Duomenų struktūros ir algoritmai

5 paskaita

2022-03-16

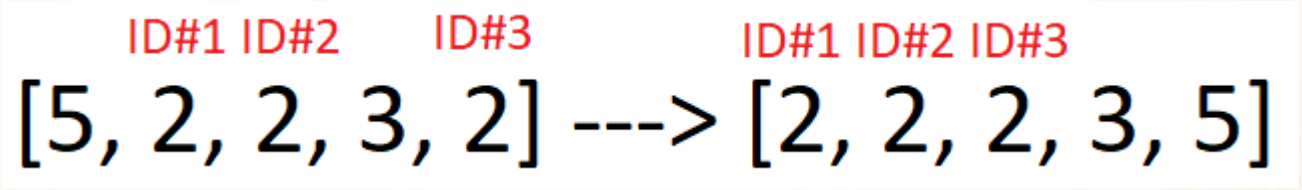
5 paskaitos tikslas

- Išanalizuoti rikiavimo algoritmų pseudokodus.
- Įvertinti rikiavimo algoritmų sudėtingumus.
- Suprasti rikiavimo algoritmų eigos vizualizacijas.

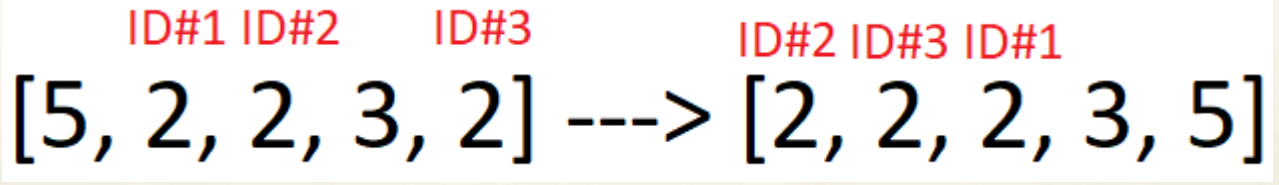
Rikiavimo algoritmo stabilumas

Stabiliu vadinamas toks rikiavimo algoritmas, kuriuo išrikiuojant sąrašą išlaikomas vienodų elementų tarpusavio eiliškumas.

Stabilaus rikiavimo pavyzdys:



Nestabilaus rikiavimo pavyzdys:



Rikiavimo algoritmai ir jų sudėtingumai gali būti įvertinti

1. Pagal sąrašo elementų palyginimų skaičių.
2. Pagal sąrašo elementų sukeitimų skaičių.
3. Pagal vidinės atminties naudojimą.
4. Pagal išorinės atminties naudojimą (*external sorting*).
5. Pagal iškviestų rekursijų skaičių.
6. Pagal stabilumą.
7. Pagal pritaikomumą (pavyzdžiui, *bubble sort* ar *quick sort* algoritmuose galima pridėti papildomas sąlygas, kurios pagerina sudėtingumą).
8. Pagal nuskaitymų ir įrašymų skaičių tekstiniame faile (pavyzdžiui, *external sorting*).
9. Pagal kitas galimas modifikacijas.

Išrinkimo algoritmas (angl. Selection sort)

SELECTION-SORT(A)
1. n ← length(A)
2. **for** j ← 0 **to** n – 2 **do**
3. smallest ← j
4. **for** i ← j + 1 **to** n – 1 **do**
5. **if** A[i] < A[smallest] **then**
6. smallest ← i
7. swap(A[j] , A[smallest])

Algoritmo sudėtingumas (pagal palyginimų skaičių):

$$\sum_{i=1}^{n-1}(n-i) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \sim O(n^2),$$

čia n = length(A).

8	5	2	6	9	3	1	4	0	7

GeeksforGeeks vizualizacija:
https://www.youtube.com/watch?v=xWBP4lzkoyM

Išrinkimo algoritmo sudėtingumas

Blogiausiu atveju: $O(n^2)$,
Geriausiu atveju: $O(n^2)$,
Vidutiniu atveju: $O(n^2)$.

Vidinės atminties naudojimas blogiausiu atveju: O(1).

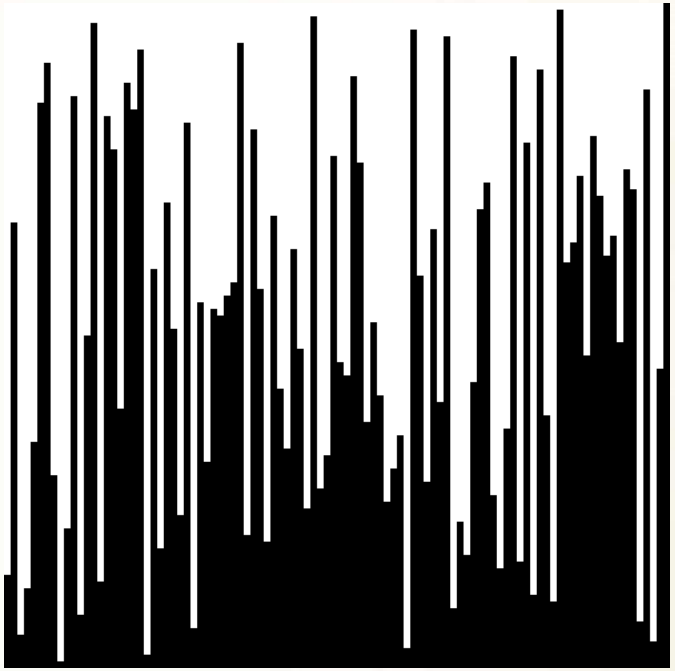
Algoritmo vykdymo metu vidutiniškai atliekama maždaug $\frac{n^2}{2}$ palyginimų ir n sukeitimų (geriausiu, blogiausiu ir vidutiniu atveju).

Ar algoritmas stabilus? – NE

Galima užduotis per egzaminą: „pateikite pavyzdį, įrodantį, kad išrinkimo algoritmas nėra stabilus“.

Burbuliuko algoritmas (angl. Bubble sort)

```
BUBBLE-SORT(A):
1.    n ← length(A)
2.    for i ← 0 to n – 1 do
3.        swapped ← False
4.        for j ← 0 to n – i – 1 do
5.            if A[ j ] > A[ j + 1 ] then
6.                swap(A[ j ], A[ j + 1 ])
7.                swapped ← True
8.        if swapped = False then
9.            break
```



GeeksforGeeks vizualizacija:
https://www.youtube.com/watch?v=nmhjrI-aW5o

Burbuliuko algoritmo sudėtingumas

Blogiausiu atveju: $O(n^2)$,
Geriausiu atveju: $O(n^2)$ ir $O(n)$ – sudėtingumas pagerinamas, kai algoritmas stabdomas paskutinės iteracijos metu, jei nebuvo elementų sukeitimų (*swap*).
Vidutiniu atveju: $O(n^2)$.

Vidinės atminties naudojimas blogiausiu atveju: $O(1)$.

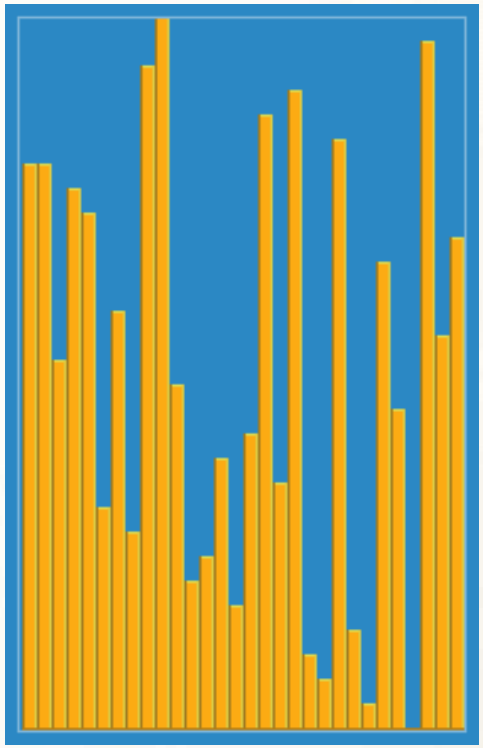
Algoritmo vykdymo metu atliekama maždaug $\frac{n^2}{2}$ palyginimų ir $\frac{n^2}{2}$ sukeitimų (blogiausiu ir vidutiniu atveju).

Ar algoritmas stabilus? – TAIP

Įterpimo algoritmas (angl. Insertion sort)

INSERTION-SORT(A):

```
1.   n ← length(A)
2.   for i ← 1 to n – 1 do
3.       key ← A[ i ]
4.       j ← i – 1
5.       while j ≥ 0 and key < A[ j ] do
6.           A[ j+1 ] ← A[ j ]
7.           j ← j – 1
8.       A[ j+1 ] ← key
```



GeeksforGeeks vizualizacija:
https://www.youtube.com/watch?v=OGzPmgsI-pQ

Įterpimo algoritmo sudėtingumas

Blogiausiu atveju: $O(n^2)$,
Geriausiu atveju: $O(n)$,
Vidutiniu atveju: $O(n^2)$.

Vidinės atminties naudojimas blogiausiu atveju: $O(1)$.

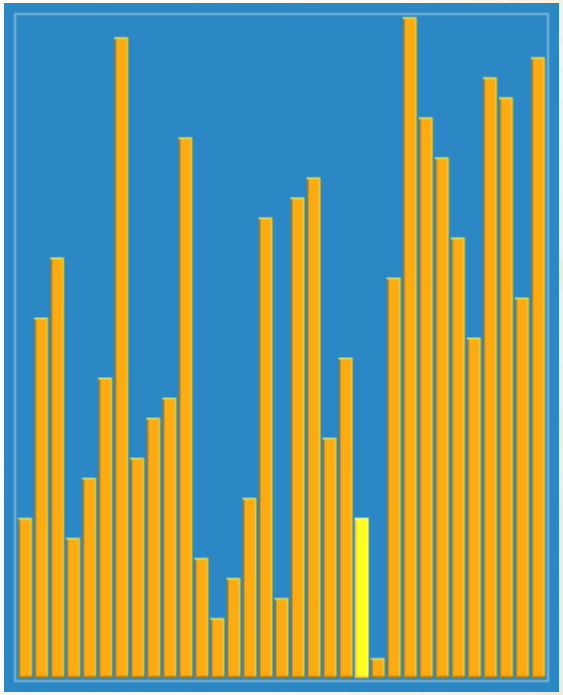
Algoritmo vykdymo metu vidutiniškai atliekama maždaug $\frac{n^2}{4}$ palyginimų ir $\frac{n^2}{8}$ sukeitimų, blogiausiu atveju – maždaug $\frac{n^2}{2}$ palyginimų ir $\frac{n^2}{4}$ sukeitimų.

Ar algoritmas stabilus? – TAIP

Rikiavimas Šelo metodu (angl. Shell sort)

SHELL-SORT(A):

```
1.   n ← len(A)
2.   gap ← [ n / 2 ] # sveikoji dalis
3.   while gap > 0 do
4.       for i ← gap to n – 1 do
5.           temp ← A[ i ]
6.           j ← i
7.           while  j ≥ gap and A[ j – gap ] > temp do
8.               A[ j ] ← A[ j – gap ]
9.               j ← j – gap
10.          A[ j ] ← temp
11.      gap ← [ gap / 2 ]
```



Galimi tarpo parinkimai:

$gap \leftarrow [\, n / 2^k \,]$ (Shell, 1959),

$gap \leftarrow 2 * [\, n / 2^{k+1} \,] + 1$ (Frank & Lazarus, 1960).

GeeksforGeeks vizualizacija:

https://www.youtube.com/watch?v=SHcPqUe2GZM

Dar daugiau galimų tarpų...

General term ($k \geq 1$)	Concrete gaps	Worst-case time complexity	Author and year of publication
$\left\lfloor \frac{N}{2^k} \right\rfloor$	$\left\lfloor \frac{N}{2} \right\rfloor, \left\lfloor \frac{N}{4} \right\rfloor, \ldots, 1$	$\Theta\left(N^2\right)$ [e.g. when $N = 2^p$]	Shell, 1959[4]
$2\left\lfloor \frac{N}{2^{k+1}} \right\rfloor + 1$	$2\left\lfloor \frac{N}{4} \right\rfloor + 1, \ldots, 3, 1$	$\Theta\left(N^{\frac{3}{2}}\right)$	Frank & Lazarus, 1960[8]
$2^k - 1$	$1, 3, 7, 15, 31, 63, \ldots$	$\Theta\left(N^{\frac{3}{2}}\right)$	Hibbard, 1963[9]
$2^k + 1$, prefixed with 1	$1, 3, 5, 9, 17, 33, 65, \ldots$	$\Theta\left(N^{\frac{3}{2}}\right)$	Papernov & Stasevich, 1965[10]
Successive numbers of the form $2^p 3^q$ (3-smooth numbers)	$1, 2, 3, 4, 6, 8, 9, 12, \ldots$	$\Theta\left(N \log^2 N\right)$	Pratt, 1971[1]
$\frac{3^k - 1}{2}$, not greater than $\left\lceil \frac{N}{3} \right\rceil$	$1, 4, 13, 40, 121, \ldots$	$\Theta\left(N^{\frac{3}{2}}\right)$	Pratt, 1971[1]
$\prod_I a_q$, where $a_q = \min\left\{n \in \mathbb{N} : n \geq \left(\frac{5}{2}\right)^{q+1}, \forall p : 0 \leq p < q \Rightarrow \gcd(a_p, n) = 1\right\}$ $I = \left\{0 \leq q < r \mid q \neq \frac{1}{2}\left(r^2 + r\right) - k\right\}$ $r = \left\lfloor \sqrt{2k + \sqrt{2k}} \right\rfloor$	$1, 3, 7, 21, 48, 112, \ldots$	$O\left(N^{1+\sqrt{\frac{8\ln(5/2)}{\ln(N)}}}\right)$	Incerpi & Sedgewick, 1985,[11] Knuth[3]
$4^k + 3 \cdot 2^{k-1} + 1$, prefixed with 1	$1, 8, 23, 77, 281, \ldots$	$O\left(N^{\frac{4}{3}}\right)$	Sedgewick, 1986[6]
$\begin{cases} 9\left(4^{k-1} - 2^{\frac{k}{2}}\right) + 1 & k \text{ even}, \\ 4^{k+1} - 6 \cdot 2^{(k+1)/2} + 1 & k \text{ odd} \end{cases}$	$1, 5, 19, 41, 109, \ldots$	$O\left(N^{\frac{4}{3}}\right)$	Sedgewick, 1986[12]
$h_k = \max\left\{\left\lfloor \frac{5h_{k-1}}{11} \right\rfloor, 1\right\}, h_0 = N$	$\left\lfloor \frac{5N}{11} \right\rfloor, \left\lfloor \frac{5}{11}\left\lfloor \frac{5N}{11} \right\rfloor \right\rfloor, \ldots, 1$	Unknown	Gonnet & Baeza-Yates, 1991[13]
$\left\lceil \frac{1}{5}\left(9 \cdot \left(\frac{9}{4}\right)^{k-1} - 4\right) \right\rceil$	$1, 4, 9, 20, 46, 103, \ldots$	Unknown	Tokuda, 1992[14]
Unknown (experimentally derived)	$1, 4, 10, 23, 57, 132, 301, 701$	Unknown	Ciura, 2001[15]

Šaltinis: https://en.wikipedia.org/wiki/Shellsort

Šelo rikiavimo algoritmo sudėtingumas

Blogiausiu atveju: nuo $O(n \log^2 n)$ iki $O(n^2)$.
Geriausiu atveju: $O(n \log n)$,
Vidutiniu atveju: priklauso nuo tarpų parinkimo (vidutiniškai $\sim O(n \log n)$ arba $\sim O(n^{1,2})$).

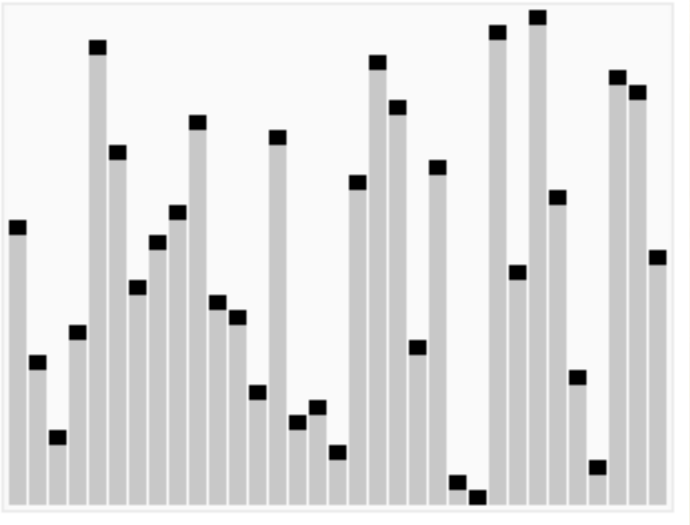
Vidinės atminties naudojimas blogiausiu atveju: $O(n)$.

Ar algoritmas stabilus? – NE

Spartaus rikiavimo algoritmas (angl. Quick sort)

PARTITION(A, low, high):

```
1.  i ← (low – 1)
2.  pivot ← A[high]
3.  for j ← low to high – 1 do
4.      if A[ j ] ≤ pivot then
5.          i ← i + 1
6.          swap(A[ i ], A[ j ])
7.  swap(A[ i + 1 ], A[high])
8.  return (i + 1)
```



QUICK-SORT(A, low, high): # iš pradžių low = 0, high = length(A) – 1

```
1.  if low < high then
2.      pi ← PARTITION(A, low, high)
3.      QUICK-SORT(A, low, pi – 1)
4.      QUICK-SORT(A, pi + 1, high)
```

GeeksforGeeks vizualizacija:
https://www.youtube.com/watch?v=PgBzjlCcFvc

Spartaus rikiavimo algoritmo sudėtingumas

Blogiausiu atveju: $O(n^2)$,
Geriausiu atveju: $O(n \log n)$,
Vidutiniu atveju: $O(n \log n)$.

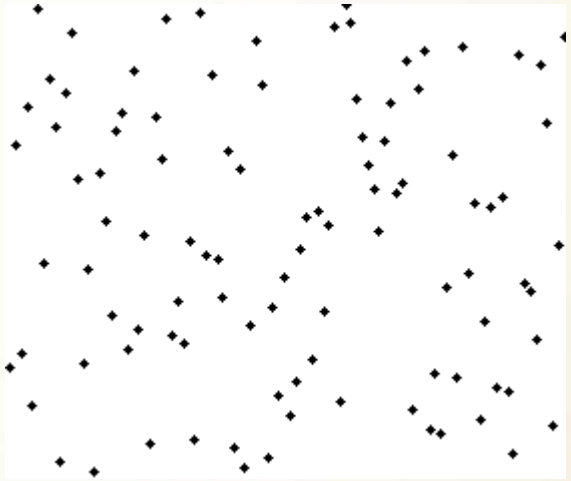
Vidinės atminties naudojimas blogiausiu atveju: $O(1)$.

Ar algoritmas stabilus? – TAIP ir NE (priklauso nuo realizacijos, t. y. pivot kintamojo elgsenos)

Sąlajos rikiavimas (angl. Merge sort)

MERGE(A, l, m, r): # sujungia posąrašius A[l .. m] ir A[m+1 .. r] į išrikiuotą sąrašą A[l .. r]

```
1.  n1 ← m – l + 1
2.  n2 ← r – m
3.  L ← [0 .. n1 – 1] # naujas masyvas
4.  R ← [0 .. n2 – 1] # naujas masyvas
5.  for i ← 0 to n1 – 1 do
6.      L[ i ] ← A[ l + i ]
7.  for j ← 0 to n2 – 1 do
8.      R[ j ] ← A[ m + 1 + j ]
9.  i ← 0, j ← 0, k ← l
10. while i < n1 and j < n2 do
11.     if L[ i ] ≤ R[ j ] then
12.         A[ k ] ← L[ i ]
13.         i ← i + 1
14.     else
15.         A[ k ] ← R[ j ]
16.         j ← j + 1
17.     k ← k + 1
18. while i < n1 do
19.     A[ k ] ← L[ i ]
20.     i ← i + 1
21.     k ← k + 1
22. while j < n2 do
23.     A[ k ] ← R[ j ]
24.     j ← j + 1
25.     k ← k + 1
```



MERGE-SORT(A, l, r):

```
1.  if l < r do
2.      m ← [ (l + (r – 1)) / 2 ] # sveikoji dalis
3.      MERGE-SORT(A, l, m)
4.      MERGE-SORT(A, m + 1, r)
5.      MERGE(A, l, m, r)
```

GeeksforGeeks vizualizacija:
www.youtube.com/watch?v=JSceec-wEyw

Sąlajos rikiavimo algoritmo sudėtingumas

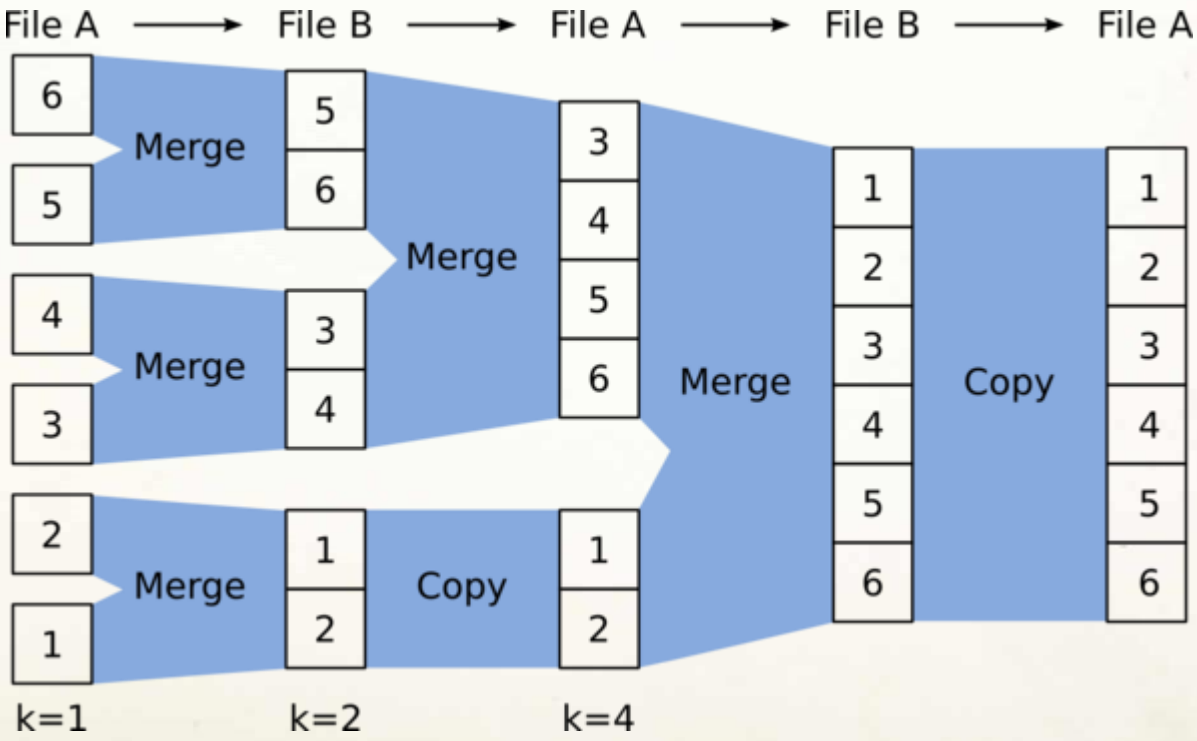
Blogiausiu atveju: O(n log n),
Geriausiu atveju: O(n log n),
Vidutiniu atveju: O(n log n).

Vidinės atminties naudojimas blogiausiu atveju: O(n).

Ar algoritmas stabilus? – TAIP

Išorinis rikiavimas (angl. External sorting) (arba išorinis rikiavimas sujungimu)

Rikiavimo pavyzdys:



Išorinio rikiavimo algoritmo sudėtingumas

Blogiausiu atveju: ~O(n log n),
Geriausiu atveju: ~O(n log n),
Vidutiniu atveju: ~O(n log n).

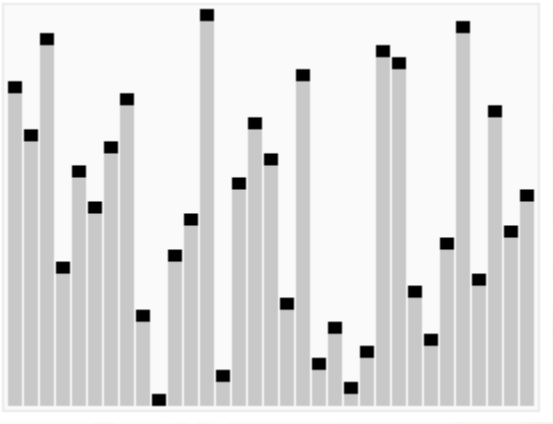
Išorinio rikiavimo principas analogiškas sąlajos rikiavimui, tačiau algoritmo vykdymo laiką papildomai sąlygoja failų nuskaitymo ir įrašymo greitis.

Vidinės atminties naudojimas blogiausiu atveju: O(n).

Ar algoritmas stabilus? – TAIP

Piramidės rikiavimas (angl. Heap sort)

```
HEAPIFY(A, n, i):
1.   largest ← i
2.   l ← 2 * i + 1 # vaikas kairėje
3.   r ← 2 * i + 2 # vaikas dešinėje
4.   if l < n and A[ i ] < A[ l ] then
5.       largest ← l
6.   if r < n and A[largest] < A[ r ] then
7.       largest ← r
8.   if largest ≠ i then
9.       swap(A[ i ], A[largest])
10.      HEAPIFY(A, n, largest)
```



```
HEAP-SORT(A):
    n ← length(A)
    for i ← n downto 0 do
        HEAPIFY(A, n, i)
    for i ← n – 1 downto 1 do
        swap(A[ i ], A[ 0 ])
        HEAPIFY(A, i, 0)
```

GeeksforGeeks vizualizacija:
www.youtube.com/watch?v=MtQL_ll5KhQ

Piramidės rikiavimo algoritmo sudėtingumas

Blogiausiu atveju: O(n log n),
Geriausiu atveju: O(n log n),
Vidutiniu atveju: O(n log n).

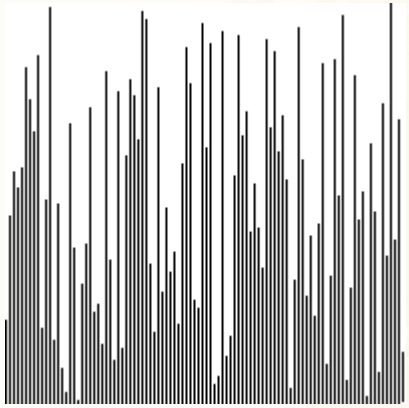
Vidinės atminties naudojimas blogiausiu atveju: O(1).

Ar algoritmas stabilus? – NE

Skaitmeninis rikiavimas (angl. Radix sort)

COUNTING-SORT(A, exp1):

```
1.  n ← length(A)
2.  output ← [0 .. n – 1] # naujas masyvas
3.  count ← [0 .. 9] # naujas masyvas
4.  for i ← 0 to n – 1 do
5.      index ← A[ i ] / exp1
6.      count[ index % 10 ] ← count[ index % 10 ] + 1
7.  for i ← 1 to 9 do
8.      count[ i ] ← count[ i ] + count[ i – 1 ]
9.  i ← n – 1
10. while i ≥ 0 do
11.     index ← A[ i ] / exp1
12.     output[ count[ index % 10] – 1] ← A[ i ]
14.     count[ (index) % 10 ] ← count[ (index) % 10 ] – 1
15.     i ← i – 1
16. i ← 0
17. for i ← 0 to n – 1 do
18.     A[ i ] ← output[ i ]
```



RADIX-SORT(A):

```
max1 ← max(A)
exp ← 1
while max1 / exp > 0 do
    countingSort(A, exp)
    exp ← exp * 10
```

GeeksforGeeks vizualizacija:
https://www.youtube.com/watch?v=nu4gDuFabIM

Skaitmeninio rikiavimo algoritmo sudėtingumas

Blogiausiu atveju: O(n k),
Geriausiu atveju: O(n k),
Vidutiniu atveju: O(n k),

Čia k – maksimalus simbolių skaičius vienam elementui,
n – sąrašo ilgis.

Vidinės atminties naudojimas blogiausiu atveju: O(n k).

Ar algoritmas stabilus? – TAIP

Rikiavimo algoritmų vizualizacijos

Rikiavimo pavyzdžiai ir pseudokodo analizė:
https://www.youtube.com/watch?v=nmhjrI-aW5o&list=PLGvfHSgImk4bKYlXpzPVEN-ictZhrGK04

Didesnės apimties sąrašų rikiavimo vizualizacija:
www.youtube.com/watch?v=kPRA0W1kECg

Šaltiniai:
https://www.geeksforgeeks.org/sorting-algorithms/
http://panthema.net/2013/sound-of-sorting/

Ačiū už dėmesį.

Klausimai?