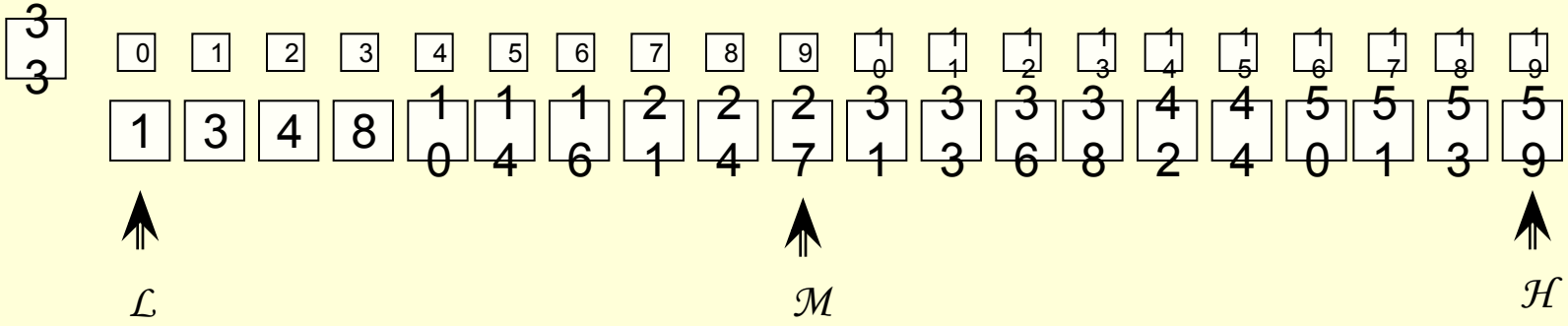
Методы сортировки и поиска

Краткий обзор

Двоичный (бинарный) поиск в упорядоченном массиве



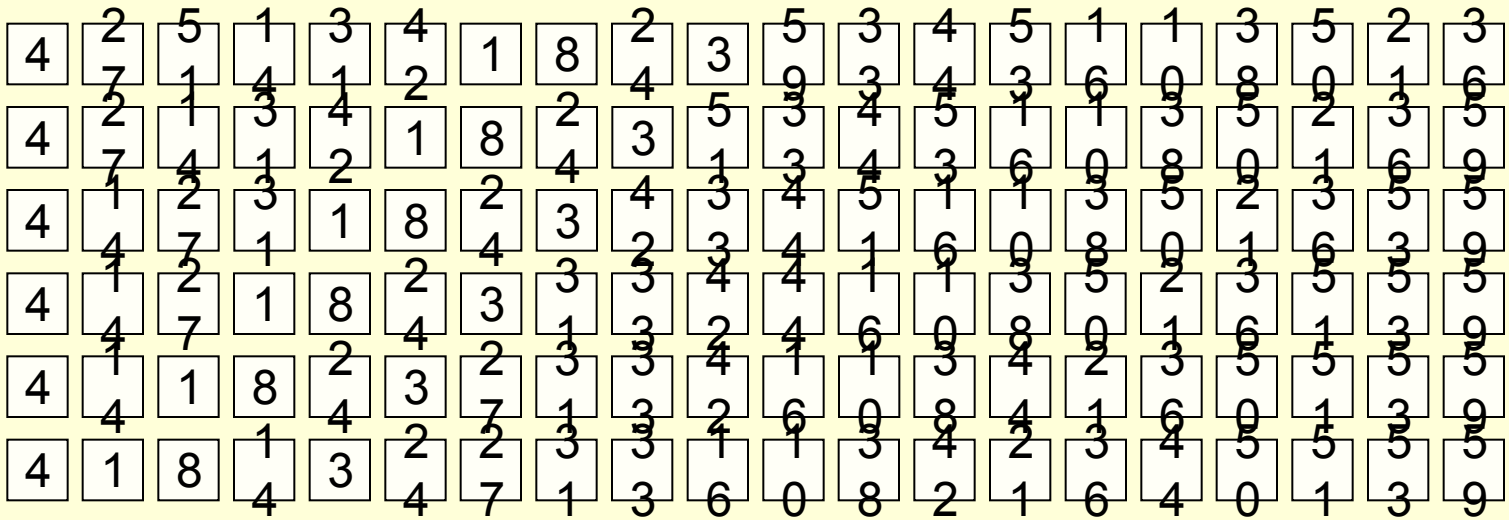

```
const  int   NMAX = 100;
int  a[NMAX];

int binSearch(int data[], int rzm_data, int search_key)
{
   int L,H,M;
   L=0;
   H=rzm_data-1;
   while (L < H)
   {          M = (L + H) / 2;
               if (data[M] < search_key)  L = M + 1;  else H = M;
    }
    if (data[L] == search_key)   return  L; else return  -1;
}
```

Сортировки массива

Сортировка простыми обменами (метод «пузырька»)

```
void  bubbleSort(int  data[], int  rzm_data)
{  int   i, j, tmp;
    for (i=1; i<rzm_data; i++)
        for (j=0; j<rzm_data-i; j++)
            if (data[j] > data [j+1])
            {   tmp = data[j];
                data[j] = data[j+1];
                data[j+1] = tmp;
            }
}
```



Шейкер-сортировка

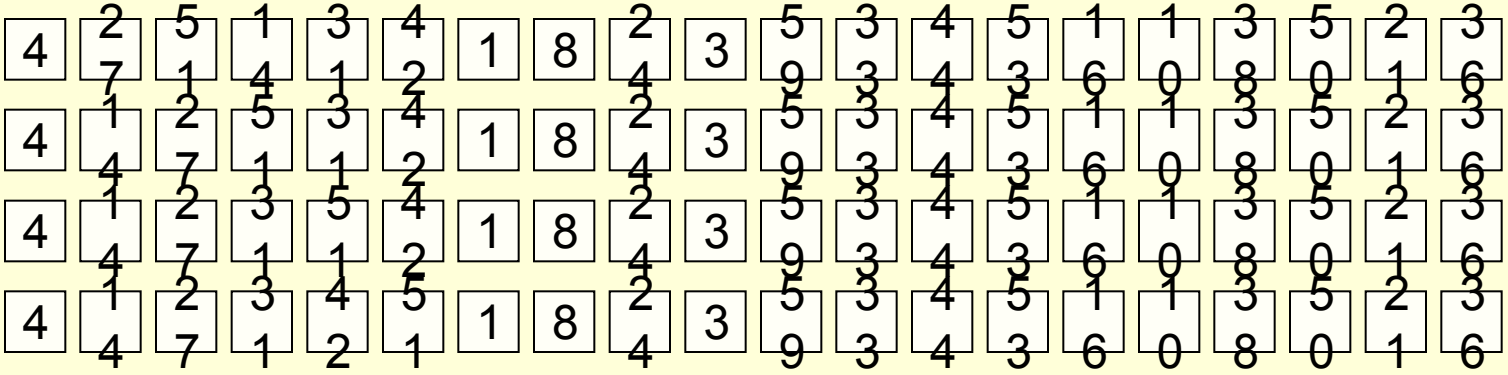
Это модификация «пузырьковой» сортировки, которая учитывает два дополнительных требования:

- 1) устранение «лишних» просмотров массива, т.е. если массив уже отсортирован за первые проходы, последующие проходы не делаем. Пример: 12,3,5,7,9,10.
- 2) смена направлений прохода массива: сначала проходим от начала к концу, по том – от конца к началу, потом снова от начала к концу и т.д. Это позволяет уменьшить число проходов по массиву. Пример: 5,7,9,10,12,3.

Сортировка простым выбором

```
void Sort_vybor(int data[], int rzm_data)
 { int i,j,k,x;
for (i=rzm_data-1; i>0;i--)
 {
    k = i;
    for (j=0; j<i; j++)
        if (data[j]>data[k])  k=j;
    if (k!=i)
     { x = data[k];
       data[k] = data[i];
       data[i] = x;
     }
 }
 }
```

Сортировка простыми вставками (простыми включениями)




```
void simpleInsertionSort(int data[], int rzm_data)
{   int  i,j,c;
    for (i=1; i<rzm_data; i++)
     { c = data[i];
       j = i-1;
       while (j>=0 && data[j]>c)
       {   data[j+1] = data[j];
           j--;
       }
       data[j+1] = c;
     }
}
```

Сортировка двоичными (бинарными) вставками

```
void BinaryInsertionSort(int data[], int rzm_data)
{   int  i,j,left,right,sred,x;
     for (i=1; i<rzm_data; i++)
         if (data[i-1]>data[i])
            {   x=data[i];
               left=0;
               right=i-1;
               do
                 {   sred=(left+right)/2;
                      if (data[sred]<x)   left= sred+1;
                         else right= sred-1;
                 }
               while (left<=right);
               for (j=i-1; j>=left; j--)  data[j+1]= data[j];
               data[left]=x;
            }
}
```

Сортировка Шелла (сортировка с убывающим шагом)

Идея метода: делим массив на k1 групп, каждую группу сортируем, далее делим массив на k2 групп (k2<k1), снова сортируем полученные группы, далее на k3 групп (k3<k2<k1), и т.д. в конце делим на kn групп, причем kn = 1, т.е. в конце сортируем весь массив.

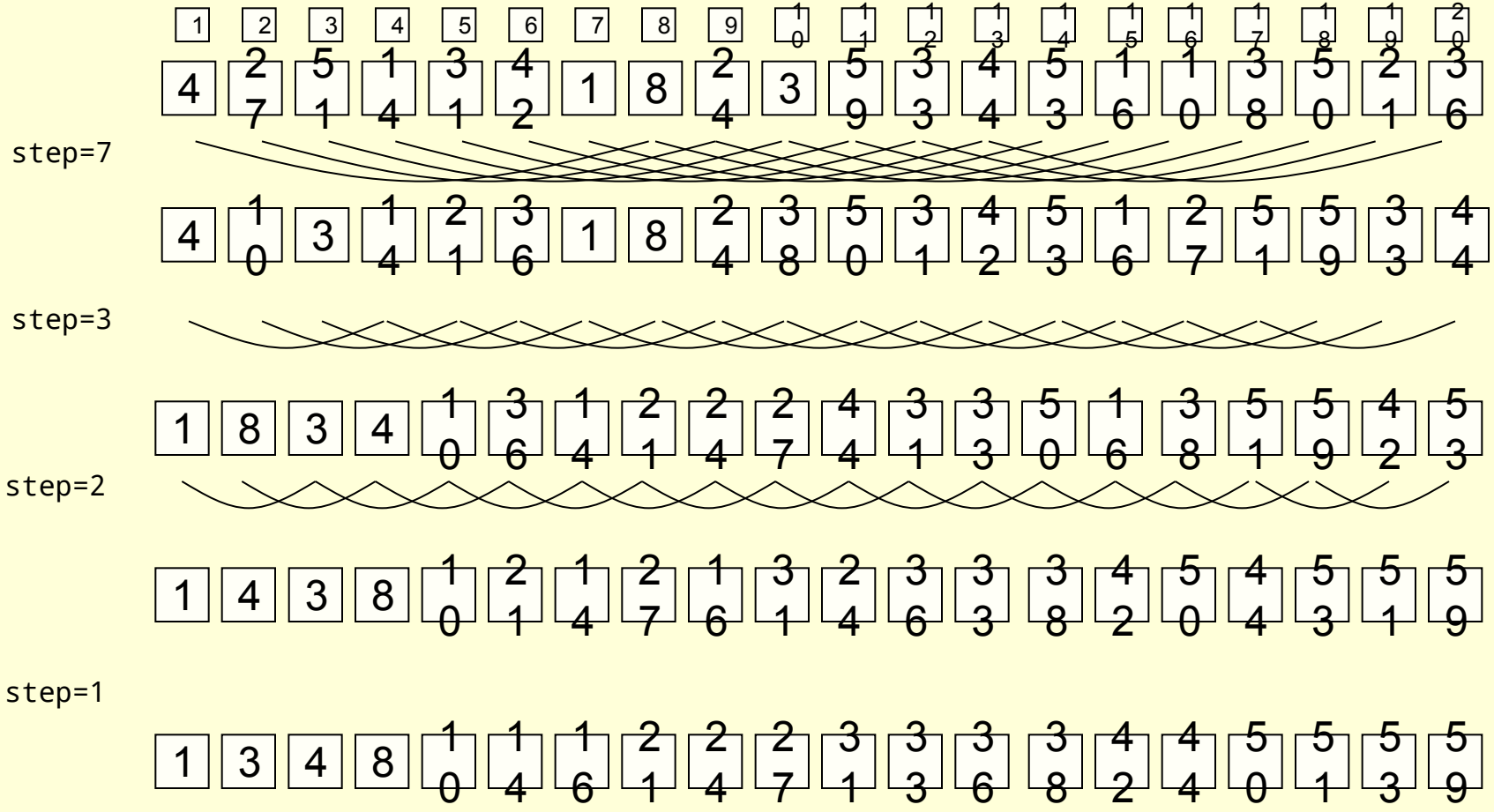
Значения k1, k2, k3, …, kn называются приращениями.

Метод предложен Д.Л.Шеллом в 1959 году.

Д.Кнут дает оценку метода при грамотном выборе значений приращений – $O(n^{1,2})$.

Лучшим считается выбор в качестве значений приращений взаимно простых чисел.

Сортировка Шелла



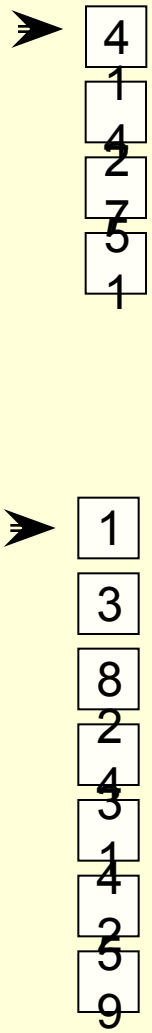
Сортировка Шелла

```
void ShellSort(int data[], int rzm_data)
{  int step,i,j,c;
    step = rzm_data;            // Шаг поисков и вставки
    do
    {  // Вычисляем новый шаг
        step = step / 3 + 1;
    // Производим сортировку простыми вставками с заданным шагом
        for (i=step; i<rzm_data; i++)
         {   c = data[i];
             j = i-step;
            while (j >= 0 && data[j] > c)
              {  data[j+step] = data[j];
                 j = j-step;
              }
            data[j+step] = c;
         }
     }
    while (step !=1);
}
```

Количество перестановок элементов
(по результатам экспериментов со случайным массивом)

	n = 25	n = 1000	n = 100000
Сортировка Шелла	50	7700	2 100 000
Сортировка простыми вставками	150	240 000	2.5 млрд.

Алгоритм слияния упорядоченных массивов




```
void merge(int a[], int b[], int na, int nb,
                int c[], int &nc)
 {  int ia,ib,ic;
    ia = ib = ic = 0;
    while (ia < na && ib < nb)
    {   if (a[ia]<b[ib])
         {  c[ic] = a[ia];
            ia++;
         }
        else
         {  c[ic] = b[ib];
            ib++;
         }
        ic++;
    }
    while (ia < na)
    { c[ic] = a[ia];
      ia++; ic++;
     }
    while (ib < nb)
     { c[ic] = b[ib];
       ib++; ic++;
     }
    nc = ic;
 }
```

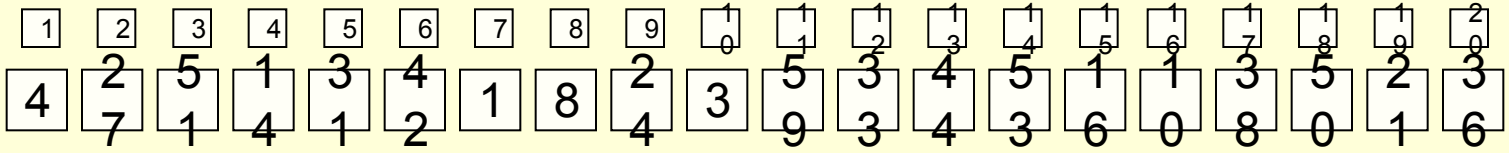
Сортировка фон Неймана (слиянием)

Метод основан на идее слияния двух отсортированных частей массива, поэтому первоначально массив разбивается на две части, которые независимо сортируются, а затем результаты сливаются в единое целое. С каждой из частей выполняются аналогичные действия, до тех пор, пока количество элементов в сортируемой части массива не станет равно двум. В этом случае выполняется простое сравнение элементов и их перестановка, если она необходима.

Метод слияний – один из первых в теории алгоритмов сортировки. Он предложен Дж. Фон Нейманом в 1945 году.

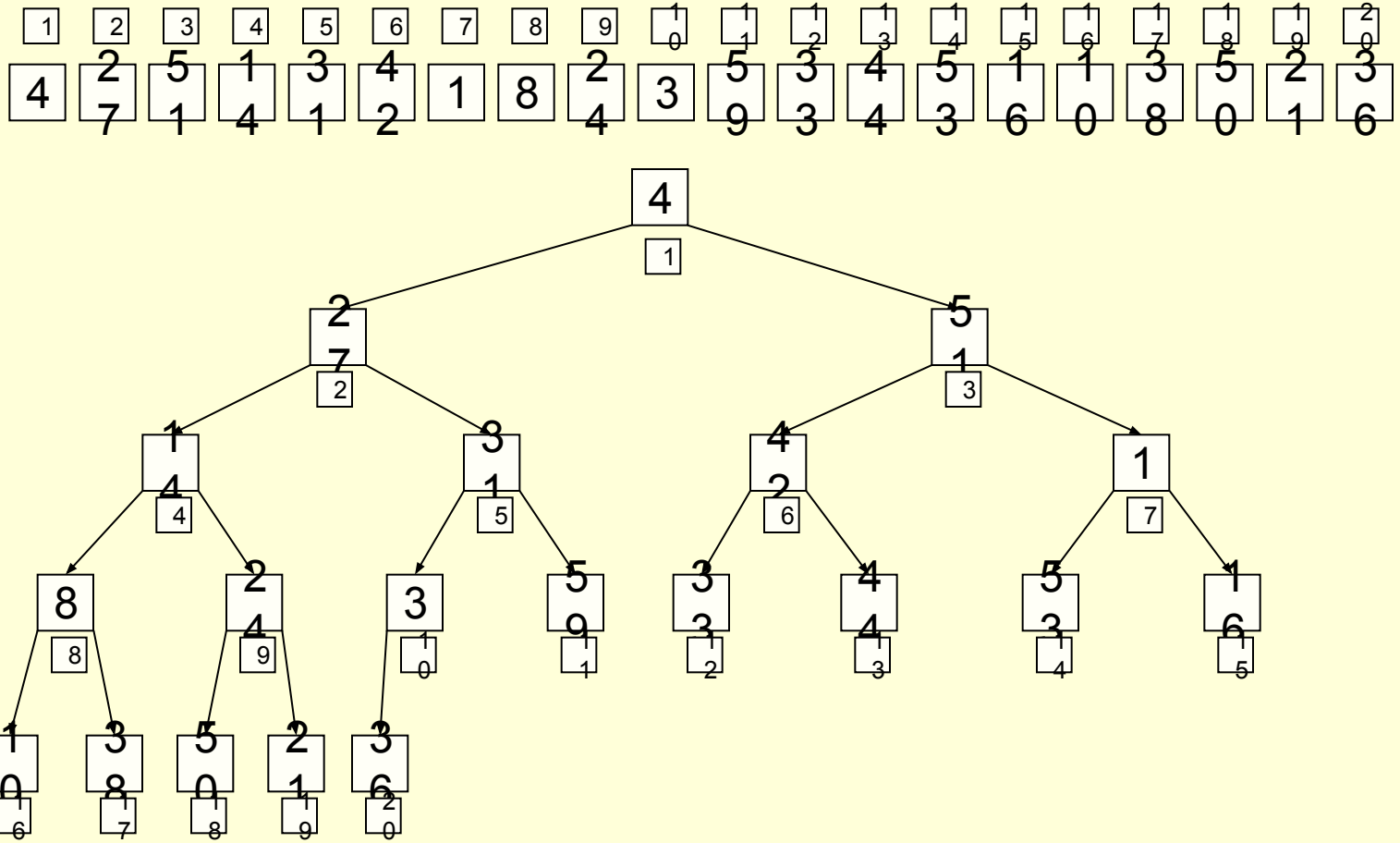
В алгоритме метода реализован один из фундаментальных принципов информатики – «разделяй и властвуй».

Сортировка фон Неймана (слиянием)



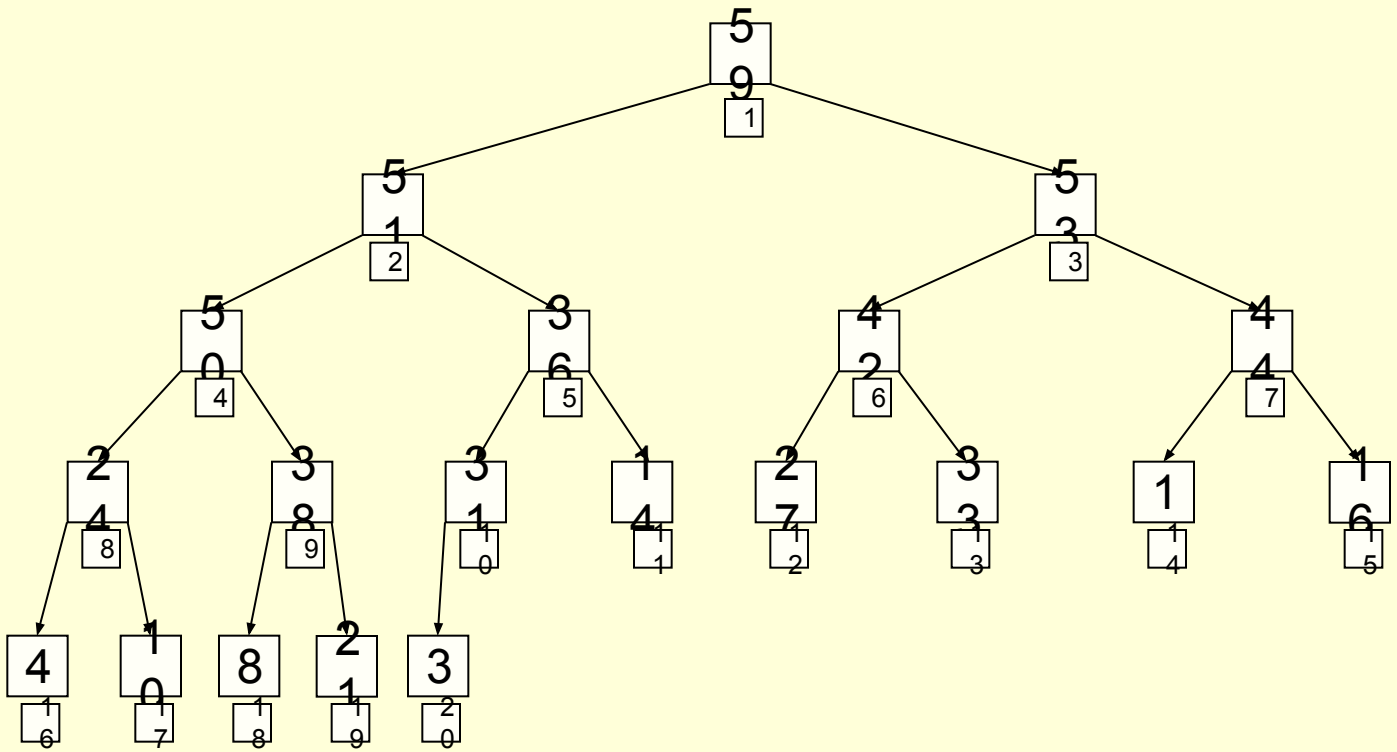
И так далее…

Пирамидальная сортировка (сортировка «кучей»)



И так далее…

Пирамидальная сортировка (продолжение)



И так далее…

Быстрая сортировка (сортировка Хоара)

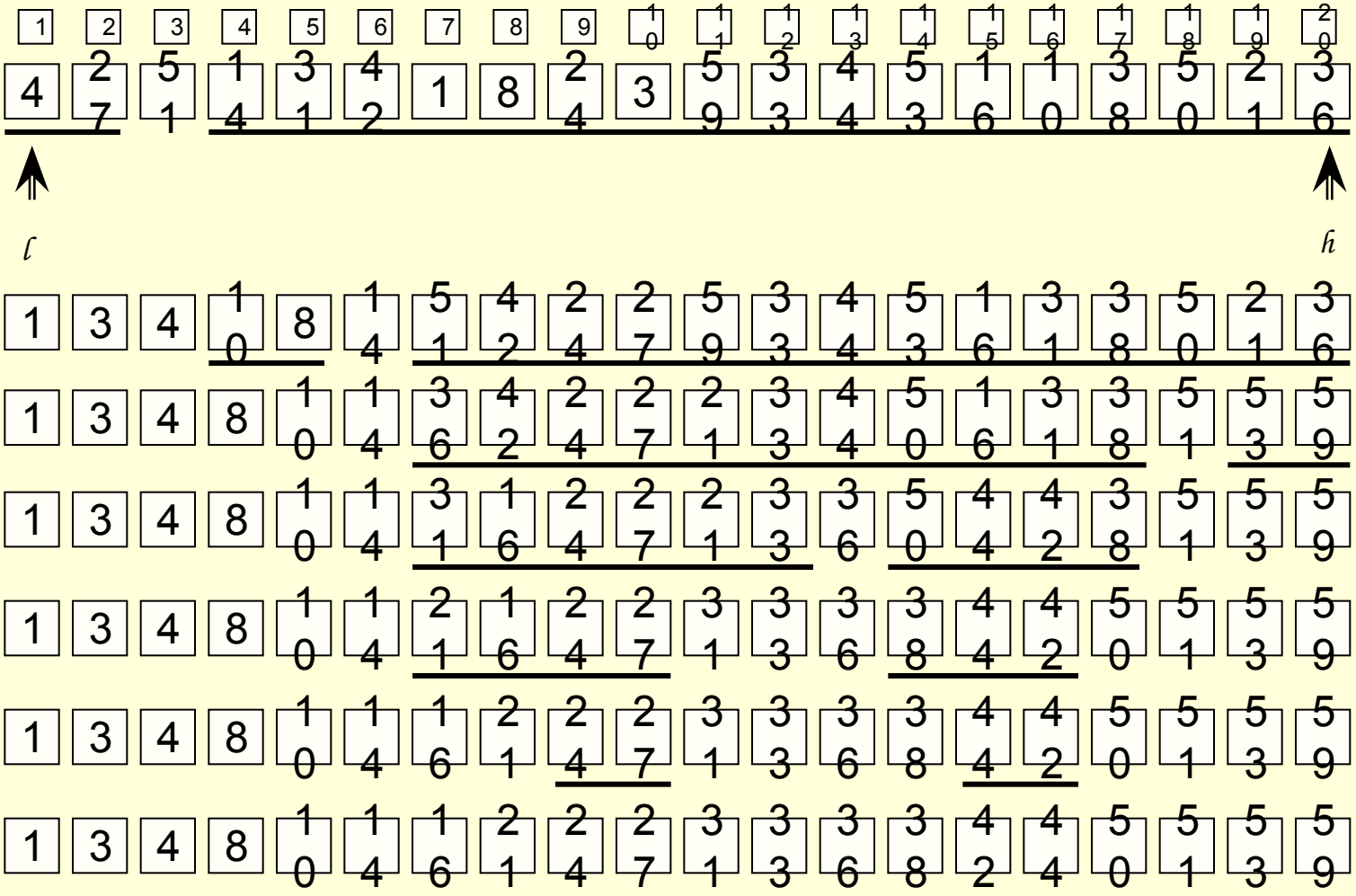
В алгоритме быстрой сортировки сочетаются три идеи:

- 1) разделение сортируемого массива на две части, левую и правую;
- 2) взаимное упорядочение двух подмассивов так, чтобы все элементы левого подмассива не превосходили элементов правого подмассива;
- 3) рекурсия, при которой подмассив упорядочивается точно таким же способом, как и весь массив.

Разделение массива на два взаимно упорядоченных подмассива:

- Обозначим за X элемент, находящийся посередине массива M
- Осуществляем два встречных просмотра массива: двигаемся слева направо, пока не найдем элемент M[i]>X и справа налево, пока не найдем элемент M[j]<X. Эти элементы «нарушают порядок»!
- Меняем местами элементы «нарушающие порядок»
- Продолжаем процесс "встречных просмотров с обменами", пока два идущих навстречу друг другу просмотра не встретятся.

Быстрая сортировка (сортировка Хоара)

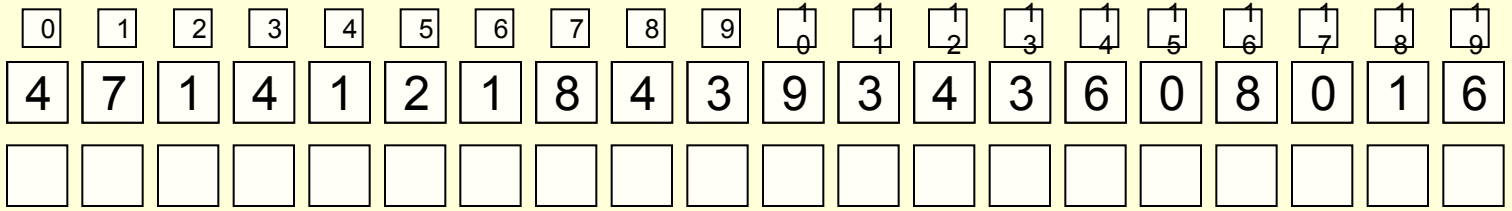


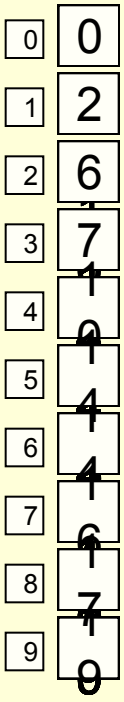
Быстрая сортировка (сортировка Хоара)

```
void quicksort(int A[], int L, int R)
{   int i,j,bar,tmp;
    i=L;
    j=R;
    bar=A[(L+R)/2];
    do
     {  while  (A[i]<bar)  i++;
        while  (bar<A[j])  j--;
        if (i<=j)
         {  if (i<j) { tmp=A[i]; A[i]=A[j]; A[j]=tmp; }
            i++; j--;
         }
     }
    while (i<=j);
    if (L<j) quicksort(A,L,j);
    if (i<R) quicksort(A,i,R);
}
```

Вызов в основной программе: `quicksort(a,0,n-1);`

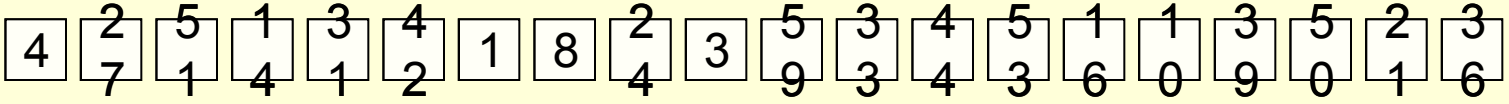
Сортировка подсчетом



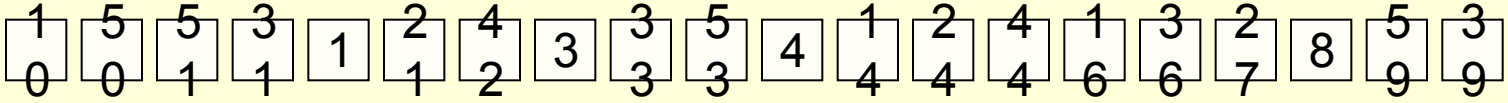



```
for (i=0; i< k; i++)
    C[i]=0;
for (i=0; i<n; i++)
    C[A[i]]=C[A[i]] + 1;
for (j=1; j<k; j++)
    C[j]=C[j] + C[j - 1];
for (i=n-1; i>=0; i--)
 {  C[A[i]] = C[A[i]] - 1;
    B[C[A[i]]] = A[i];
 }
```

Цифровая сортировка



После сортировки по последней цифре:



После устойчивой сортировки по первой цифре: