Структуры и алгоритмы обработки данных, 2

Лекция 2 (+3)

Сортировка

(новый раздел)

1. Постановка задачи
2. Простые алгоритмы сортировки
   - Сортировка выбором
   - Сортировка обменами
   - Сортировка вставками
3. Быстрая сортировка
   - Процедура разделения
   - Алгоритм QuickSort
   - Модификации
4. Анализ алгоритма

Сортировка 2

Дан массив a: **array** [1..n] **of** Ordinal

Отрезок (сегмент) массива a[p..q] ***упорядочен***:

Sort (a, p, q) ≡ (∀ i: p ≤ i < q: a[i] **≤** a[i+1])

Предусловие: a[1..n] = a0[1..n]

Постусловие: Sort (a, 1, n) & &
Перестановка(a[1..n], a0[1..n]),

где *Перестановка*(a[1..n], a0[1..n]) ≡

≡ (∀ i: 1 ≤ i ≤ n: (**N** j: 1 ≤ j ≤ n: a[i] = a0[j]) =

= (**N** j: 1 ≤ j ≤ n: a[i] = a [j]))

Сортировка 3

Простые алгоритмы сортировки

Сортировка выбором

Идея. Пример.

Список (удаление и вставка)

упОрядочИвание	авдеИинупОряоч
аупОрядочИвние	авдеИинОупряоч
авупОрядочИние	авдеИинОоупряч
авдупОряочИние	авдеИинОопуряч
авдеупОряочИни	авдеИинОопруяч
авдеИупОряочни	авдеИинОопруяч
авдеИиупОряочн	авдеИинОопручя

Устойчивость сортировки – сохранение относительного порядка элементов с равными ключами

Сортировка 4

Сортировка выбором.

Пример.

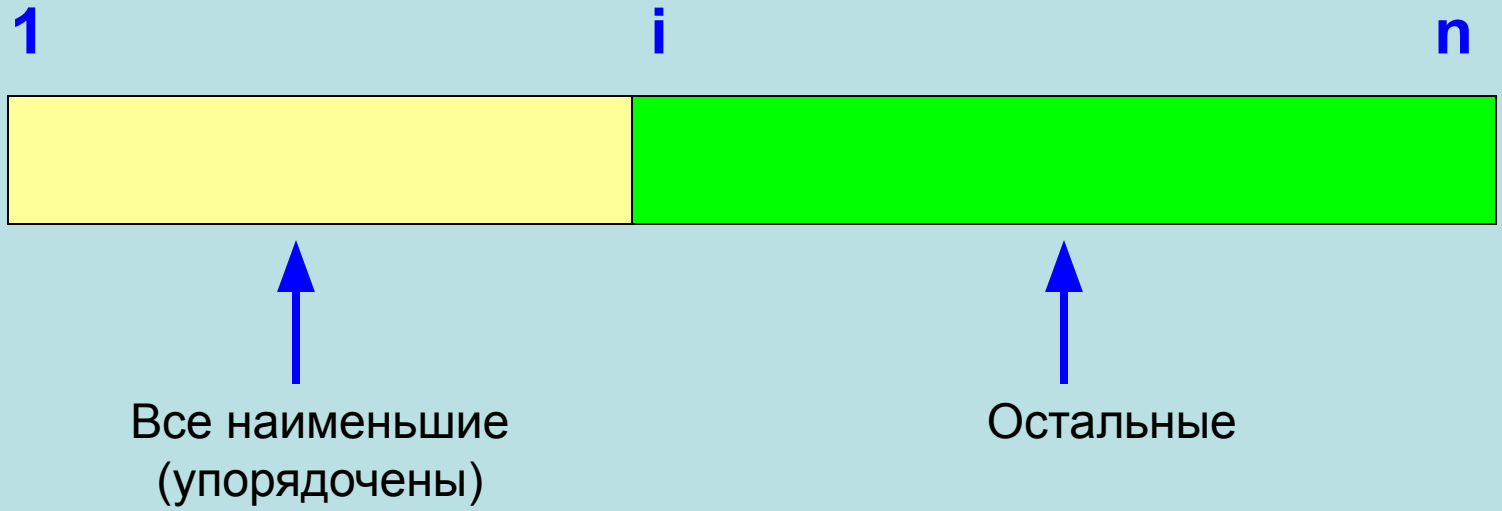
Массив (обмен элементов).

упОрядочИвание	авдеИинчяпуоОр
апОрядочИвуние	авдеИинояпучОр
авОрядочИпуние	авдеИиноОпучяр
авдряОочИпуние	авдеИиноОпучяр
авдеяОочИпунир	авдеИиноОпрчяу
авдеИОочяпунир	авдеИиноОпруяч
авдеИиочяпунОр	авдеИиноОпручя

Простые алгоритмы сортировки

Сортировка выбором

Инвариант внешнего цикла:



$$(a[1..i) \le a[i..n]) \;\&\; Sort\,(a, 1, i-1) \;\&\; (1 \le i \le n)$$

Если во внутреннем цикле ищем первое вхождение минимума, то сортировка в списке *устойчивая*, а в массиве *неустойчивая*

Сортировка выбором

```
for i:=1 to n−1 do
begin
    {поиск минимума из a[i..n]}
    k := i; min := a[i];
    for j:=i+1 to n do
    if a[j] < min then
    begin k := j; min := a[k] end;
    {обмен a[k] ↔ a[i] }
     a[k] := a[i];  a[i] := min
end {for}
```

Анализ сортировки выбором

C_i – число ***сравнений*** при выборе минимального элемента на i-ом шаге

(в сегменте из $(n - i + 1)$ элементов)

$$C_i = n - i$$

Суммарно по всем шагам:

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

Анализ сортировки выбором

Пусть M_i – число ***перестановок*** на i-ом шаге, включая обновления текущего минимума (в сегменте из $(n - i + 1)$ элементов).

В ***худшем*** *случае* (обратно упорядоченный массив):

$$M_i = (n - i) + 3$$

Тогда суммарно по всем шагам ($i \in 1..n-1$):

$$M_{max} = (n^2 - n)/2 + 3(n - 1)$$

Анализ сортировки выбором

В среднем:

Вероятность того, что произойдет обновление

текущего минимума

на j-ом шаге внутреннего цикла = 1/j

(любой, в том числе последний из j элементов - минимальный).

Т.о. за i-й шаг среднее (М.О.) число перестановок

$= 1/2 + 1/3 + 1/4 + \ldots + 1/(n-i+1) = H_{n-i+1} - 1,$

где частичная сумма гармонического ряда

$H_k = 1 + 1/2 + 1/3 + \ldots + 1/k,$

$H_k = \ln k + \gamma + 1/(2k) + \ldots$, где γ - постоянная Эйлера.

Итак, за i-й шаг внешнего цикла среднее число присваиваний

$H_{n-i+1} - 1 + 3 = H_{n-i+1} + 2 \approx \ln (n-i+1) + \gamma + 2$

Анализ сортировки выбором *в среднем*

В среднем за весь внешний цикл суммарное число перестановок есть

$$M = \sum_{i=1}^{n-1} [\ln(n-i+1) + (\gamma + 2)] =$$

$$= \sum_{i=1}^{n} \ln i + (n-1)(\gamma + 2) = ?$$

$$\sum_{i=1}^{n} \ln i \approx \int_{1}^{n} \ln x dx = (x \ln x - x)_1^n = n \ln n - n + 1$$

$$? \approx n \ln n + n(\gamma + 1)$$

Простые алгоритмы сортировки

Сортировка обменами («пузырьковая»)

1 проход	*2 проход*	*3 проход*	*4 проход*
упОрядочИвание	пОрудочИваниея	ОпрдоуИваниечя	ОпдорИваниеучя
пуОрядочИвание	ОпрудочИваниея	ОпрдоуИваниечя	ОпдорИваниеучя
пОурядочИвание	ОпрудочИваниея	ОпрдоуИваниечя	ОдпорИваниеучя
пОруядочИвание	ОпрудочИваниея	ОпдроуИваниечя	ОдопрИваниеучя
пОруядочИвание	ОпрдуочИваниея	ОпдоруИваниечя	ОдопрИваниеучя
пОрудяочИвание	ОпрдоучИваниея	ОпдоруИваниечя	ОдопИрваниеучя
пОрудоячИвание	ОпрдоучИваниея	ОпдорИуваниечя	ОдопИвраниеучя
пОрудочяИвание	ОпрдоуИчваниея	ОпдорИвуаниечя	ОдопИварниеучя
пОрудочИявание	ОпрдоуИвчаниея	ОпдорИвауниечя	ОдопИванриеучя
пОрудочИвяание	ОпрдоуИвачниея	ОпдорИвануиечя	ОдопИваниреучя
пОрудочИваяние	ОпрдоуИванчиея	ОпдорИваниуечя	ОдопИваниеручя
пОрудочИваняие	ОпрдоуИваничея	ОпдорИваниеучя	
пОрудочИваниеяе	ОпрдоуИваниечя		
пОрудочИваниеея			

Простые алгоритмы сортировки

Сортировка обменами («пузырьковая»)

Продолжение

5 проход	*6 проход*	*7 проход*	*8 проход*
ОдопИваниеручя	дОоИваниепручя	дОИваниеопручя	дИваниеОопручя
дОопИваниеручя	дООоИваниепручя	дИОваниеопручя	дИваниеОопручя
дОопИваниеручя	дОоИваниепручя	дИОваниеопручя	двИаниеОопручя
дОопИваниеручя	дОИованиепручя	дИвОаниеопручя	дваИниеОопручя
дООпИваниеручя	дОИвоаниепручя	дИваОниеопручя	дваИниеОопручя
дОоИвпаниеручя	дОИваониепручя	дИванОиеопручя	дваИиneОопручя
дОоИвапниеручя	дОИваноиепручя	дИвани Оеопручя	дваИиенОопручя
дОоИванпиеручя	дОИвани оепручя	дИваниеОопручя	
дОоИваниперучя	дОИваниеопручя		
дОоИваниепручя			

Простые алгоритмы сортировки

Сортировка обменами («пузырьковая»)

Продолжение

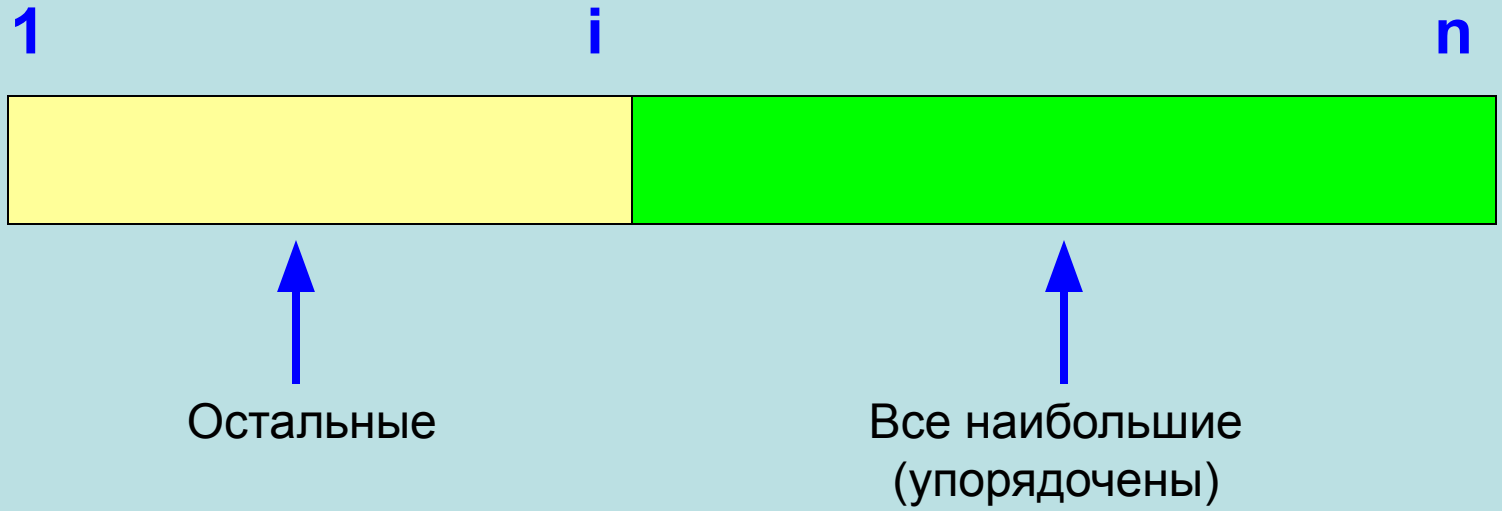
9 проход	*10 проход*	*11 проход*
двaИиенОопручя	вадИеинОопручя	авдеИинОопручя
вдаИиенОопручя	авдИеинОопручя	авдеИинОопручя
вадИиенОопручя	авдИеинОопручя	авдеИинОопручя
вадИиенОопручя	авдИеинОопручя	авдеИинОопручя
вадИиенОопручя	авдеИинОопручя	----------
вадИеинОопручя		
		авдеИинОопручя

Простые алгоритмы сортировки

Сортировка обменами

Инвариант внешнего цикла

For i:=n downto 2 do



$$(a[1..i] \le a(i..n]) \;\&\; Sort\,(a, i+1, n) \;\&\; (1 \le i \le n)$$

Сортировка обменами (пузырьковая)

```
for i:=n downto 2 do
begin
  {просмотр a[1..i] с обменами соседних элементов}
  for j := 2 to i do
    if a[j−1] > a[j] then a[j−1] ↔ a[j] ;
  {a[i] = max a[1..i] }
end
```

Анализ сортировки обменами (пузырьковой)

$$C_i = i-1; \quad C = \sum_{i=2}^{n}(i-1) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}.$$

В худшем случае

$$M_i = 3(i-1); \; M = \frac{3}{2}n(n-1).$$

В среднем : $M_i = \frac{3}{2}(i-1); \; M = \frac{3}{4}n(n-1).$

Действительно :

$$\begin{Bmatrix} k_1, k_2, \ldots, k_i, \ldots, k_j, \ldots, k_{n-1}, k_n \\ k_n, k_{n-1}, \ldots, k_j, \ldots, k_i, \ldots, k_2, k_1 \end{Bmatrix}$$

$k_i \leftrightarrow k_j$ один раз на 2 последовательности.

Всего перестановок пар $C_2^n = \frac{n(n-1)}{2}$

и всего обменов $\frac{n(n-1)}{4}$

Простые алгоритмы сортировки

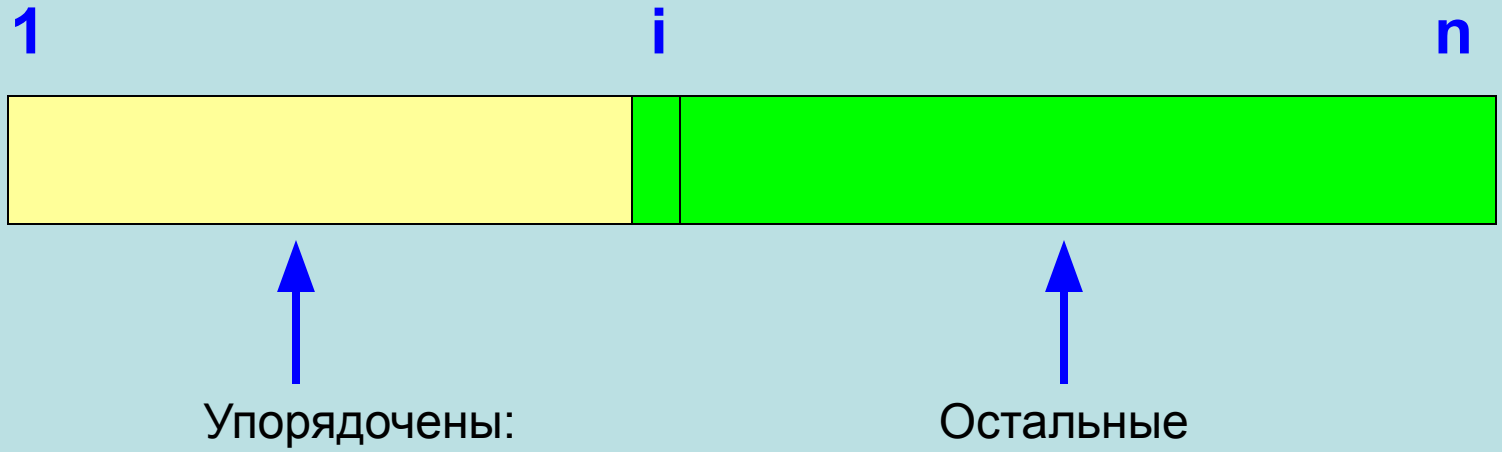
Сортировка вставками (включением)

упОрядочИвание	дОопручяИвание
пуОрядочИвание	дИОопручявание
Опурядочивание	вдИОопручяание
Опруядочивание	авдИОопручяние
Опруядочивание	авдИнОопручяие
дОпруяочИвание	авдИинОопручяе
дОопруячИвание	авдеИинОопручя

Простые алгоритмы сортировки

Сортировка ВСТАВКАМИ

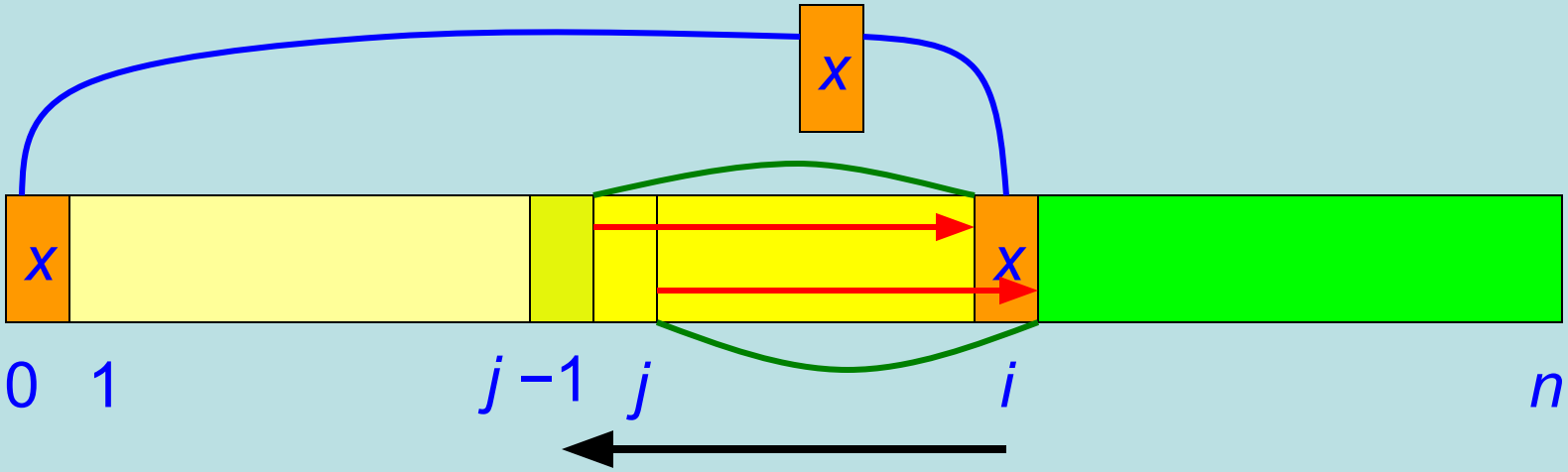
Инвариант внешнего цикла:



Sort (a, 1, i -1)

После i – го шага: Sort (a, 1, i)

Сортировка прямыми ВСТАВКАМИ



Sort (a, 1, i -1) & (x < a0[j..i-1]) & (a[j+1..i]=a0[j..i-1])

```
j := i;
while x < a[j -1] do
begin  a[j] := a[j -1];  j := j -1  end
{ (x ≥ a[j -1]) & (x < a [j+1..i]) & (a[j] – «свободен»)}
a[j] := x;
```

Простые алгоритмы сортировки

Сортировка ВСТАВКАМИ

АЛГОРИТМ *Прямые вставки*
(StraightInsertion)

```
type index=0..nMax;
     index1=1..nMax;
     index2=0..nMax+1;
     mass=array [index] of Integer;
```

```
procedure StraightInsertion ( var a: mass; n: index1; k: index1 );
 { сортировка массива методом вставки }
  var i: index;
      j: index1;
      x: Integer;
 begin
  for i:=2 to n do
   begin { включить a[i] на соотв. место в a[1..i] }
    x := a[i]; a[0] := x; { a[0] - "барьер" }
    j := i;
    while x < a[j-1] do
     begin  { 1<=j<=i }
      a[j] := a[j-1];
      j := j-1
     end { while };
    a[j] := x
   end { for }
 end { StraightInsertion };
```

Сортировка ПРЯМЫМИ ВСТАВКАМИ

Анализ

Число сравнений ≈ число перемещений.

Число сравнений на i –ом шаге $C_i = i/2$ в среднем.

Тогда по всем шагам $i \in 2..n$

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

$$= \frac{1}{4}(n(n+1) - 2) \approx \frac{1}{4}n^2$$

Сортировка БИНАРНЫМИ ВСТАВКАМИ

```
for i:=2 to n do
begin
   { Найти место a[i] среди a[1..i-1] }
   x := a[i]; L:=1; R:=i;
   while L≠R do
   begin
    m:=(L+R) div 2;
    if a[m] < x then L:=m+1 else R:=m
   end; {(L∈1..i) & (a[L-1] < x ≤ a[L])}
   {сдвинуть a[L..i-1] на позицию вправо на место a[L+1..i]}
   for j:=i downto L+1 do a[j] := a[j-1];
   {вставить x на место a[L]}
   a[L]:=x
end {for}
```

Сортировка БИНАРНЫМИ ВСТАВКАМИ Анализ алгоритма

Число сравнений на i –ом шаге $C_i \le \lceil \log_2 i \rceil$.

По всем шагам ($i \in 2..n$)

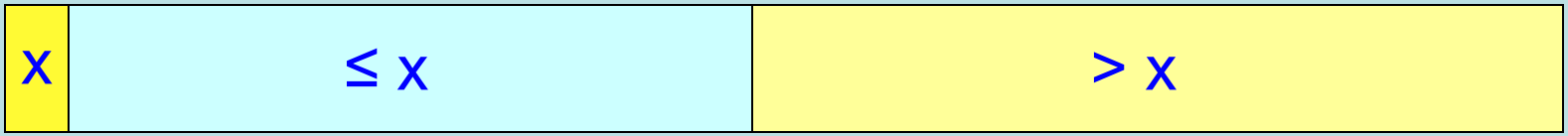
$$C = \sum_{i=1}^{n} \lceil \log_2 i \rceil \approx$$

$$\approx \sum_{i=1}^{n} \ln i \approx \int_{1}^{n} \ln x dx = (x \ln x - x)_1^n = n \ln n - n + 1$$

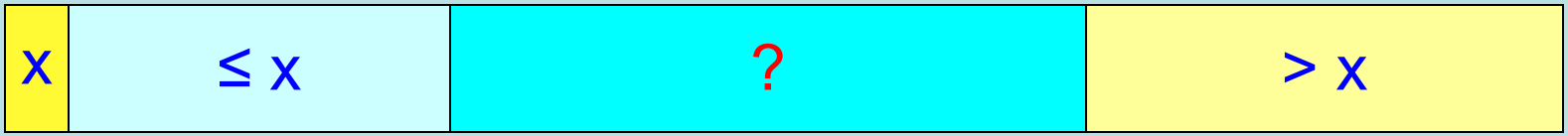
$$C \approx n \log_2 n$$

Быстрая сортировка (**QuickSort**)

Основа – процедура разделения Partition



Инвариант цикла:



& ($m < q \le p + 1 \le n + 1$)

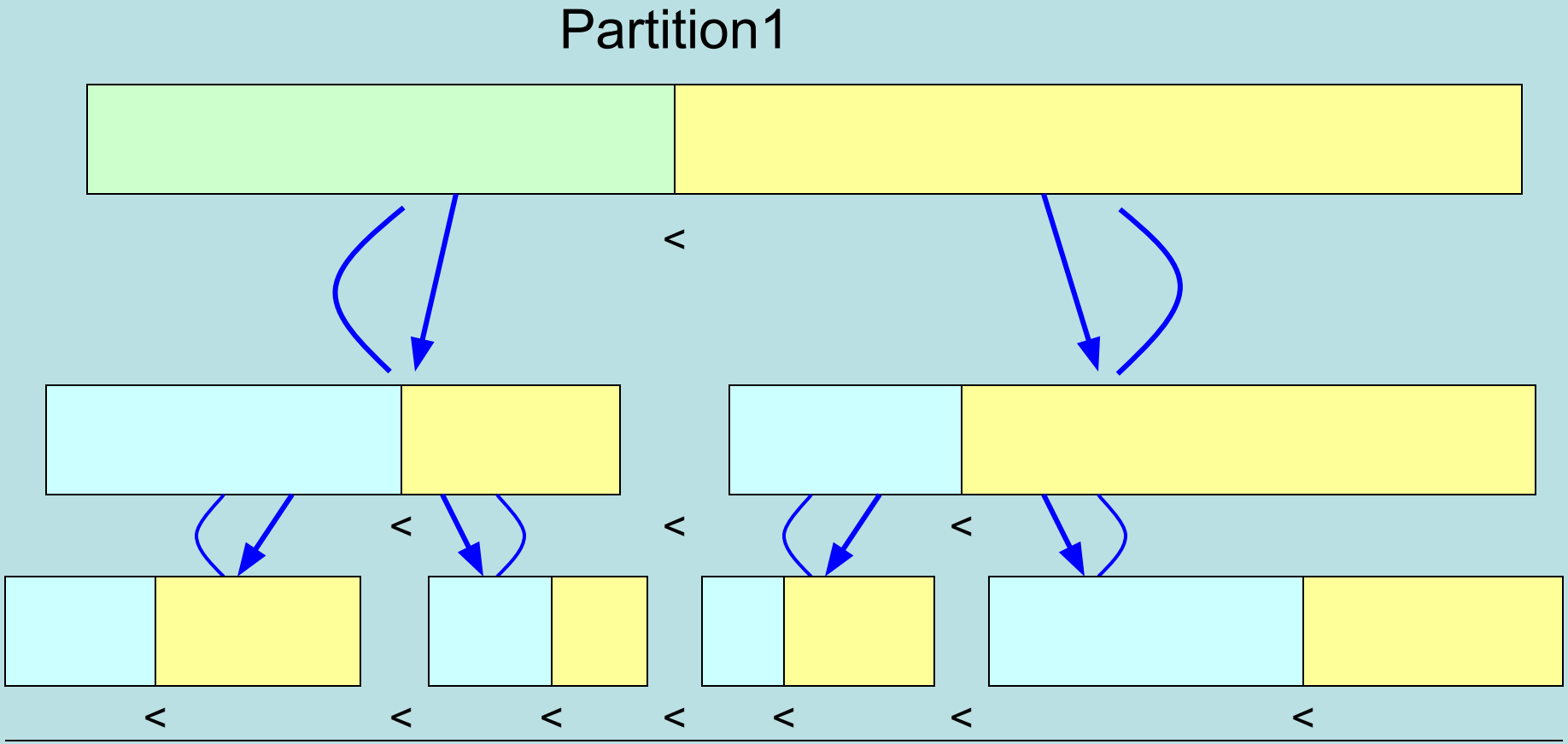
Условие завершения цикла $p + 1 = q$

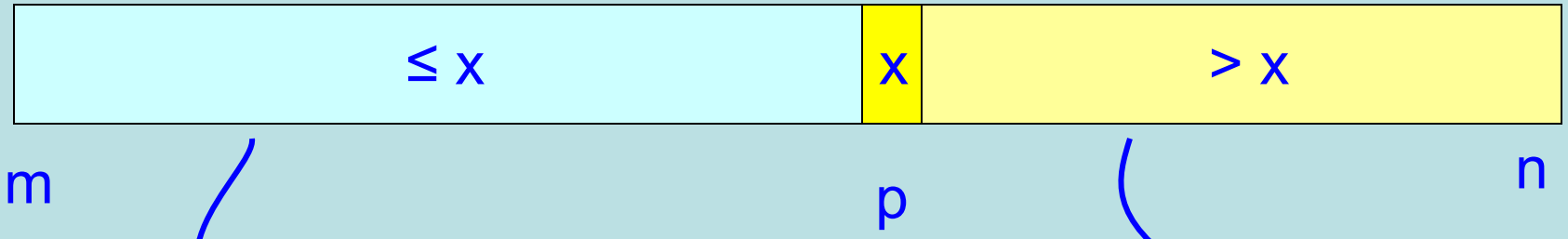
Быстрая сортировка (**QuickSort**)

```
procedure Partition1 ( var a : mass;
                                  m, n: index1; var p: index2);
  var  x, y:     integer;
       mn2, q: index2;
 begin  { m < n }
     mn2:=(m+n) div 2;
 x := a[mn2];   a[mn2]:= a[m];   a[m]:=x;
     q := m+1;       p := n;
 { inv:  (m<q<=(p+1)<=(n+1)) &
  (ALL k: m<=k<q: a[k]<=x) & (ALL k: p<k<=n: a[k]>x)}
```

```
while q<= p do
  begin
      if a[q] <= x then q := q+1
       else
        if a[p] > x then p := p-1
         else { a[p]<= x < a[q] }
          begin
           y    := a[p];
           a[p] := a[q];
           a[q] := y;
           q := q+1;
           p := p-1
          end { if };
  end { while };
  a[m] := a[p];
  a[p] := x;
end { Partition1 };
```

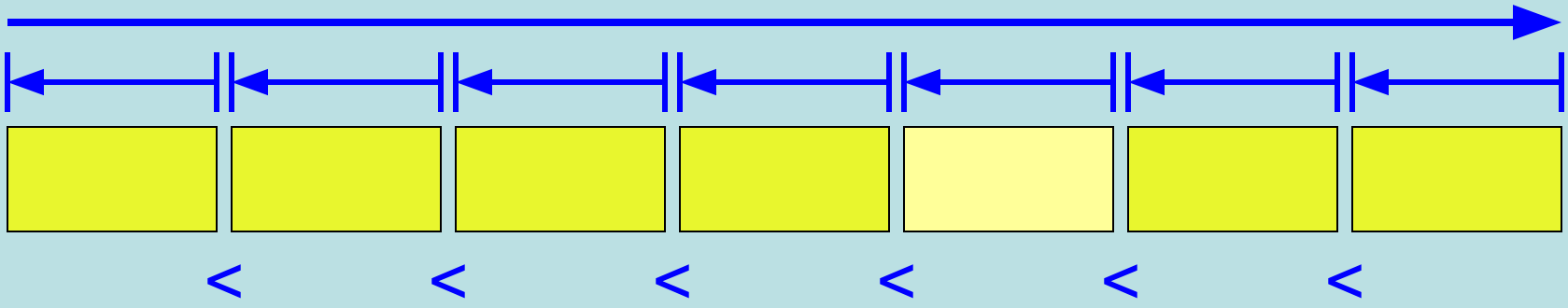
Быстрая сортировка:
разделение и слияние





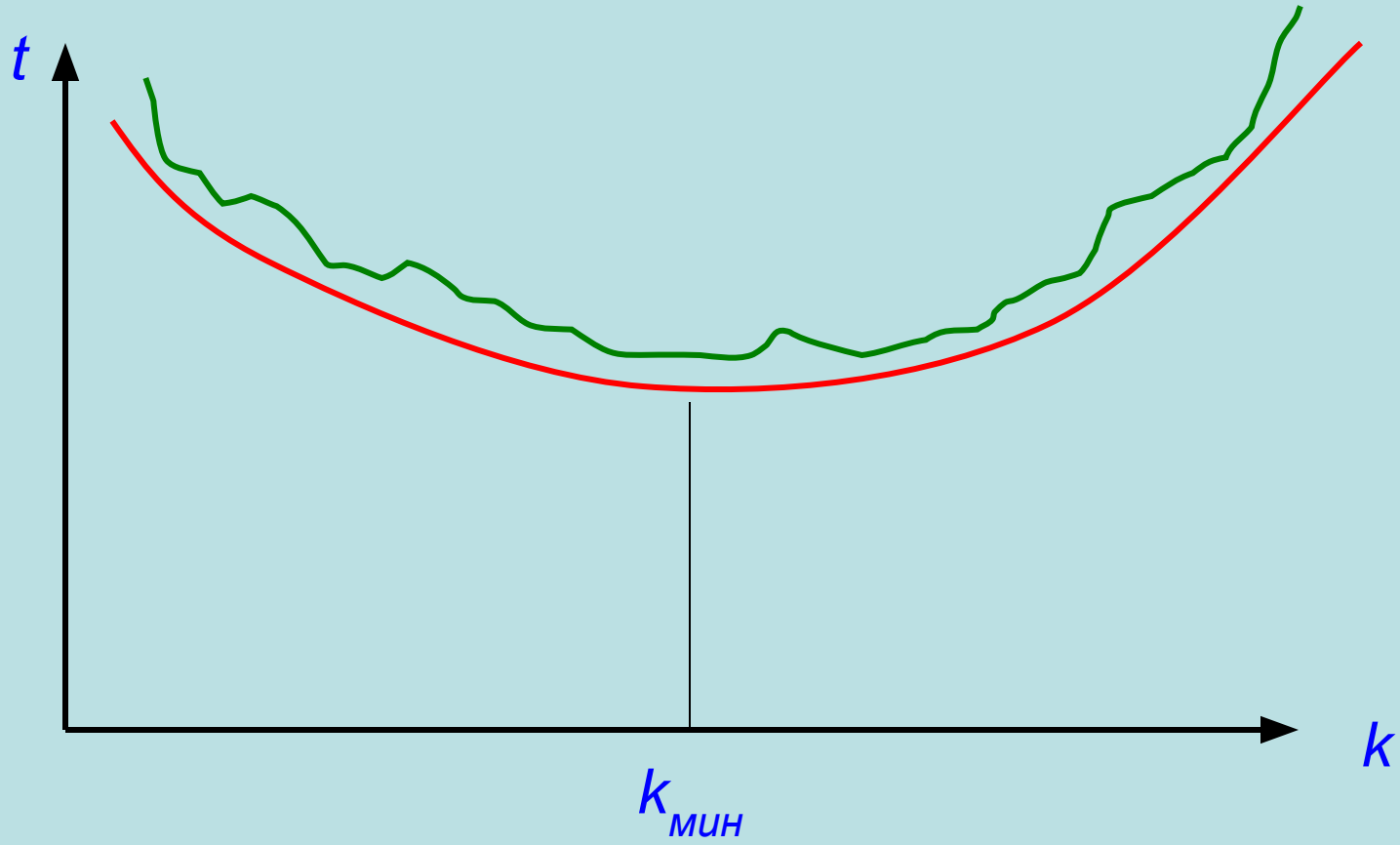

```
procedure Sortp1 (var a: mass; m, n: index1; k: index1 );
    var p, q: index2;
  begin
      Partition1 ( a, m, n, p );
      if p-m>k then  Sortp1 ( a, m, p-1, k );
       q:=p+1;
      if n-q+1>k then  Sortp1 ( a, q, n, k )
    { подмассив длиной <= k  не сортируем !
        1<=k<=50 , при k=1 - полная сортировка }
  end { Sortp1 };
```

- **begin** { QuickSort1 }
- Sortp1(a, 1, n, k);
- **if** k>1 **then** { окончательная сортировка
- простым методом }
- StraightInsertion (a, n ,1)
- **end** { QuickSort1 };



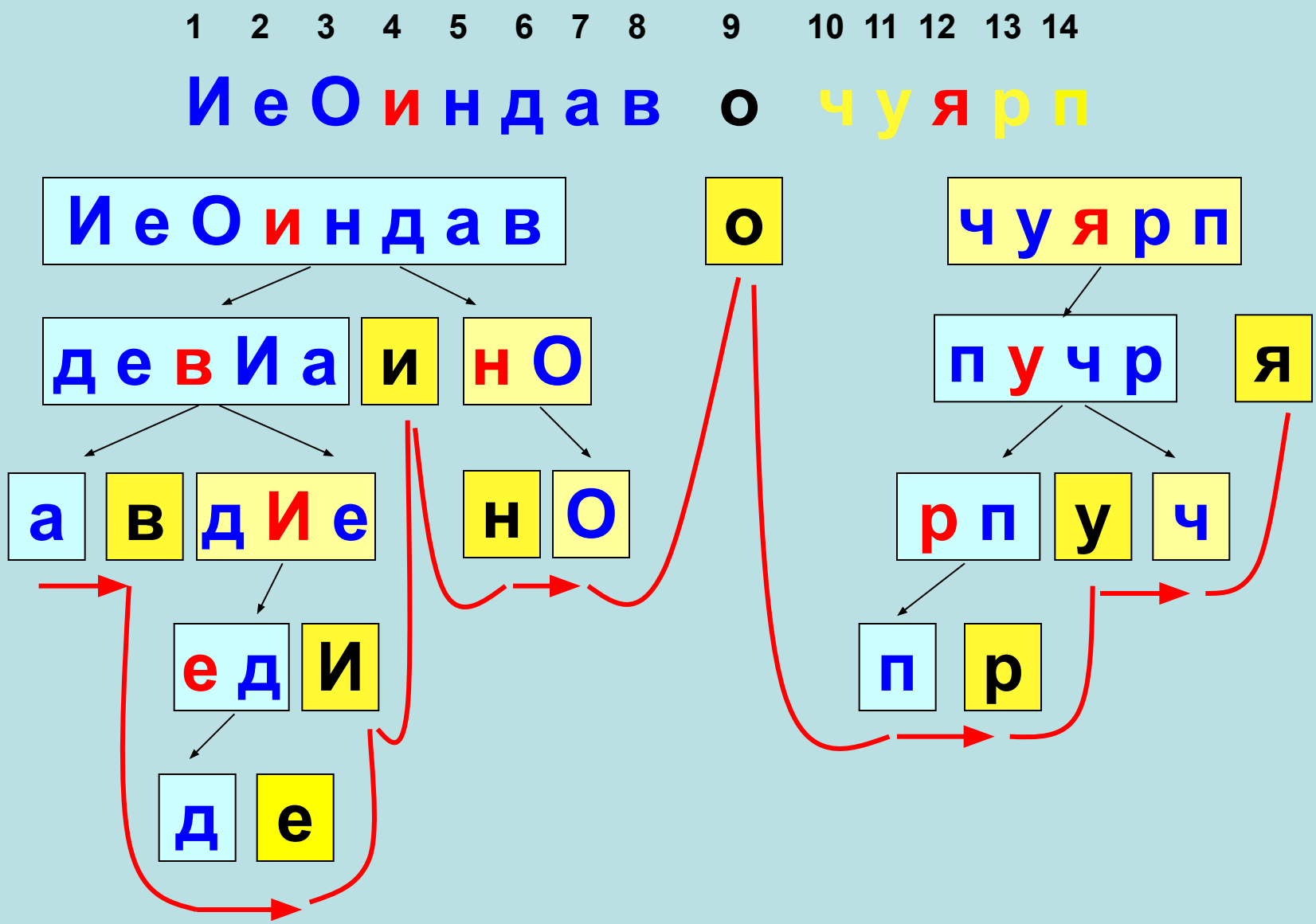
Быстрая сортировка

Неполная сортировка. Выбор *k*



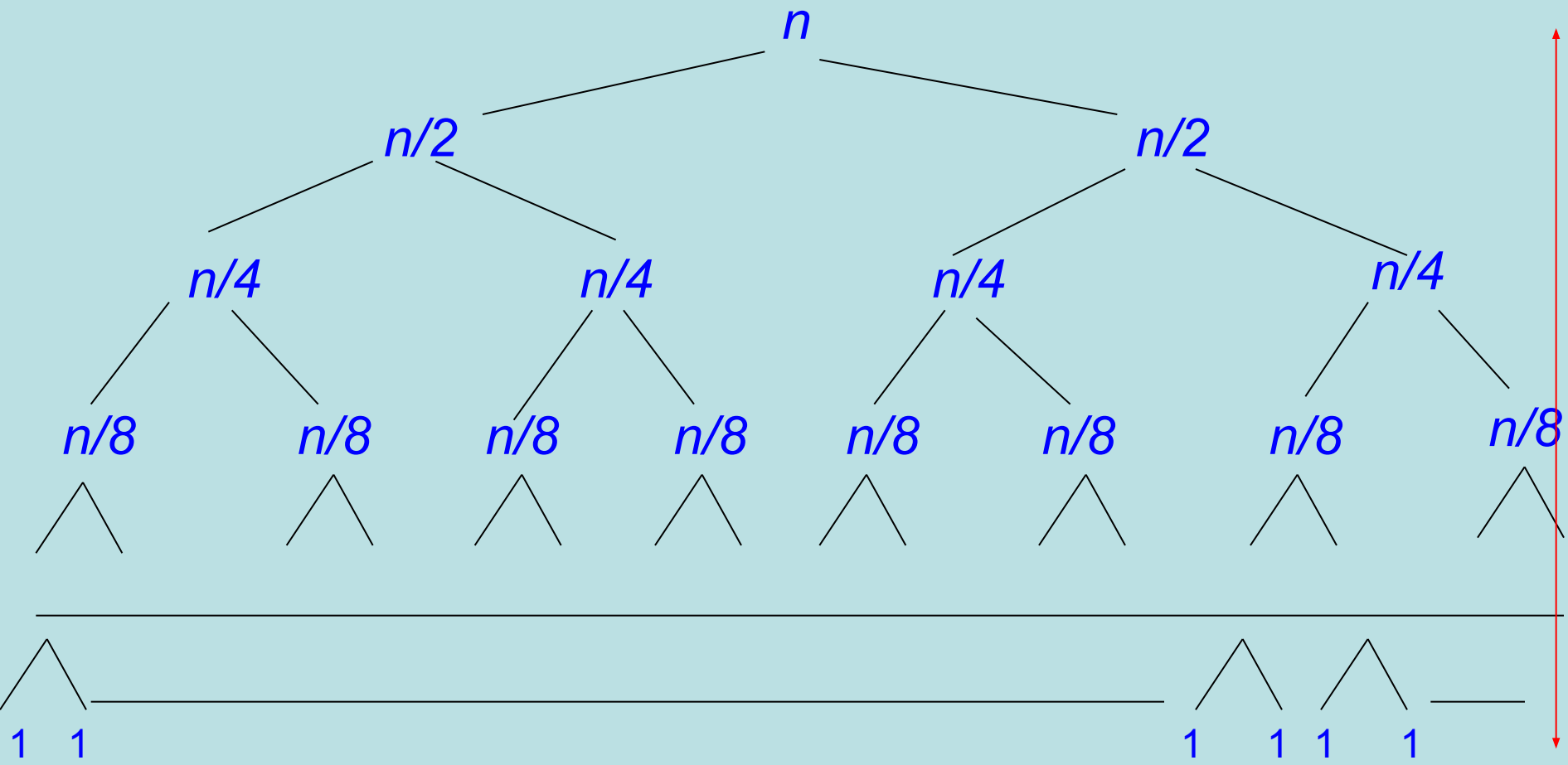
Первый вызов Partition

1	2	3	4	5	6	7	8	9	10	11	12	13	14
у	п	О	р	я	д	о	ч	И	в	а	н	и	е
о	п	О	р	я	д	у	ч	И	в	а	н	и	е
о	е	О	р	я	д	у	ч	И	в	а	н	и	п
о	е	О	р	я	д	у	ч	И	в	а	н	и	п
о	е	О	и	я	д	у	ч	И	в	а	н	р	п
о	е	О	и	н	д	у	ч	И	в	а	я	р	п
о	е	О	и	н	д	у	ч	И	в	а	я	р	п
о	е	О	и	н	д	а	ч	И	в	у	я	р	п
о	е	О	и	н	д	а	в	И	ч	у	я	р	п
И	е	О	и	н	д	а	в	о	ч	у	я	р	п


1 2 3 4 5 6 7 8 9 10 11 12 13 14
И е О и н д а в о ч у я р п
И е О и н д а в
о
ч у я р п
д е в И а
и
н О
п у ч р
я
а
в
д И е
н
О
р п
у
ч
е д
И
п
р
д
е

Быстрая сортировка (**QuickSort**)

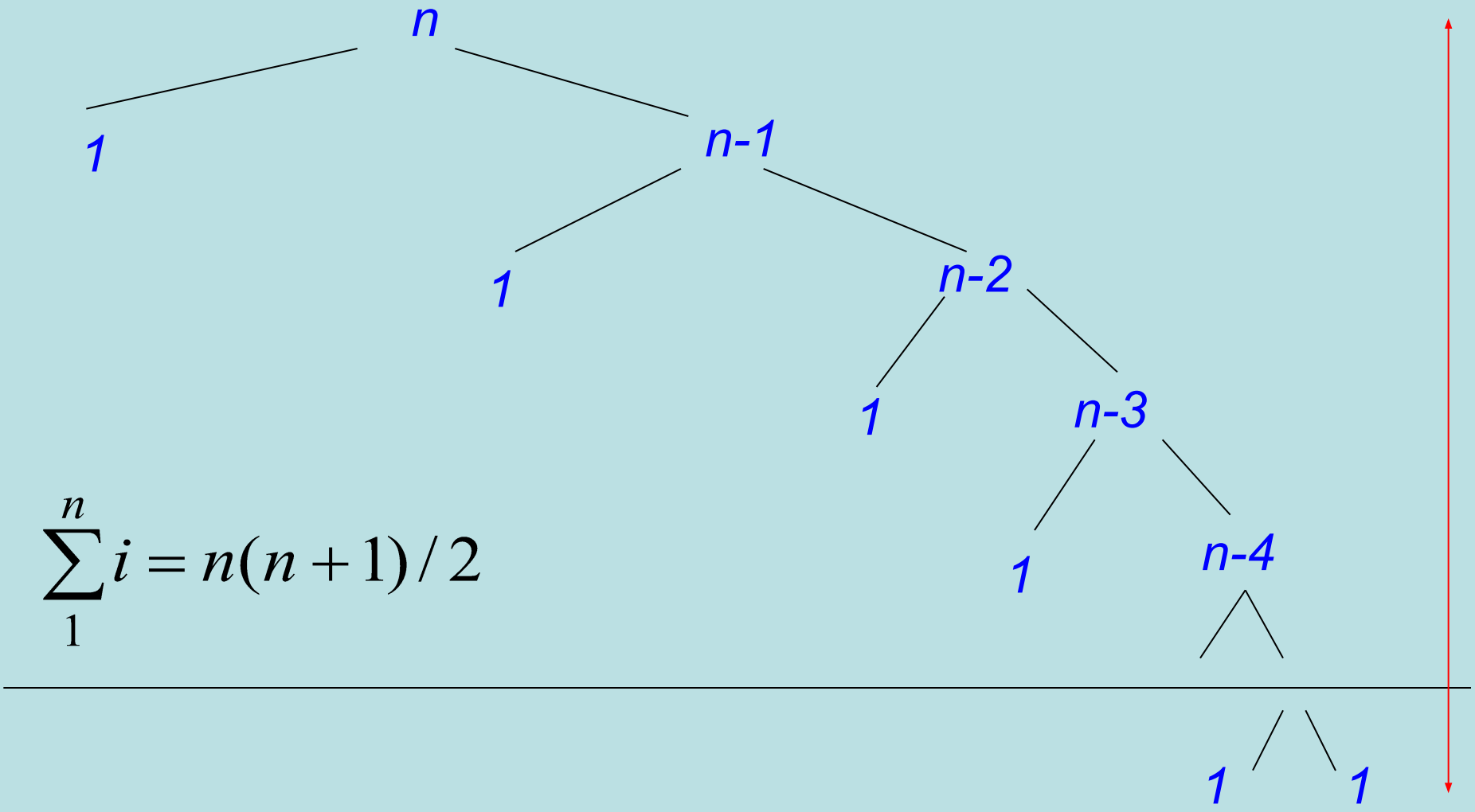
Анализ. Лучший случай



$n \log_2 n$

Быстрая сортировка (**QuickSort**)

Анализ. Худший случай



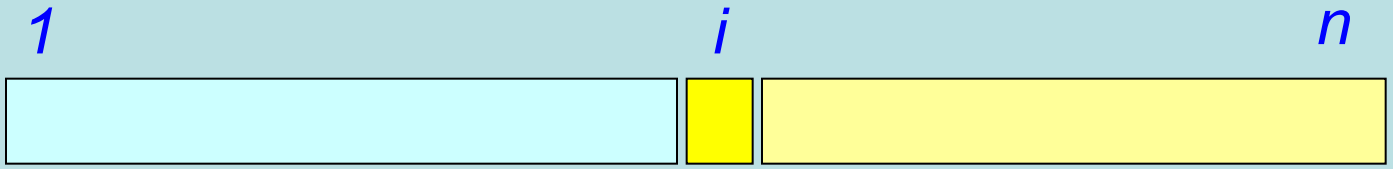
$$\sum_{1}^{n} i = n(n+1)/2$$

Быстрая сортировка (**QuickSort**)

Анализ. Промежуточный случай

Быстрая сортировка (**QuickSort**)

Анализ. В среднем



$q(n)$ – среднее число сравнений

$$q(n) \le Cn + \frac{1}{n}\sum_{i=1}^{n}[q(i-1)+q(n-i)] =$$

$$= Cn + \frac{2}{n}\sum_{i=0}^{n-1} q(i); \quad q(0) = q(1) = b;$$

Докажем, что $q(n) \le k\, n \ln n, \quad k = 2(b + C)$, при $n \ge 2$

По индукции :

а) $n = 2: \quad q(2) \le C \cdot 2 + (q(0) + q(1)) = 2(C + b) = k \le k \cdot 2\ln 2$

б) пусть $q(i) \le k\, i \ln i, \quad i \in 2..(n-1)$; докажем, что $q(n) \le k\, n \ln n$

Быстрая сортировка (**QuickSort**)

Анализ. В среднем (продолжение)

$$q(n) \le Cn + \frac{2}{n}(q(0) + q(1)) + \frac{2}{n}\sum_{i=2}^{n-1} ki \ln i = ?$$

Рассмотрим

$$\sum_{i=2}^{n-1} i \ln i \le \int_2^n x \ln x dx = \frac{1}{2}(x^2 \ln x - \frac{1}{2}x^2)_2^n =$$

$$= \frac{1}{2}(n^2 \ln n - \frac{1}{2}n^2 - 4\ln 2 + \frac{1}{2}4) < \frac{1}{2}(n^2 \ln n - \frac{1}{2}n^2)$$

Быстрая сортировка (**QuickSort**)

Анализ. В среднем (продолжение)

$$? = Cn + \frac{4b}{n} + \frac{k}{n}(n^2 \ln n - \frac{1}{2}n^2) =$$

$$= kn\ln n - \frac{k}{2}n + Cn + \frac{4b}{n} = kn\ln n + \frac{4b}{n} - bn =$$

$$= kn\ln n + b(\frac{4}{n} - n) \le kn\ln n$$

Итак $q(n) \le kn\ln n,$ или $q(n) = O(n\log n)$

Приложение Асимптотические оценки

Обозначения $O(f(n))$, $\Omega(f(n))$, $\Theta(f(n))$

Для указания множества функций,

которые *не более* чем в постоянное число раз превосходят $f(n)$ при достаточно большом n,

используется обозначение $O(f(n))$.

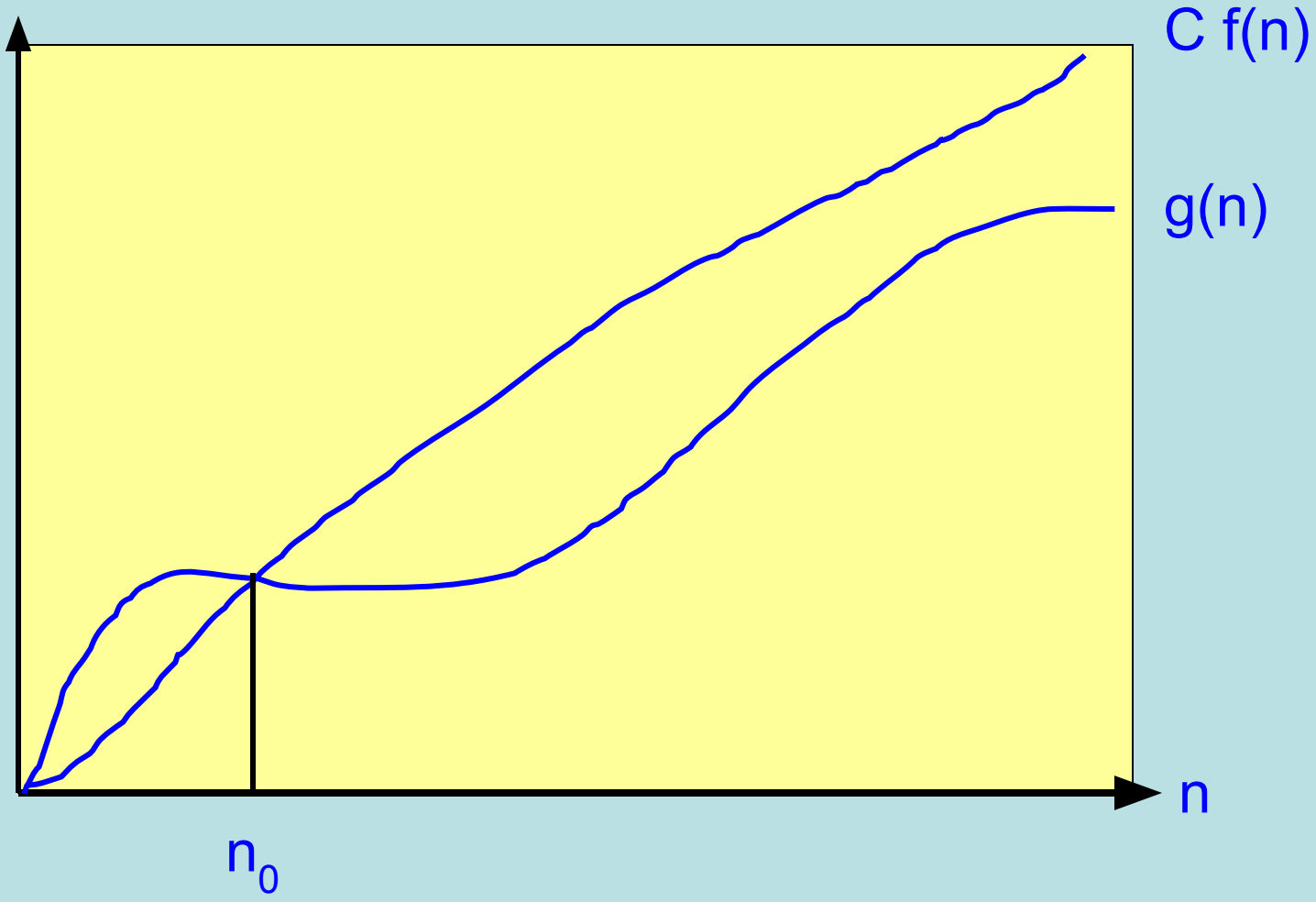
Запись $g(n) \in O(f(n))$ означает, что

существуют:

1) вещественная константа $C > 0$ и
2) натуральная константа n_0 ,

для которых $g(n) \le C\, f(n)$ при всех $n \ge n_0$

Асимптотические оценки
Обозначение $O(f(n))$



Асимптотические оценки

Обозначения $O(f(n))$, $\Omega(f(n))$, $\Theta(f(n))$

Для указания множества функций,

которые *не менее* чем в постоянное число раз превосходят $f(n)$ при достаточно большом n, используется обозначение $\Omega(f(n))$.

Запись $g(n) \in \Omega(f(n))$ означает, что

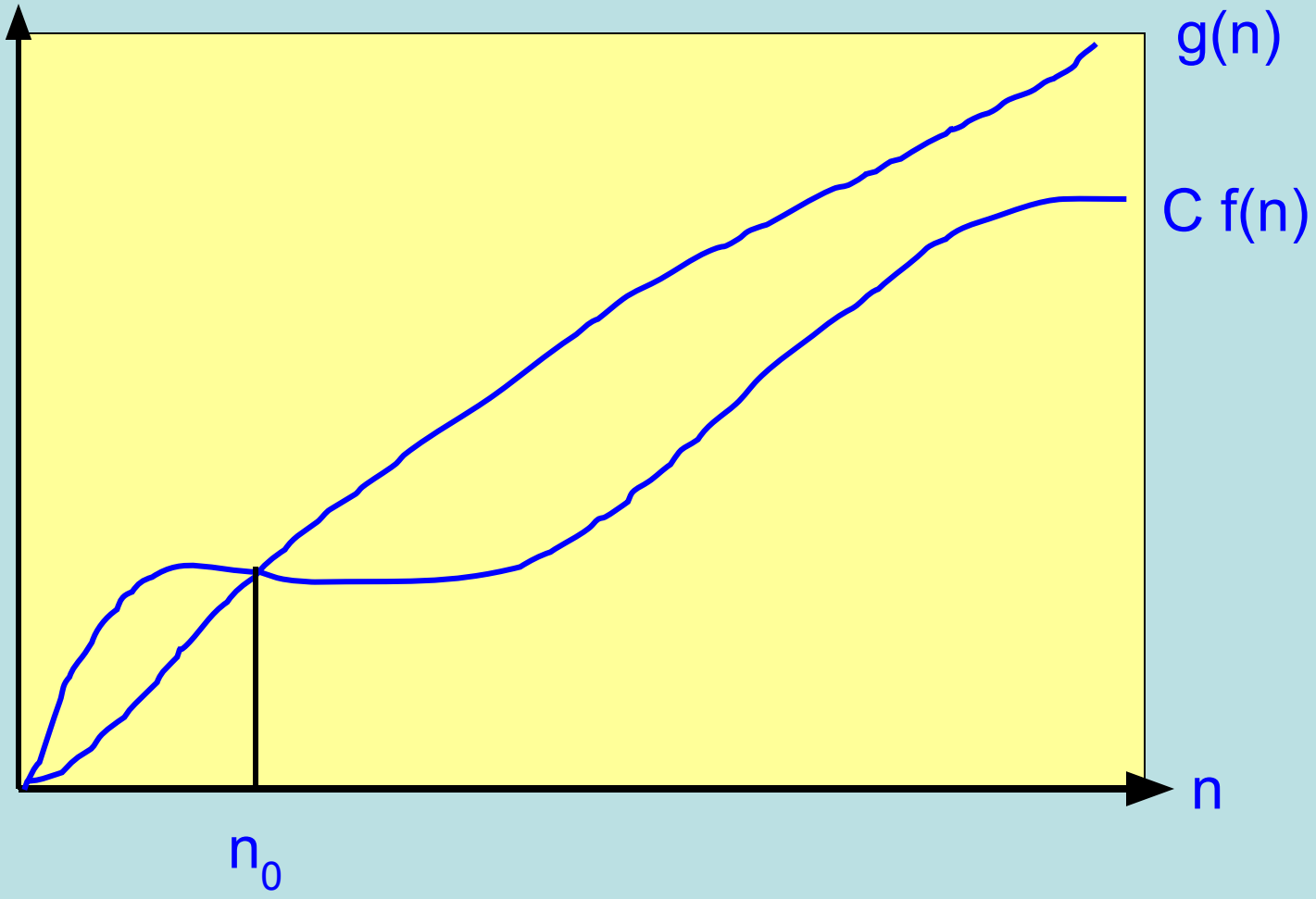
существуют :

1) вещественная константа $C > 0$ и

2) натуральная константа n_0 ,

для которых $g(n) \geq C\, f(n)$ при всех $n \geq n_0$.

Асимптотические оценки
Обозначение $\Omega(f(n))$



Асимптотические оценки

Обозначения $O(f(n))$, $\Omega(f(n))$, $\Theta(f(n))$

Для указания множества функций того же порядка, что и $f(n)$ при достаточно большом n, используется обозначение $\Theta(f(n))$

Запись $g(n) \in \Theta(f(n))$ означает, что существуют :

1) вещественные константы $C_1 > 0$ и $C_2 > 0$ и
2) натуральная константа n_0 , для которых

$C_1 f(n) \le g(n) \le C_2 f(n)$ при всех $n \ge n_0$

Асимптотические оценки
Обозначение $\Theta(f(n))$