```
BinarySearch(A[1..n]: sorted array ↗ of integers; x: integer)
@1   if x ≥ A[n] then
@2      if A[n] = x then return n
@3      else return −1
@4   left ← 1
@5   right ← n
@6   while right − left > 1 do
@7      m ← ⌊(left + right)/2⌋
@8      if A[m] > x then
@9         right ← m
@10     else
@11        left ← m
@12     done
@13  if A[left] = x then return left
@14  return −1
```

Лема 0.1. *Нека $A[1..n]$ е непразен масив от сортирани във възходящ ред цели числа, x е цяло число. Тогава:*

1. *ако $x \geq A[n]$, $BinarySearch(A, x)$ завършва за време $\Theta(1)$, а ако $x < A[n]$, $BinarySearch(A, x)$ завършва за време $\Theta(\log n)$.*

2. *ако $BinarySearch(A, x)$ завърши, то резултатът е ℓ, за който:*

$$\ell = \begin{cases} -1, & \text{ако } x \notin A[1..n] \\ \max\{1 \leq i \leq n \,|\, A[i] = x\}, & \text{иначе.} \end{cases}$$

Доказателство. Ако $x \geq A[n]$, то $BinarySearch(A, x)$, то проверката на ред @1 дава истина, и процедурата завършва след изпълнението на два if-оператора за $\Theta(1)$ време.

Нека $x < A[n]$. Нека ℓ_i и r_i са стойностите на $left$ и $right$ преди i-тото изпълнение на ред @6. Тогава:

$$\ell_1 = 1 \text{ и } r_1 = n.$$

Да означим с $d_i = r_i - l_i$. Нека $m_i = \lfloor(\ell_i + r_i)/2\rfloor$. Тогава в зависимост от сравнението на ред @8 е в сила точно едно от двете:

$$(\ell_{i+1} = \ell_i \text{ и } r_{i+1} = m_i) \quad \text{или}$$
$$(\ell_{i+1} = m_i \text{ и } r_{i+1} = r_i).$$

Да допуснем, че се изпълнява тялото на цикъла. Тогава $\ell_i + 1 < r_i$ и следователно $m_i \geq (\ell_i + \ell_i + 2)/2 > \ell_i$ и $m_i \leq (r_i + r_i - 2)/2 = r_i - 1 < r_i$. Следователно $m_i \in (\ell_i, r_i)$. Сега лесно се убеждаваме, че:

$$m_i - \ell_i \leq \tfrac{\ell_i + r_i + 1}{2} - \ell_i = \tfrac{r_i - \ell_i + 1}{2} = \tfrac{d_i + 1}{2} \text{ и}$$
$$r_i - m_i \leq r_i - \tfrac{\ell_i + r_i}{2} = \tfrac{r_i - \ell_i}{2} = \tfrac{d_i}{2}.$$

Тъй като $d_{i+1} = r_{i+1} - \ell_{i+1} \in \{m_i - \ell_i, r_i - m_i\}$, то получаваме, че:

$$d_{i+1} \leq \frac{d_i + 1}{2}, \text{ откъдето } d_{i+1} - 1 \leq \frac{d_i - 1}{2}.$$

Тъй като $d_1 - 1 = (n-2)$, то по индукция получаваме, че $d_i - 1 \leq (n-2)2^{i-1}$, или $d_i \leq (n-2)2^{-(i-1)} + 1$. Следователно, ако while-цикълът се изпълни при $i = \log_2(2n)$, ще имаме, че $d_i \leq \frac{n-2}{n} + 1 < 2$. От друга страна d_i е цяло, откъдето $d_i \leq 1$. Но това означава, че $r_i - \ell_i \not> 1$, тоест най-късно на тази стъпка тялото на цикъла няма да бъде изпълнено. Следователно while-цикълът се изпълнява най-много $\Theta(\log_2 n + 1)$ пъти.

Сега ще покажем коректността. Нека с $I(i)$ означим твърдението:

$$I(i) : x \in A[1..n] \Rightarrow (x < A[r_i] \& x \geq A[\ell_i]).$$

Ако процедурата завърши преди ред @4, то $x \geq A[n]$ и от ред @2 и @3 се вижда, че $x = A[n]$ и резултатът е $\ell = n$ или $x > A[n]$ и резултатът е $\ell = -1$. Така, може да се съсредоточим на случая, когато се изпълнява ред @4, тоест $x < A[n]$. Ще покажем с индукция по i, че $I(i)$ е вярно.

- $I(1)$. Ако $x \in A[1..n]$, то тъй като $A[1..n]$ е сортиран във възходящ ред, то $x \geq A[1] = A[\ell_1]$. Тъй като ред @4 се изпълнява, то $x < A[n] = A[r_1]$.

- Нека $I(i)$ е истина. Ако $r_{i+1} = m_i$ и $\ell_{i+1} = \ell_i$, то проверката на ред @8 е дала $x < A[m_i]$ и следователно $x < A[r_{i+1}]$. Второто условие следва от $I(i)$. От друга страна, ако $r_{i+1} = r_i$ и $\ell_{i+1} = m_i$, то x $/A[m_i]$ и следователно $x \geq A[m_i]$. Оттук, отново $x \geq A[\ell_{i+1}]$, докато $x < A[r_{i+1}] = A[r_i]$ следва от $I(i)$.

Сега при завършването на while-цикъла знаем, че ако $x \in A[1..n]$, то $A[\ell_i] \leq x < A[r_i]$ и $r_i - \ell_i \leq 1$. Тъй като $A[1..n]$ е сортиран във възходящ ред, то ако $x \in A[1..n]$, $r_i - \ell_i = 1$ и $x = A[\ell_i]$. Тогава резултатът е $\ell = \ell_i$ и от $I(i)$ знаем, че $x < A[r] = A[\ell_i + 1]$, тоест ℓ_i е най-големият индекс, за който $x = A[\ell]$.

Обратно, ако $x = A[\ell]$ на ред @13, то $x \in A[1..n]$ и следователно $A[\ell_i] \leq x < A[r_i]$. Тъй като $r_i - \ell_i \leq 1$, то може да завършим както горе. Следователно, резултатът е $\ell = -1$ точно когато $x \notin A[1..n]$. □

```
Merge(A[1..n]: array of integers; ℓ,m,r: indices in A; B[1..n]: array buffer)
@1   for i = ℓ to r − 1 do
@2       B[i] ← A[i]
@3   i ← ℓ
@4   j ← m
@5   s ← ℓ
@6   while s < r do
@7       low ← j ≥ r or (i < m and B[i] ≤ B[j])
@8       if low then
@9           A[s] ← B[i]
@10          i ← i + 1
@11      else
@12          A[s] ← B[j]
@13          j ← j + 1
@14      s ← s + 1
@15  done
```

Лема 0.2. *При вход $A[1..n]$ от цели числа и индекси $1 \leq \ell \leq m \leq r \leq n+1$, процедурата $Merge(A, \ell, m, r, B)$ завършва за време $\Theta(r - \ell)$. Нещо повече, ако $B[\ell..r-1]$ и $A[\ell..r-1]$ не споделят обща памет и $A[\ell..m-1]$ и $A[m..r-1]$ са сортирани във възходящ ред, то ако A' е резултатният масив A, $A[\ell..r-1]$ е сортиран във възходящ ред.*

Доказателство. Ясно е, че for-цикълът на ред @1 извършва $r - \ell$ операции. Итерациите на while-цикъла също са $r - \ell$, защото всяка итерация увеличава s с точно 1 и s не се променя на друго място освен на ред @14. Тъй като $s = \ell$ преди първата итерация на while-цикъла, то броят на всички итерации е $r - \ell$ като е ясно, че всяка операция на редовете @7–@14 има константна сложност. Следователно процедурата завършва за време $\Theta(r - \ell)$.

Нека $B[\ell..r-1]$ и $A[\ell..r-1]$ не споделят обща памет. Тогава for-цикълът на ред @1 копира елементите на масива $A[\ell..r-1]$ в $B[\ell..r-1]$. Тъй като в рамките всяка итерация на while-цикъла увеличава точно една от променливите i и j с едно, а другата остава не променена, то i и j се променят общо $r - \ell$ пъти. Тъй като $j = m$ преди първата итерация и не се променя след като стане r, защото тогава low е истина, то j се променя най-много $r - m$ пъти. От друга страна, ако j се променя по-малко от $r - m$ пъти, то low е истина само когато $i < m$ и тогава i се променя най-много $m - \ell$ пъти. Това обаче означава, че i и j се променят общо по-малко от $(r - m) + (m - \ell) = r - \ell$ пъти, което е противоречие. Следователно j се променя точно $r - m$ пъти и оттук i се променя точно $m - \ell$ пъти.

Следователно по време на while-цикъла $j \in [m, r]$ и $i \in [\ell, m]$. Това означава, че на редове @9 и @10 се променят единствено стойности от масива $A[\ell..r-1]$ и тъй като тази памет няма общи елементи с $B[\ell..r-1]$, то масивът $B[\ell..r-1]$ не се променя.

Нека i_k, j_k, s_k са стойностите на i, j, s преди k-тата итерация на while-цикъла. Ще докажем, че:

$$\begin{aligned} I(k) : A[\ell..s_k - 1] \quad & \text{е сортирана пермутация на елементите } B[\ell..i_k - 1] \cup B[m..j_k - 1] \text{ и} \\ & ((i_k < m \& j_k > m) \Rightarrow B[i_k] > B[j_k - 1] \& (i_k > \ell \& j_k < r) \Rightarrow B[j_k] \geq B[i_k - 1]). \end{aligned}$$

- При $k = 1$, $i_1 = \ell$, $j_1 = m$ и $s_1 = \ell$, така че $I(1)$ говори за празните множества, предпоставките на втората част са лъжа.

- Нека $I(k)$ е истина. Нека се изпълнява тялото на цикъла. Да допуснем първо, че $i_k < m$ и $j_k < r$. Тогава low е истина точно когато $B[i_k] \leq B[j_k]$. Така, ако low е истина, то $A[s_k] = B[i_k]$ и $i_{k+1} = i_k$, $s_{k+1} = s_k + 1$, $j_{k+1} = j_k$. Тъй като $B[i_k]$ е не по-малък от всеки елемент в $B[\ell..i_k - 1]$, защото $B[\ell..m]$ е сортиран и тъй като от $I(k)$, $B[i_k] > B[j_k - 1]$ и отново $B[m..j_k - 1]$ е сортиран във възходящ ред, то първата част на $I(k+1)$ е ясна. За втората част, тъй като $B[i_{k+1} + 1] \geq B[i_{k+1}] > B[j_k - 1] = B_{j_{k+1} - 1}]$ то първата импликация е ясна. Втората следва от това, че low е истина.

  Ако low е лъжа, то $B[i_k] > B[j_k]$ и случаят е симетричен. Накрая, да допуснем, че $j_k = r$ или $i_k = m$. Тъй като $j_k + i_k < m + r$, то е възможно точно едно от равенствата. Това показва, че low е истина точно когато $j_k = r$ и е лъжа точно когато $i_k = m$. Както и по-горе се убеждаваме, че $I(k+1)$ се запазва.

От това, че $I(r-\ell)$ е истина и $i_{r-\ell}=m$, $j_{r-\ell}=r$, $s_{r-\ell}=r$, следва, че $A[\ell..r-1]$ е сортирана пермутация на $B[\ell..m-1]\cup B[m..r-1]$, което е точно копието на първоначалния подмасив $A[\ell..r-1]$. $\square$

```
MergeSort(A[1..n]: array of integers; ℓ,r: indices in A;B[1..n]: array buffer)
@1   if r − ℓ ≤ 1 return
@2   m ← ⌊(r + ℓ)/2⌋
@3   MergeSort(A, ℓ, m, B)
@4   MergeSort(A, m, r, B)
@5   Merge(A, ℓ, m, r, B)
```

Лема 0.3. *За всеки вход $A[1..n]$, $B[1..n]$ и $1\le \ell\le r\le n+1$ завършва за време $O((r-\ell)\log(r-\ell)$. Нещо повече, ако масивите A и B не споделят обща памет, то резултатният $A[\ell..r-1]$ е сортирана пермутация входния подмасив $A[\ell..r-1]$.*

Доказателство. Доказателството ще направим с индукция по $d=r-\ell$. Ако $d\le 1$, то процедурата завършва поради проверката на ред @1. В противен случай $r-\ell>1$ и $m=\lfloor(\ell+r)/2\rfloor$. Тогава:

$$m-\ell\le(\ell+r)/2-\ell=(r-\ell)/2=d/2$$
$$m-\ell\le r-(\ell+r-1)/2-\ell=(r-\ell+1)/2=(d+1)/2.$$

Тъй като $d>1$, то $(d+1)/2<(d+d)/2=d$. Следователно може да приложим индуктивната хипотеза за $d'=m-\ell$ и $d''=r-m$, откъдето @3 и @4 завършват. Както видяхме ред @5 също завършва.

Сега да видим, че процедурата е коректна. Наистина, ако $r-\ell\le 1$, то $A[\ell..r-1]$ е съставен от един или от нула елемента. И в двата случая е сортиран. Ако $d>1$, то както видяхме, може да приложим индуктивното предположение за $MergeSort(A,\ell,m,B)$ и $MergeSort(A,m,r,B)$. Тогава A,B и ℓ,m,r удовлетворяват условията на предишната лема и следователно $A[\ell..r-1]$ е сортирана пермутация на входния $A[\ell..r-1]$.

Накрая, за времевата сложност. Нека $T(d)$ е времевата сложност в лошия случай на MergeSort, когато индексите ℓ и r удовлетворяват $r-\ell=d$. Нека $T'(d)=\max_{d'\le d}T(d')$. Тогава $T'(d)$ е растяща функция. Тогава от предишната лема и анализа по-горе имаме, че:

$$T'(d)\le T'(\lfloor d/2\rfloor)+T'(\lfloor(d+1)/2\rfloor)+c.d.$$

ако $d=2^k$, получаваме, че:

$$T'(2^k)\le T'(2^{k-1})+T'(2^{k-1})+c.2^k=2T'(2^{k-1})+c.2^k.$$

Следователно:

$$2^{k-i}T'(2^i)\le 2^{k-i+1}(T'(2^{i-1})+c.2^k.$$

Събирайки тези неравенства за $i=1\dots k$, получаваме, че:

$$T'(2^k)\le 2^{k+1}(T'(2^0)+c.k2^k=a2^{k+1}+ck.2^k.$$

Тъй като $d=2^k$, то $T'(d)\le 2ad+cd\log d$. Накрая, ако $d\in(2^{k-1},2^k]$, то $2^k<2d$ и $k<\log d+1$. $\square$

Теорема 0.1. *Нека $\mathcal{S}$ е детерминиран алгоритъм, който при вход масив от цели числа $A[1..n]$ намира пермутация π на числата $\{1,2,\dots,n\}$, за която:*

$$A[\pi(1)]\le A[\pi(2)]\le\cdots\le A[\pi(n)].$$

Ако единствените операции, в които участва масивът A в рамките на алгоритъма $\mathcal{S}$ са сравнения от вида $if A[i]<A[j]then$, където i и j може да са константи или променливи, то има $c>0$, за което за всяко $n\ge 9$ има масив $A[1..n]$, за който $\mathcal{S}$ извършва поне $cn\log_2 n$ сравнения.

Доказателство. Да допуснем противното и нека за всяко $c>0$, има $n\in\mathbb{N}$, за което при всеки вход $A[1..n]$ от цели числа, $\mathcal{S}$ коректно сортира A и прави по-малко $cn\log_2 n$ сравнения.

Да разгледаме поведението на алгоритъма $\mathcal{S}$ върху всевъзможните пермутации π на числата $\{1,2,\dots,n\}$. За всяка такава пермутация, очевидно, $\mathcal{S}$ трябва да върне π^{-1}. Да разгледаме изчисленията (редиците от операции), които извършва $\mathcal{S}$ върху пермутациите π. На всяко изчисление $\mathcal{S}$ върху пермутация π съпоставяме редицата от $\alpha(\pi)\in\{0,1\}^*$, като $|\alpha(\pi)|$ е броят сравнения, извършени от $\mathcal{S}$ върху π и за $i\le|\alpha(\pi)|$:

$$\alpha_i(\pi)=0\iff \mathcal{S}\text{ е получил отговор истина при } i\text{-тото сравнение при вход }\pi.$$

Да допуснем, че $\alpha(\pi_1)=\alpha(\pi_2)$ за някои пермутации π_1 и π_2 на $\{1,2,\dots,n\}$. Тогава, тъй като изпълнението на $\mathcal{S}$ не зависи от π_1 и π_2, освен при сравненията, то изпълненията на $\mathcal{S}$ върху π_1 и π_2 ще бъдат идентични. Следователно резултатът, който $\mathcal{S}$ ще върне при вход π_1 и π_2 ще бъде един и съща, тоест $\pi_1^{-1}=\pi_2^{-1}$. Но това означава, че $\pi_1=\pi_2$.

С това показахме, че α е инекция с $dom(\alpha) = \{\pi \,|\, \pi$ пермутация на $\{1, 2, \ldots, n\}\}$. Тъй като $|\alpha(\pi)| \leq cn \log n$, то $range(\alpha) \subseteq \{0,1\}^{[cn \log n]}$. Следователно:

$$n! = |\{\pi \,|\, \pi \text{ пермутация на } \{1, 2, \ldots, n\}\}| = |dom(\alpha)| \leq |range(\alpha)| \leq 2^{cn \log_2 n}.$$

Логаритмуваме от двете страни и получаваме:

$$\log_2 n! \leq cn \log_2 n.$$

От друга страна:

$$\log_2 n! = \sum_{k=1}^{n} \log_2 k = \log_2 e \sum_{k=2}^{n} \ln k.$$

Остана да забележим, че:

$$\sum_{k=2}^{n} \ln k = \sum_{k=1}^{n-1} \int_{k}^{k+1} \ln(k+1) dx \geq \sum_{k=1}^{n-1} \int_{k}^{k+1} \ln x dx = \int_{1}^{n} \ln x dx,$$

защото $\ln x$ е растяща функция. След интегриране по части получаваме, че:

$$\int_{1}^{n} \ln x dx = x \ln x \Big|_{1}^{n} - \int_{1}^{n} x d \ln x = n \ln n - n + 1.$$

Следователно:

$$\log_2 e(n \ln n - n + 1) \leq cn \log_2 n, \text{ тоест } n \log_2 n - n \log_2 e + 1 \leq cn \log_2 n.$$

Последното обаче, не е вярно за $c = \frac{1}{2}$ и $n \geq 9 > e^2$. $\square$

Лема 0.4. *Има алгоритъм с времева сложност $O(n \log n)$, който по даден масив $A[1..n]$ от числа намира броя на инверсиите в A, тоест намира $\sigma(A)$, където:*

$$\sigma(A) = \{(i,j) \,|\, i < j \,\&\, A[i] > A[j]\}.$$

Доказателство. Нека $m = \lfloor (n+1)/2 \rfloor$ и да означим с $A' = A[1..m]$ и $A'' = A[m+1..n]$. Тогава е ясно, че:

$$\sigma(A) = \sigma(A') \cup \sigma(A'') \cup \{(i,j) \,|\, i \leq m < j \,\&\, A[i] > A[j]\}.$$

Нека кръстим $M = \{(i,j) \,|\, i \leq m < j \,\&\, A[i] > A[j]\}$ и B' and B'' са сортираните пермутации на A' и A'' съответно. Тогава

$$|M| = \{(i,j) \,|\, B'[i] > B''[j]\}.$$

Ако за j, индекс от B'', означим $s(j) = \{i \,|\, B'[i] > B''[j]\}$, то е ясно, че:

$$\{(i,j) \,|\, B'[i] > B''[j]\} \quad = \quad \bigcup_{j} s(j) \text{ като } s(j) \cap s(j') = \emptyset \text{ за } j \neq j'.$$

Накрая, да забележим, че при сливането на B' и B'', когато се добавя j от B'' имаме, че $B''[j] < B'[i]$ за всички неразгледани до този момент индекси i от B'. Следователно:

$$s(j) = |B'| - i_j,$$

където i_j е стойността на индекса i от B' в момента, в който се добавя j. Това показва, че стойностите $s(j)$ са известни в момента на добавянето на елемента j в Merge и следователно, $|M| = \sum_{j=1}^{|B''|} |s(j)|$ може да бъде намерено по време на сливането, без асимтотично да увеличава времевата сложност на MergeSort. От друга страна $|\sigma(A')|$ и $|\sigma(A'')|$ могат да бъдат намерени при рекурсивните извиквания на $MergeSort$.

$\square$