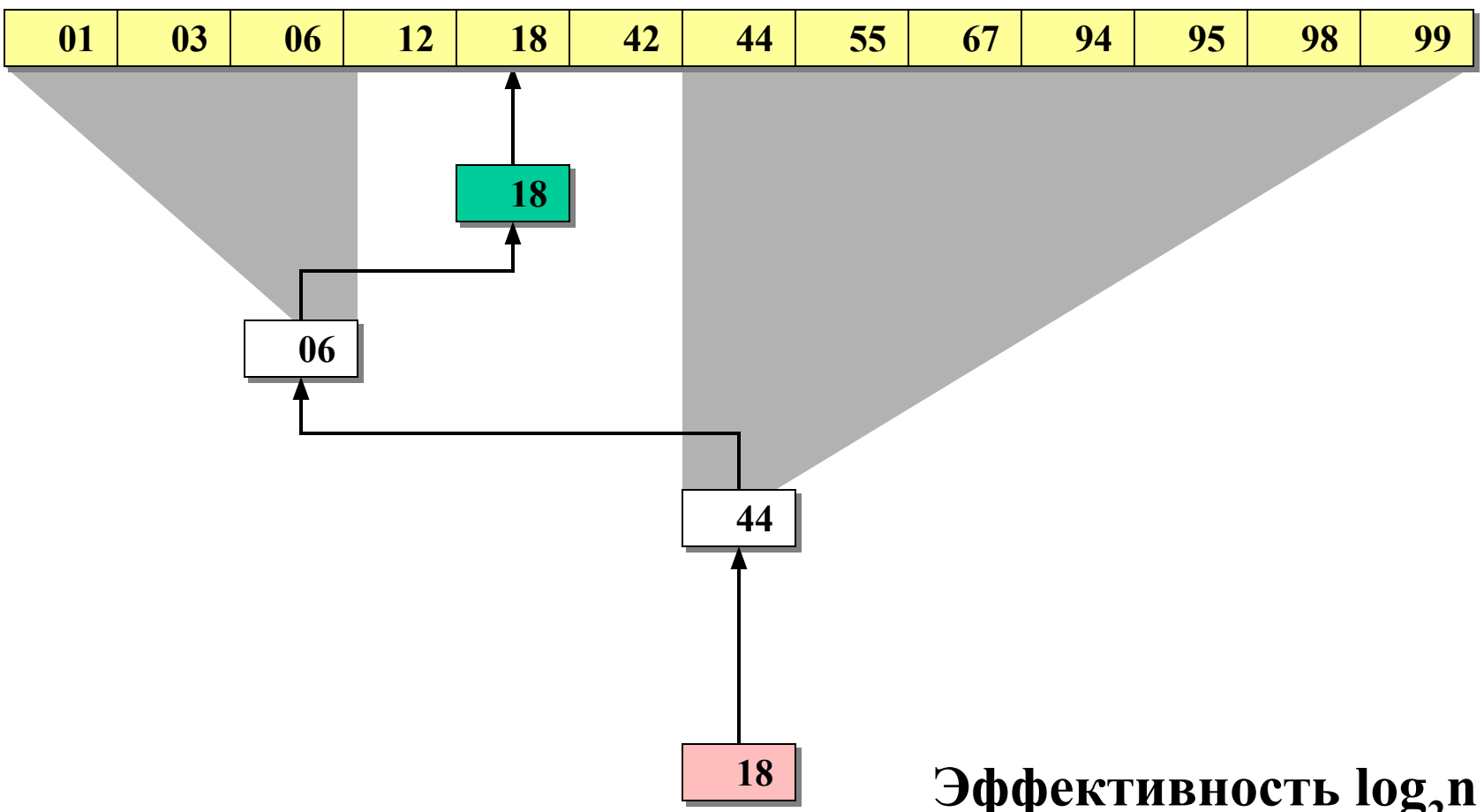
Пример линейного поиска в массиве

```
int LinearSearch(PROD* p, int n, short Val)
{
   for(int i = 0; i < n; i++)
      if(p[i].code == Val)
         return(i);

   return(-1);
}
```

Двоичный поиск в массиве



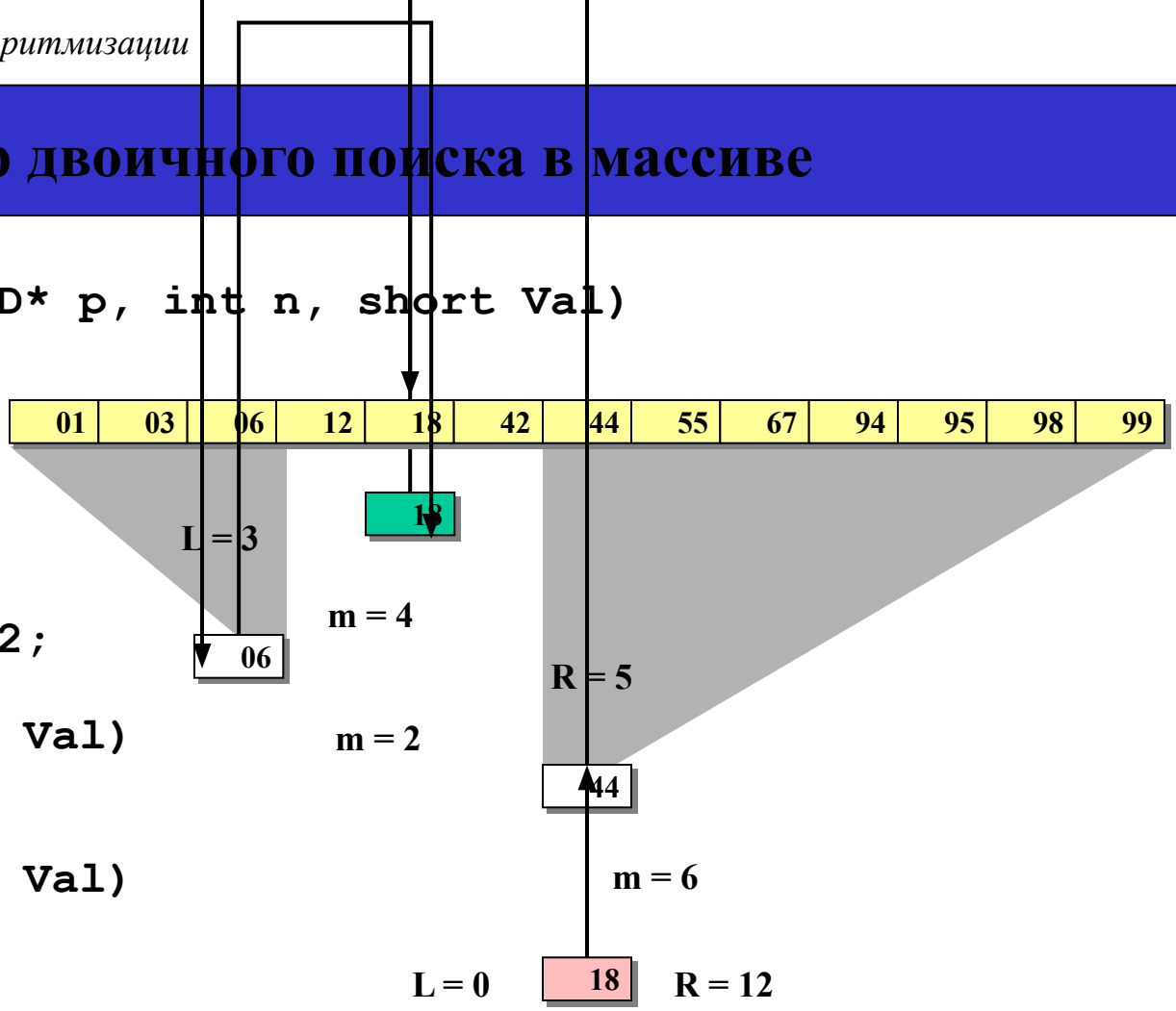
Двоичный поиск в массиве



Эффективность $\log_2 n$

Пример двоичного поиска в массиве

```
int BinarySearch(PROD* p, int n, short Val)
{
   int L = 0;
   int R = n-1;

   while(L <= R)
   {
      int m = (L+R)/2;

      if(p[m].code < Val)
         L = m+1;
      else
      if(p[m].code > Val)
         R = m-1;
      else
         return(m);
   }

   return(-1);
}
```



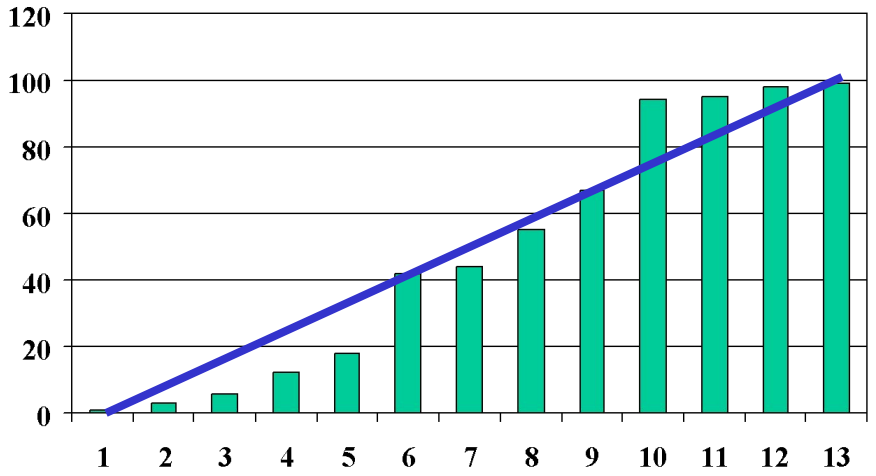
Интерполяционный поиск в массиве

01	03	06	12	18	42	44	55	67	94	95	98	99

12

06

18

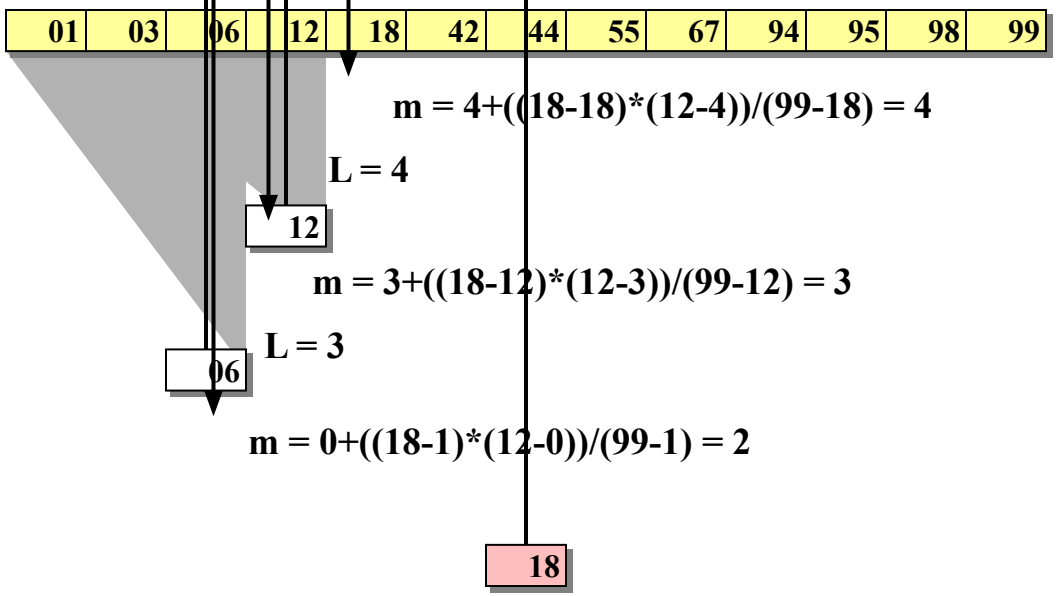



```
int ind = L+((V-a[L])*(R-L))/(a[R]-a[L]);
```

Эффективность $\log_2 n$

Пример интерполяционного поиска в массиве

```
int InterpolationSearch(PROD* p, int n, short Val)
{
   int L = 0;
   int R = n-1;

   while(L <= R)
   {
      int m = L+((Val-p[L].code)*(R-L))/(p[R].code-p[L].code);

      if(p[m].code < Val)
         L = m+1;
      else
      if(p[m].code > Val)
         R = m-1;
      else
         return(m);
   }

   return(-1);
}
```



Поиск в строках

наука

Поиск в строках

- **Прямой поиск**
- **Алгоритм Кнута, Морриса и Пратта**
- **Алгоритм Боуера и Мура**

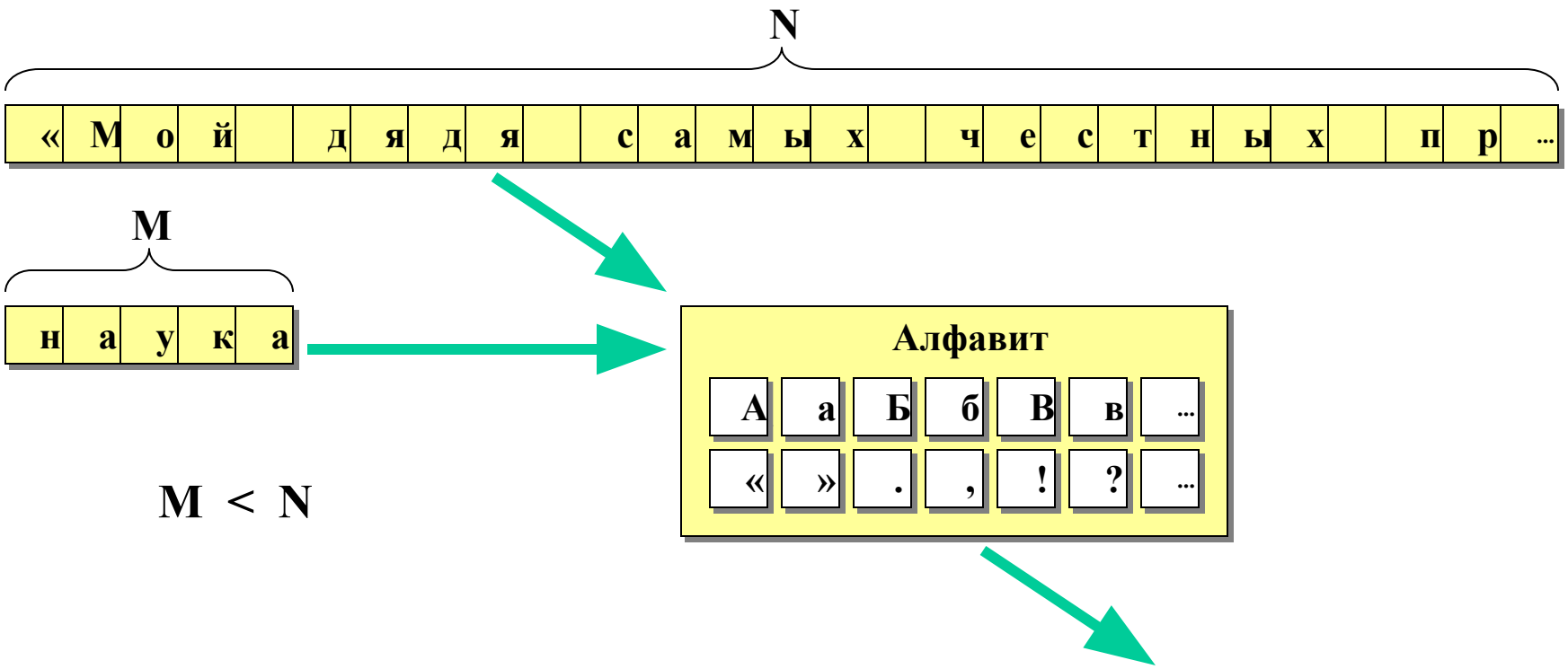
А. С. Пушкин

"ЕВГЕНИЙ ОНЕГИН"

ГЛАВА ПЕРВАЯ

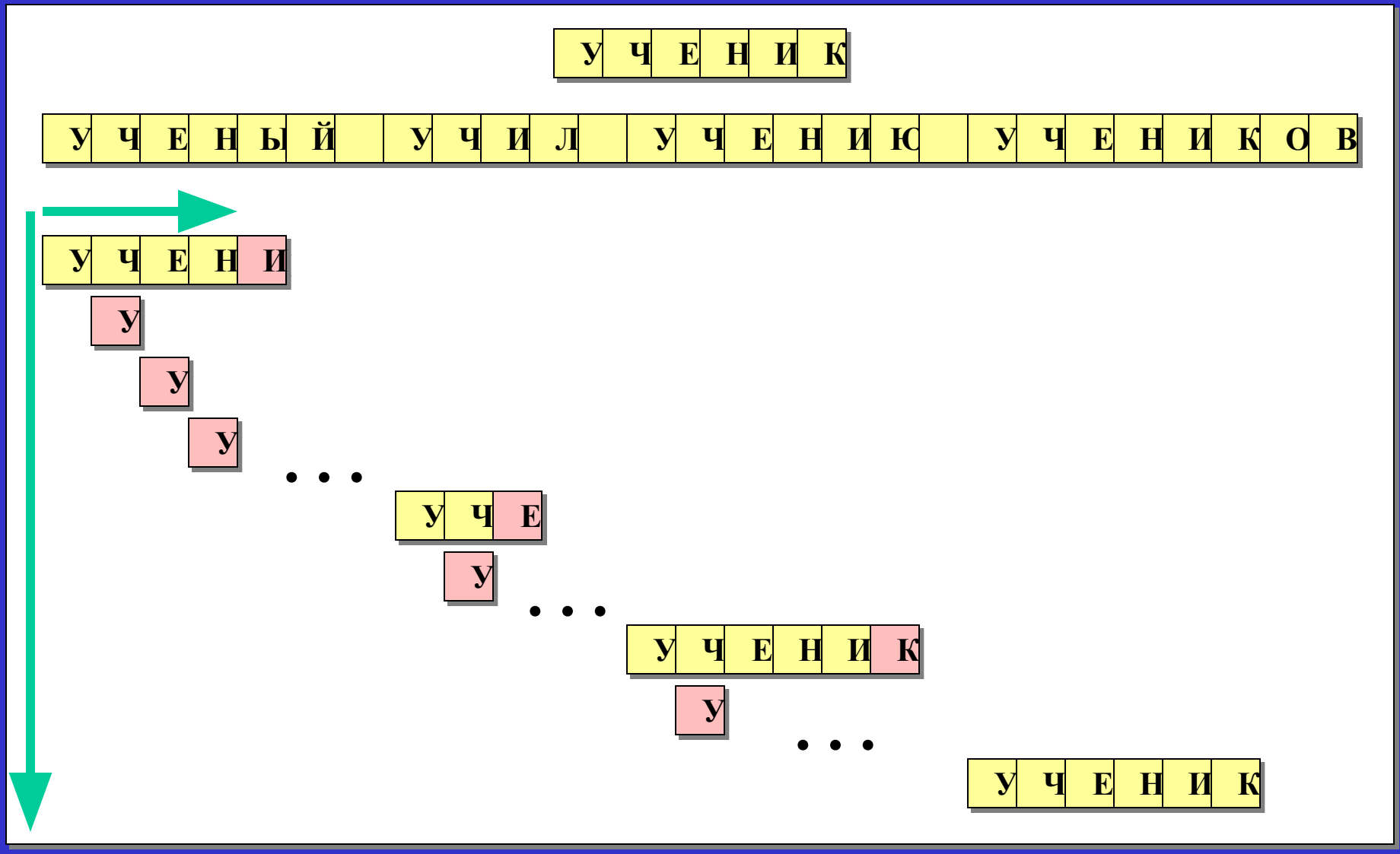
«Мой дядя самых честных правил,
Когда не в шутку занемог,
Он уважать себя заставил
И лучше выдумать не мог.
Его пример другим **наука**;
Но, боже мой, какая скука
С больным сидеть и день и ночь,
Не отходя ни шагу прочь!
Какое низкое коварство
Полуживого забавлять,
Ему подушки поправлять,
Печально подносить лекарство,
Вздыхать и думать про себя:
Когда же черт возьмет тебя!»

Задача поиска в строках



Строка (текст) *- неупорядоченный массив символов, принадлежащих к некоторому алфавиту*

Прямой поиск в строке



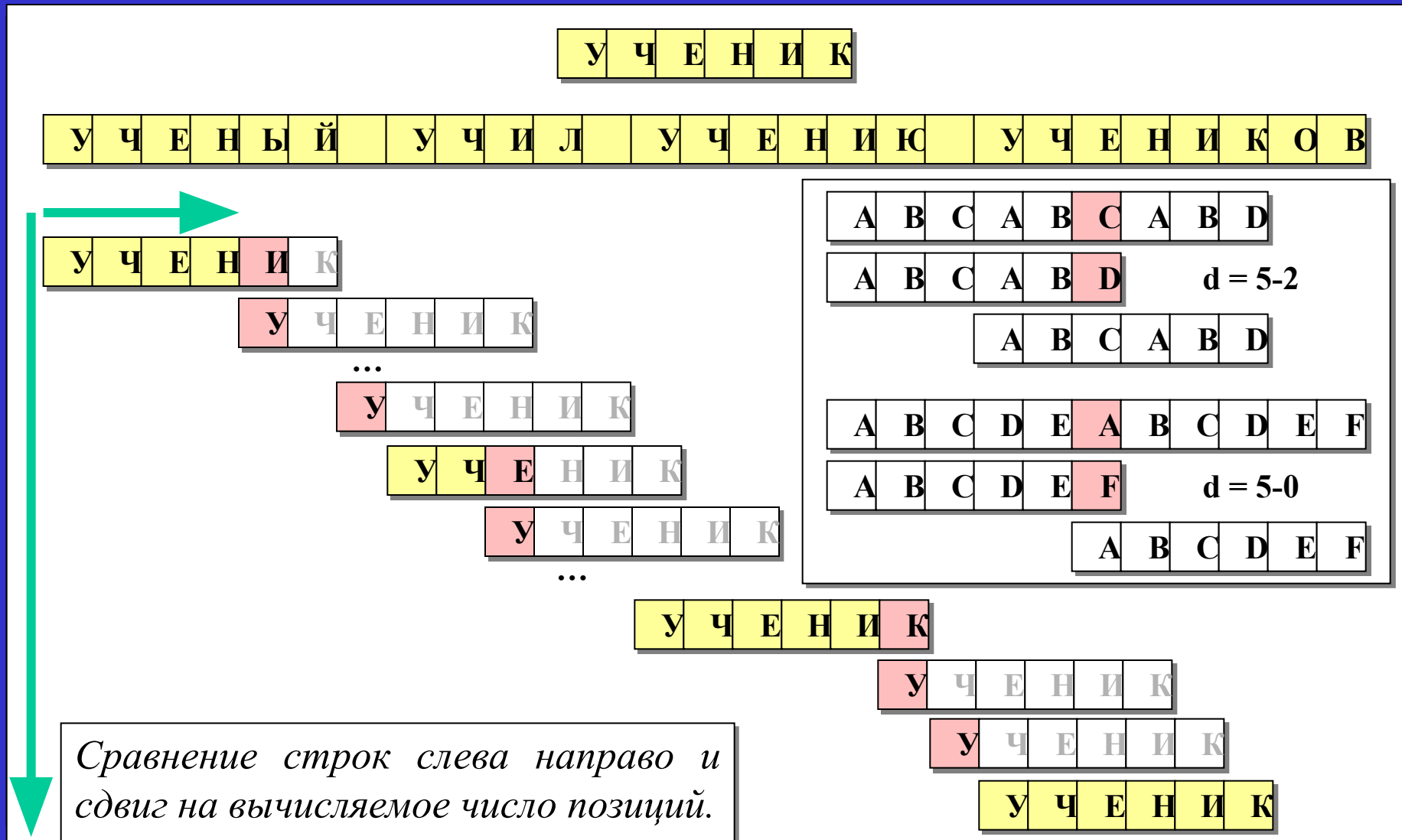
Прямой поиск в строке

```
int StringSearch(char text[], char val[])
{
   int n = strlen(text);   // Длина строки
   int m = strlen(val);// Длина образца

   for(int i = 0; i <= n-m; i++)
   {
      bool found = true;

      for(int j = 0; j < m; j++)
         if(text[i+j] != val[j])
         {
            found = false; break;
         }

      if(found) return(i);
   }

   return(-1);
}
```

Прямой поиск в строке (вариант с указателями)

```
char* StringSearch(char* text, char* val)
{
   for(char* p = text; *p; p++)
   {
      bool found = true;

      for(char *v = val, *s = p; *v; v++, s++)
         if(*v != *s || !*s)
         {
            found = false;
            break;
         }

      if(found) return(p);
   }

   return(NULL);
}
```

Алгоритм Кнута, Морриса и Пратта



Алгоритм Кнута, Морриса и Пратта

```
char* KMP(char* text, char* val)
{
   char* sea = NULL;
   int n = strlen(text);   // Длина строки
   int m = strlen(val);// Длина образца
   int* tab = new int[m];  // Массив префиксов

   Вычисление префиксной функции

   Поиск

   delete [] tab;
   return(sea);
}
```

Алгоритм Кнута, Морриса и Пратта. Префиксная функция

```
// Вычисление префиксной функции

tab[0] = 0;

for(int i = 1, j = 0; i < m; i++)
{
while(j > 0 && val[j] != val[i])
    j = tab[j-1];

if(val[j] == val[i])
    j++;

tab[i] = j;
}
```

Примеры

A	B	C	A	B	C
0	0	0	1	2	3

A	A	C	A	A	C
0	1	0	1	2	3

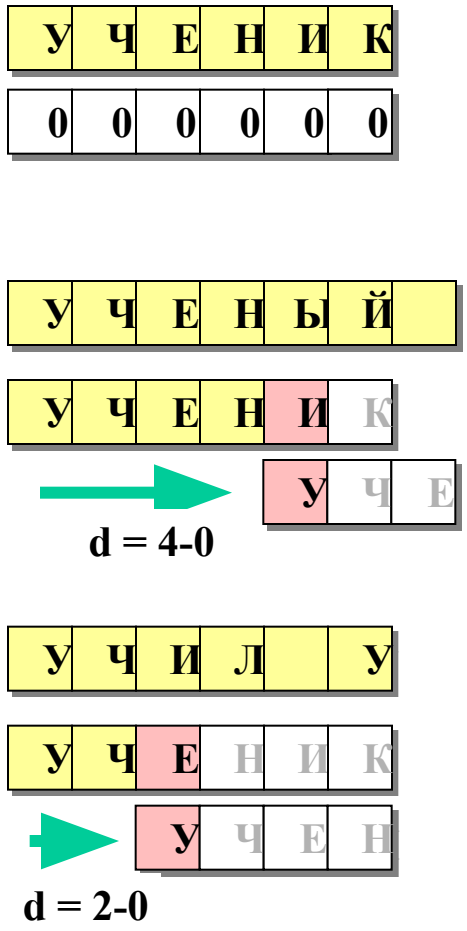
A	A	A	A	A	A
0	1	2	3	4	5

У	Ч	Е	Н	И	К
0	0	0	0	0	0

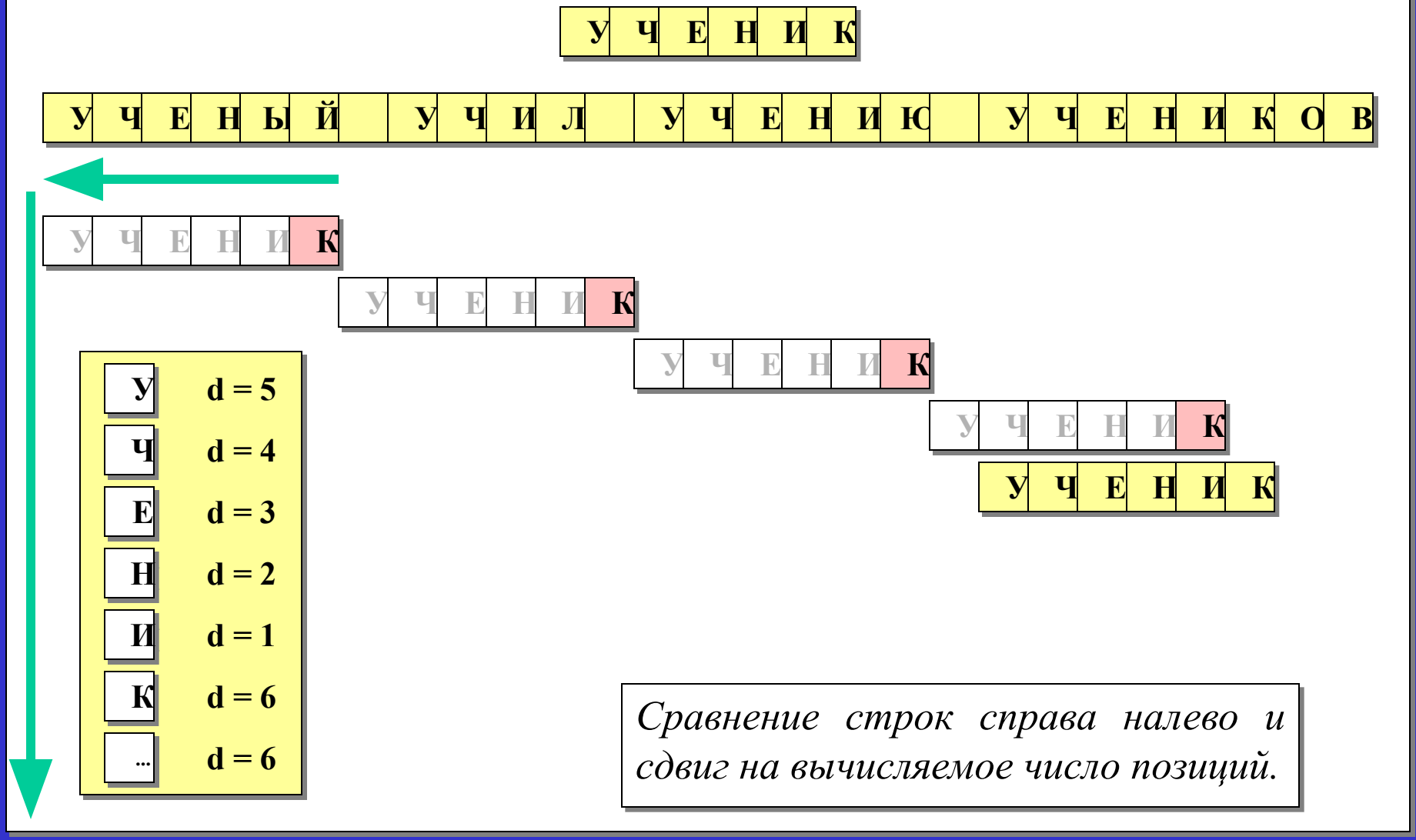
Алгоритм Кнута, Морриса и Пратта. Поиск

```
// Поиск

for(int i = 0, j = 0; i < n; i++)
{
while(j > 0 && val[j] != text[i])
    j = tab[j-1];

if(val[j] == text[i])
    j++;

if(j == m)
{
    sea = text+i-j+1;

    break;
}
}
```

Примеры



Алгоритм Боуера – Мура - Хорспула



Алгоритм Боуера – Мура - Хорспула

```
char* BMH(char* text, char* val)
{
    int n = strlen(text);     // Длина строки
    int m = strlen(val);  // Длина образца
    int tab[256];              // Массив сдвигов

    Заполнение массива сдвигов

    Поиск

    return(NULL);
}
```

Алгоритм Боуера – Мура – Хорспула. Заполнение массива

```
// Заполнение массива сдвигов

for(int i = 1; i < 256; i++)
    tab[i] = m;

for(int j = 0; j < m-1; j++)
    tab[val[j]] = m-j-1;
```

Примеры

A	B	C	A	B	C

...
A = 2
B = 1
C = 3
...

У	Ч	Е	Н	И	К
5	4	3	2	1	6

Алгоритм Боуера – Мура – Хорспула. Поиск

```
for(int i = 0; i <= n-m;)
{
   bool found = true;

   for(int j = m-1; j >= 0; j--)
      if(text[i+j] != val[j])
      {
         i += tab[text[i+m-1]];
      found = false;
      break;
       }

   if(found) return(text+i);
}
```

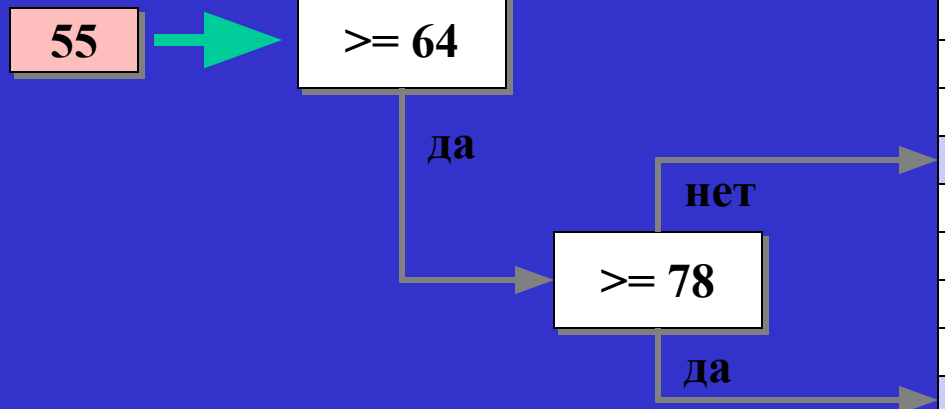
Поиск в файлах базы данных

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

Блок 1	Блок 2	Блок 3	...

Файлы *базы данных размещаются на блочных устройствах (чтение и запись производится блоками фиксированной длины). Чтение блока - медленная операция.*

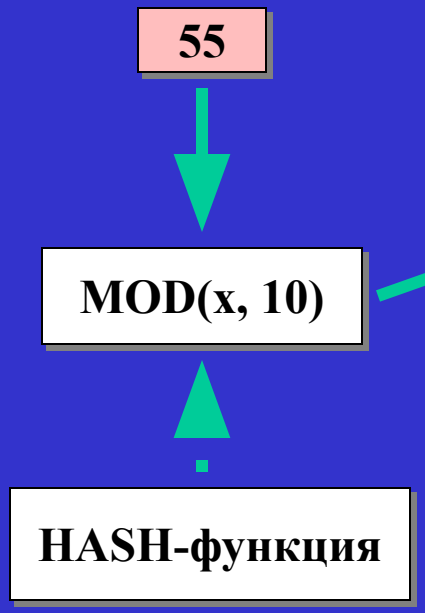
Индексно-последовательный метод доступа (ISAM)



Блок 1		
01	Ананасы	89.00
03	Абрикосы	50.00
12	Бананы	22.00
18	Сливы	34.50
Блок 2		
44	Яблоки	35.50
55	Апельсины	29.90
56	Груши	65.50
62	Вишня	99.90
Блок 3		
64	Черешня	95.00
67	Персики	50.00
68	Манго	67.00
75	Мандарины	39.50
Блок 4		
78	Малина	75.00
83	Нектарины	42.00
98	Гранаты	44.00
99	Айва	38.50

Число ступеней индекса = $\log_2 n$, где n - число блоков

Доступ по вычисляемому ключу (HASH)



Блок 0	
Блок 1	
01	Ананасы
Блок 2	
12	Бананы
62	Вишня
Блок 3	
03	Абрикосы
83	Нектарины

Блок 4	
44	Яблоки
64	Черешня
Блок 5	
55	Апельсины
75	Мандарины
Блок 6	
56	Груши
Блок 7	
67	Персики

Блок 8	
18	Сливы
68	Манго
78	Малина
98	Гранаты
Блок 9	
99	Айва

Хэш-функция *непосредственно преобразует значение ключа поиска в номер блока.*