Specialeji rūšiavimo algoritmai

Raimondas Čiegis

Matematinio modeliavimo katedra, e-paštas: rc@vgtu.lt

Lapkričio 15 d., 2022

Šioje paskaitoje susipažinsime su specialiaisiais rūšiavimo algoritmais, kurie yra spartesni už bendruosius algoritmus, kadangi naudojamės papildoma informacija apie aibės elementus arba uždavinys tenkina papildomus apribojimus.

Šioje paskaitoje susipažinsime su specialiaisiais rūšiavimo algoritmais, kurie yra spartesni už bendruosius algoritmus, kadangi naudojamės papildoma informacija apie aibės elementus arba uždavinys tenkina papildomus apribojimus.

Pradžioje nagrinėsime dar vieną universalų greitąjį rūšiavimo algoritmą, kurio realizacija yra labai paprasta – visą darbą atlieka tinkamai parinkta duomenų struktūra.

Piramidės rūšiavimo algoritmas

Prisiminsime dar vieną dvejetainio medžio variantą – piramidę arba krūvą (angl. *heap*).

Piramidės rūšiavimo algoritmas

Prisiminsime dar vieną dvejetainio medžio variantą – piramidę arba krūvą (angl. *heap*).

Jai būdingos šios dvi pagrindinės savybės:

1. Piramidė yra pilnas, subalansuotas medis: pirmiausia užpildoma medžio šaknis, paskui pirmojo lygmens viršūnės, antrojo lygmens viršūnės ir t. t. Kiekvieną lygmenį užpildome iš kairės į dešinę.

Piramidės rūšiavimo algoritmas

Prisiminsime dar vieną dvejetainio medžio variantą – piramidę arba krūvą (angl. *heap*).

Jai būdingos šios dvi pagrindinės savybės:

1. Piramidė yra pilnas, subalansuotas medis: pirmiausia užpildoma medžio šaknis, paskui pirmojo lygmens viršūnės, antrojo lygmens viršūnės ir t. t. Kiekvieną lygmenį užpildome iš kairės į dešinę.
2. Medis yra sutvarkytas taip, kad kiekvienos viršūnės vaikai yra nedidesni už pačią viršūnę.

Piramidės rūšiavimo algoritmas

Prisiminsime dar vieną dvejetainio medžio variantą – piramidę arba krūvą (angl. *heap*).

Jai būdingos šios dvi pagrindinės savybės:

1. Piramidė yra pilnas, subalansuotas medis: pirmiausia užpildoma medžio šaknis, paskui pirmojo lygmens viršūnės, antrojo lygmens viršūnės ir t. t. Kiekvieną lygmenį užpildome iš kairės į dešinę.
2. Medis yra sutvarkytas taip, kad kiekvienos viršūnės vaikai yra nedidesni už pačią viršūnę.

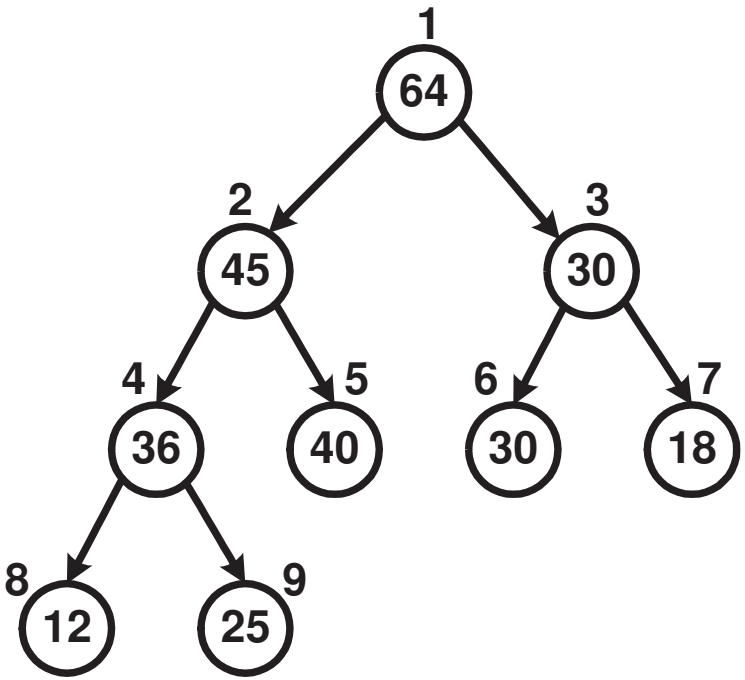
Piramidės rūšiavimo algoritmas

Prisiminsime dar vieną dvejetainio medžio variantą – piramidę arba krūvą (angl. *heap*).

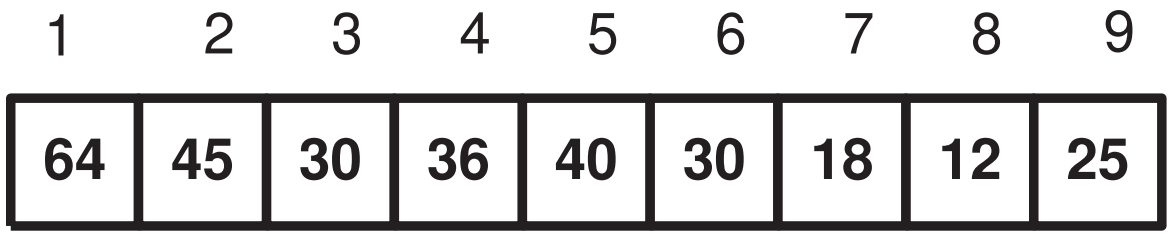
Jai būdingos šios dvi pagrindinės savybės:

1. Piramidė yra pilnas, subalansuotas medis: pirmiausia užpildoma medžio šaknis, paskui pirmojo lygmens viršūnės, antrojo lygmens viršūnės ir t. t. Kiekvieną lygmenį užpildome iš kairės į dešinę.
2. Medis yra sutvarkytas taip, kad kiekvienos viršūnės vaikai yra nedidesni už pačią viršūnę.

Iš antrosios savybės seka išvada, kad piramidės šakninėje viršūnėje saugomas didžiausias aibės elementas.



a)



b)

FIGURE: Piramidės pavyzdys: a) dvejetainis medis, b) masyvas

Piramidė yra pilnas dvejetainis medis, todėl jos viršūnes labai patogu saugoti masyve.

Piramidė yra pilnas dvejetainis medis, todėl jos viršūnes labai patogu saugoti masyve.

Tada i-tosios viršūnės a_i vaikai yra masyvo elementai a_{2i}, a_{2i+1}.

Piramidė yra pilnas dvejetainis medis, todėl jos viršūnes labai patogu saugoti masyve.

Tada i-tosios viršūnės a_i vaikai yra masyvo elementai a_{2i}, a_{2i+1}.

Lengvai randame kiekvienos piramidės viršūnės a_i tėvą: ši viršūnė saugoma $j = \lfloor i/2 \rfloor$-ajame masyvo elemente a_j.

Piramidės formavimas.

Tarkime, turime duomenis $e_1, e_2, \ldots, e_N$, iš kurių reikia sudaryti piramidės struktūrą. Šiuos elementus paeiliui talpiname į A masyvą.

Piramidės formavimas.

Tarkime, turime duomenis $e_1, e_2, \ldots, e_N$, iš kurių reikia sudaryti piramidės struktūrą. Šiuos elementus paeiliui talpiname į A masyvą.

Tada visos dvejetainio medžio viršūnės – lapai jau yra sutvarkyti. Piramidės lapų indeksai kinta nuo $(\frac{N}{2} + 1)$ iki N.

Piramidės formavimas.

Tarkime, turime duomenis $e_1, e_2, \ldots, e_N$, iš kurių reikia sudaryti piramidės struktūrą. Šiuos elementus paeiliui talpiname į A masyvą.

Tada visos dvejetainio medžio viršūnės – lapai jau yra sutvarkyti. Piramidės lapų indeksai kinta nuo $(\frac{N}{2} + 1)$ iki N.

Paskui paeiliui imame viršūnes $\frac{N}{2}, \ldots, 1$ ir rekursyviai tikriname piramidės sutvarkymo sąlygą. Jei ji pažeista, sukeičiame vietomis šią viršūnę su didžiausiu jos vaiku.

Piramidės formavimas.

Tarkime, turime duomenis $e_1, e_2, \ldots, e_N$, iš kurių reikia sudaryti piramidės struktūrą. Šiuos elementus paeiliui talpiname į A masyvą.

Tada visos dvejetainio medžio viršūnės – lapai jau yra sutvarkyti. Piramidės lapų indeksai kinta nuo $(\frac{N}{2} + 1)$ iki N.

Paskui paeiliui imame viršūnes $\frac{N}{2}, \ldots, 1$ ir rekursyviai tikriname piramidės sutvarkymo sąlygą. Jei ji pažeista, sukeičiame vietomis šią viršūnę su didžiausiu jos vaiku.

Pratęsiame duotosios viršūnės vietos tikrinimą – tam ir naudojame rekursiją.

Piramidės formavimo algoritmas

```
MakeHeap ()
begin
  (1)   for ( i=1; i ⩽ N; i++ ) do
  (2)         a_i = e_i;
        end do
  (3)   for ( i=N/2; i > 0; i - - ) do
  (4)         HeapDownOrder ( i, N );
        end do
end MakeHeap
```

```
HeapDownOrder ( p, N )
begin
  (1)   i=p;    j = 2 i;
  (2)   while  ( j ⩽ N ) do
  (3)         k = j;
  (4)         if  ( (j+1) ⩽ N ) then
  (5)           if  ( a_{j+1} > a_j ) k = j+1;
                end if
  (6)         if  ( a_i < a_k ) then
  (7)           swap (a_i, a_k);
  (8)           i = k;    j= 2 i;
  (9)         else
  (10)          j = N+1;
                end if
        end do
end HeapDownOrder
```

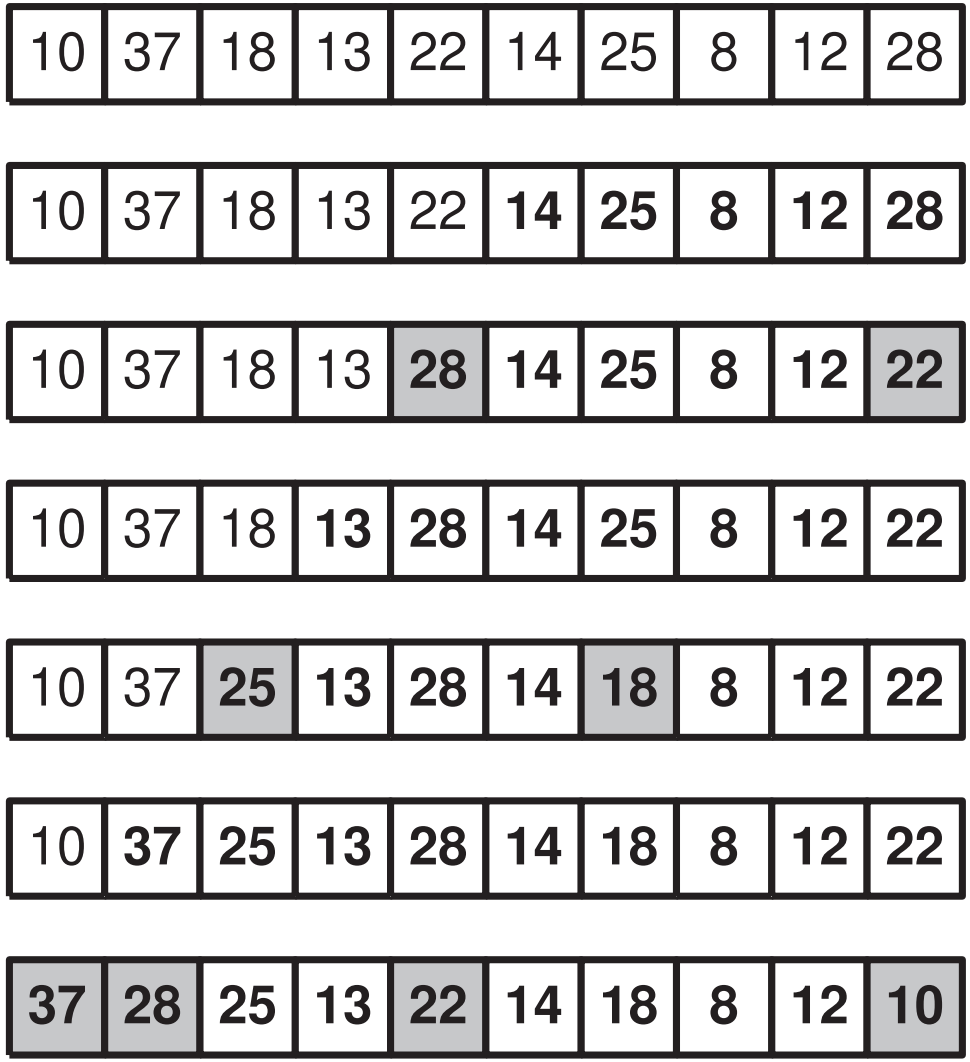


FIGURE: Skaičių masyvo pertvarkymas į piramidės struktūrą. Pusjuodžiu šriftu pavaizduoti jau sutvarkyti elementai, pilka spalva pažymėti tie elementai, kurie buvo sukeisti vietomis, vykdant eilinį piramidės formavimo žingsnį

Kadangi piramidė yra pilnas dvejetainis medis, tai jos aukštis yra nedidesnis už $\log N$.

Kadangi piramidė yra pilnas dvejetainis medis, tai jos aukštis yra nedidesnis už $\log N$.

Kiekvienos elementų tvarkymo operacijos metu įvykdome nedaugiau kaip $2 \log N$ elementų lyginimo veiksmų ir $\log N$ elementų sukeitimų.

Kadangi piramidė yra pilnas dvejetainis medis, tai jos aukštis yra nedidesnis už $\log N$.

Kiekvienos elementų tvarkymo operacijos metu įvykdome nedaugiau kaip $2 \log N$ elementų lyginimo veiksmų ir $\log N$ elementų sukeitimų.

Taigi piramidės formavimo algoritmo sudėtingumas yra

$$L(N) = N \log N, \quad S(N) = \frac{1}{2} N \log N .$$

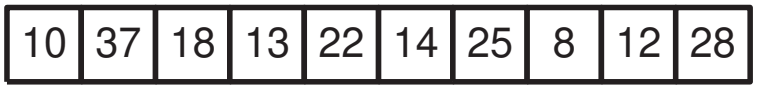
Piramidinis rūšiavimo algoritmas

```
HeapSort ()
begin
 (1)  MakeHeap ();
 (2)  for  ( i=N; i > 1; i = i-1 ) do
 (3)       swap (a_1, a_i);
 (4)       HeapDownOrder (1, i-1);
      end do
end HeapSort
```

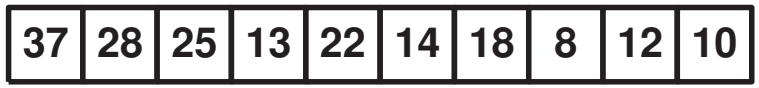
FIGURE: Piramidinis rūšiavimo algoritmas

Algoritmo sudėtingumo įvertinimas. Jau įvertinome aibės elementų pertvarkymo į piramidę skaičiavimo apimtį, ji lygi $\mathcal{O}(N \log N)$. Atlikdami (2) ciklo žingsnius blogiausiuoju atveju papildomai $2N \log N$ kartus lyginame elementus ir $N \log N$ kartų juos sukeičiame vietomis, todėl piramidės rūšiavimo algoritmo sudėtingumas yra

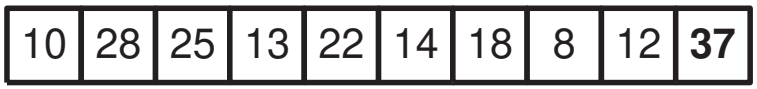
$$L(N) = 3N \log N, \quad S(N) = \frac{3}{2} N \log N .$$


10 37 18 13 22 14 25 8 12 28

a)


37 28 25 13 22 14 18 8 12 10

b)


10 28 25 13 22 14 18 8 12 37

c)


28 22 25 13 10 14 18 8 12 37

d)


12 22 25 13 10 14 18 8 28 37

e)


25 22 18 13 10 14 12 8 28 37

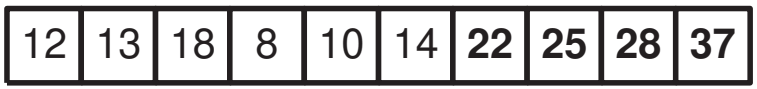
f)


8 22 18 13 10 14 12 25 28 37

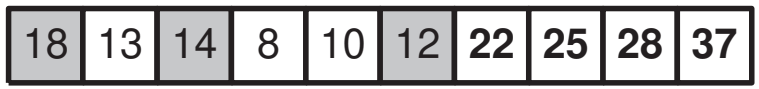
g)


22 13 18 8 10 14 12 25 28 37

h)


12 13 18 8 10 14 22 25 28 37

i)


18 13 14 8 10 12 22 25 28 37

j)

Radix rūšiavimo algoritmas

Tarkime, kad turime A aibę, kurios elementai yra natūralieji k-ženkliai skaičiai:

$$0 \leqslant a_i < 10^k ,$$

tačiau nebūtinai visi šio intevalo skaičiai yra aibės elementai.

Radix rūšiavimo algoritmas

Tarkime, kad turime A aibę, kurios elementai yra natūralieji k-ženkliai skaičiai:

$$0 \leqslant a_i < 10^k ,$$

tačiau nebūtinai visi šio intevalo skaičiai yra aibės elementai.

Rūšiavimo algoritme naudosime dešimtainės aritmetikos taisykles. Pirmiausia visus elementus suskirstome į dešimt poaibių pagal paskutinį rakto skaitmenį (vienetų poziciją).

Radix rūšiavimo algoritmas

Tarkime, kad turime A aibę, kurios elementai yra natūralieji k-ženkliai skaičiai:

$$0 \leqslant a_i < 10^k ,$$

tačiau nebūtinai visi šio intevalo skaičiai yra aibės elementai.

Rūšiavimo algoritme naudosime dešimtainės aritmetikos taisykles. Pirmiausia visus elementus suskirstome į dešimt poaibių pagal paskutinį rakto skaitmenį (vienetų poziciją).

Paskui visus poaibius jų eiliškumo tvarka sujungiame į vieną aibę.

Radix rūšiavimo algoritmas

Tarkime, kad turime A aibę, kurios elementai yra natūralieji k-ženkliai skaičiai:

$$0 \leqslant a_i < 10^k ,$$

tačiau nebūtinai visi šio intevalo skaičiai yra aibės elementai.

Rūšiavimo algoritme naudosime dešimtainės aritmetikos taisykles. Pirmiausia visus elementus suskirstome į dešimt poaibių pagal paskutinį rakto skaitmenį (vienetų poziciją).

Paskui visus poaibius jų eiliškumo tvarka sujungiame į vieną aibę.

Gautąją aibę vėl skirstome į dešimt poaibių pagal priešpaskutinį rakto skaitmenį (dešimčių poziciją).

Radix rūšiavimo algoritmas

Tarkime, kad turime A aibę, kurios elementai yra natūralieji k-ženkliai skaičiai:

$$0 \leqslant a_i < 10^k ,$$

tačiau nebūtinai visi šio intevalo skaičiai yra aibės elementai.

Rūšiavimo algoritme naudosime dešimtainės aritmetikos taisykles. Pirmiausia visus elementus suskirstome į dešimt poaibių pagal paskutinį rakto skaitmenį (vienetų poziciją).

Paskui visus poaibius jų eiliškumo tvarka sujungiame į vieną aibę.

Gautąją aibę vėl skirstome į dešimt poaibių pagal priešpaskutinį rakto skaitmenį (dešimčių poziciją).

Šį procesą kartojame k kartų.

Surūšiuosime dviženklių sveikųjų skaičių masyvą

$$A = (73, 29, 92, 14, 74, 45, 54, 18, 3, 97, 9, 61, 11, 63, 35, 37).$$

Surūšiuosime dviženklių sveikųjų skaičių masyvą

$$A = (73,\ 29,\ 92,\ 14,\ 74,\ 45,\ 54,\ 18,\ 3,\ 97,\ 9,\ 61,\ 11,\ 63,\ 35,\ 37).$$

Aibę suskirstome į dešimt poaibių pagal paskutinį rakto skaitmenį

$$\begin{aligned}
0 &: \\
1 &: \ 61,\ 11\,, \\
2 &: \ 92 \\
3 &: \ 73,\ 3,\ 63\,, \\
4 &: \ 14,\ 74,\ 54\,, \\
5 &: \ 45,\ 35\,, \\
6 &: \\
7 &: \ 97,\ 37\,, \\
8 &: \ 18\,, \\
9 &: \ 29,\ 9\,.
\end{aligned}$$

Visus poaibius jų eiliškumo tvarka sujungiame į vieną aibę

$$A = (61, 11, 92, 73, 3, 63, 14, 74, 54, 45, 35, 97, 37, 18, 29, 9).$$

Visus poaibius jų eiliškumo tvarka sujungiame į vieną aibę

$$A = (61, 11, 92, 73, 3, 63, 14, 74, 54, 45, 35, 97, 37, 18, 29, 9).$$

Gautąją aibę vėl skirstome į dešimt poaibių dabar pagal pirmąjį rakto skaitmenį

$$\begin{aligned}
0 &: \ 03, 09, \\
1 &: \ 11, 14, 18, \\
2 &: \ 29, \\
3 &: \ 35, 37, \\
4 &: \ 45, \\
5 &: \ 54, \\
6 &: \ 61, 63, \\
7 &: \ 73, 74, \\
8 &: \\
9 &: \ 92, 97.
\end{aligned}$$

Sujungę poaibius, gauname surūšiuotą elementų aibę

$$A = (3,\ 9,\ 11,\ 14,\ 18,\ 29,\ 35,\ 37,\ 45,\ 54,\ 61,\ 63,\ 73,\ 74,\ 92,\ 97).$$

Sujungę poaibius, gauname surūšiuotą elementų aibę

$$A = (3, 9, 11, 14, 18, 29, 35, 37, 45, 54, 61, 63, 73, 74, 92, 97).$$

Puiku, šiame pavyzdyje algoritmas sėkmingai surūšiavo duomenis.
Bet ar taip bus visada?

Sujungę poaibius, gauname surūšiuotą elementų aibę

$$A = (3, 9, 11, 14, 18, 29, 35, 37, 45, 54, 61, 63, 73, 74, 92, 97).$$

Puiku, šiame pavyzdyje algoritmas sėkmingai surūšiavo duomenis. Bet ar taip bus visada?

Įrodysime, kad įvykdžius Radix algoritmą visada teisingai surūšiuojame aibę.

Sujungę poaibius, gauname surūšiuotą elementų aibę

$$A = (3, 9, 11, 14, 18, 29, 35, 37, 45, 54, 61, 63, 73, 74, 92, 97).$$

Puiku, šiame pavyzdyje algoritmas sėkmingai surūšiavo duomenis. Bet ar taip bus visada?

Įrodysime, kad įvykdžius Radix algoritmą visada teisingai surūšiuojame aibę.

Užtenka išnagrinėti dviženklių skaičių atvejį, o bendrojo atvejo teisingumas įrodomas indukcijos metodu.

Imkime du dviženklius skaičius X ir Y:

$$X = 10\,a + b, \quad Y = 10\,c + d, \quad 0 \leqslant a, b, c, d \leqslant 9.$$

Imkime du dviženklius skaičius X ir Y:

$$X = 10\,a + b, \quad Y = 10\,c + d, \quad 0 \leqslant a, b, c, d \leqslant 9.$$

Nelygybė $X < Y$ yra teisinga, jei

$$(a < c) \quad \text{arba} \quad (a = c)\,\&\,(b < d).$$

Imkime du dviženklius skaičius X ir Y:

$$X = 10\,a + b, \quad Y = 10\,c + d, \quad 0 \leqslant a, b, c, d \leqslant 9.$$

Nelygybė $X < Y$ yra teisinga, jei

$$(a < c) \quad \text{arba} \quad (a = c)\,\&\,(b < d).$$

Jei $a < c$, tai antrajame Radix rūšiavimo algoritmo žingsnyje X patenka į poaibį su mažesniu numeriu nei Y.

Imkime du dviženklius skaičius X ir Y:

$$X = 10\,a + b, \quad Y = 10\,c + d, \quad 0 \leqslant a, b, c, d \leqslant 9.$$

Nelygybė $X < Y$ yra teisinga, jei

$$(a < c) \quad \text{arba} \quad (a = c)\,\&\,(b < d).$$

Jei $a < c$, tai antrajame Radix rūšiavimo algoritmo žingsnyje X patenka į poaibį su mažesniu numeriu nei Y.

Antruoju atveju pirmuoju Radix rūšiavimo algoritmo žingsniu X pateko į poaibį su mažesniu numeriu nei Y. Po antrojo žingsnio abu elementai pateks į tą patį poaibį, bet X bus įtrauktas anksčiau.

Imkime du dviženklius skaičius X ir Y:

$$X = 10\,a + b, \quad Y = 10\,c + d, \quad 0 \leqslant a, b, c, d \leqslant 9.$$

Nelygybė $X < Y$ yra teisinga, jei

$$(a < c) \quad \text{arba} \quad (a = c)\,\&\,(b < d).$$

Jei $a < c$, tai antrajame Radix rūšiavimo algoritmo žingsnyje X patenka į poaibį su mažesniu numeriu nei Y.

Antruoju atveju pirmuoju Radix rūšiavimo algoritmo žingsniu X pateko į poaibį su mažesniu numeriu nei Y. Po antrojo žingsnio abu elementai pateks į tą patį poaibį, bet X bus įtrauktas anksčiau.

Taigi abiem atvejais elementai surūšiuojami teisingai.

Algoritmo sudėtingumo analizė

Įvertinsime, kiek bazinių veiksmų reikia atlikti rūšiuojant $N < 10^k$ elementų.

Algoritmo sudėtingumo analizė

Įvertinsime, kiek bazinių veiksmų reikia atlikti rūšiuojant $N < 10^k$ elementų.

Kiekvienu Radix algoritmo žingsniu tokių veiksmų yra N, o žingsnių skaičius $k = \lg N$, todėl algoritmo apimtis yra

$$N \lg N.$$

Algoritmo sudėtingumo analizė

Įvertinsime, kiek bazinių veiksmų reikia atlikti rūšiuojant $N < 10^k$ elementų.

Kiekvienu Radix algoritmo žingsniu tokių veiksmų yra N, o žingsnių skaičius $k = \lg N$, todėl algoritmo apimtis yra

$$N \lg N.$$

Atkreipkite dėmesį į tai, kad šioje formulėje naudojame dešimtainį logaritmą

$$\log 1000 \approx 10, \quad \lg 1000 = 3.$$

Išorinio rūšiavimo uždavinys

Nagrinėsime rūšiavimo algoritmus, kai duomenų yra tiek daug, kad jie visi netelpa operatyviojoje kompiuterio atmintyje ir juos saugome talpesnėje, bet daug lėtesnėje išorinėje atmintyje.

Išorinio rūšiavimo uždavinys

Nagrinėsime rūšiavimo algoritmus, kai duomenų yra tiek daug, kad jie visi netelpa operatyviojoje kompiuterio atmintyje ir juos saugome talpesnėje, bet daug lėtesnėje išorinėje atmintyje.

Tada lėčiausia operacija yra duomenų perrašymas iš sparčiosios atminties į išorinę atmintį ir atvirkščiai. Svarbiausia yra minimizuoti tokių veiksmų apimtį.

Pateiksime suliejimo algoritmo modifikaciją, pritaikytą išorinio rūšiavimo uždaviniui spręsti.

Pateiksime suliejimo algoritmo modifikaciją, pritaikytą išorinio rūšiavimo uždaviniui spręsti.

Reikia surūšiuoti N duomenų, saugomų rinkmenoje F. Tarkime, kad operatyviojoje atmintyje telpa tik M elementų.

- Iš rinkmenos F skaitome M dydžio duomenų blokus, juos rūšiuojame kokiu nors sparčiuoju vidinio rūšiavimo algoritmu ir paeiliui užrašome į rinkmenas $F1, F2$. Paskutinio bloko ilgis gali būti ir mažesnis už M.

Pateiksime suliejimo algoritmo modifikaciją, pritaikytą išorinio rūšiavimo uždaviniui spręsti.

Reikia surūšiuoti N duomenų, saugomų rinkmenoje F. Tarkime, kad operatyviojoje atmintyje telpa tik M elementų.

- Iš rinkmenos F skaitome M dydžio duomenų blokus, juos rūšiuojame kokiu nors sparčiuoju vidinio rūšiavimo algoritmu ir paeiliui užrašome į rinkmenas $F1, F2$. Paskutinio bloko ilgis gali būti ir mažesnis už M.
- Suliejimo algoritmu sujungiame M ilgio blokus iš rinkmenų $F1, F2$, gautuosius $2M$ ilgio blokus įrašome paeiliui į rinkmenas $F3, F4$. Algoritmą kartojame tol, kol gauname surūšiuotą N ilgio bloką, užrašytą vienoje iš rinkmenų.

Skaičių masyvo rūšiavimas suliejimo algoritmu.

Rinkmenoje F užrašytas skaičių masyvas, kurio ilgis $N = 29$:

$$(4, 5, 2, 8, 4, 1, 7, 9, 2, 3, 0, 3, 8, 6, 2, 4, 9, 3, 9, 5, 0, 4, 6, 2, 5, 3, 5, 1, 0).$$

Imkime $M = 3$, tada masyvo rūšiavimo eigos pradžia yra tokia:

$M = 3$:

$\mathrm{F1} = (\,2, 4, 5 \mid 2, 7, 9 \mid 2, 6, 8 \mid 0, 5, 9 \mid 3, 5, 5\,)$

$\mathrm{F2} = (\,1, 4, 8 \mid 0, 3, 3 \mid 3, 4, 9 \mid 2, 4, 6 \mid 0, 1\,)$

$M = 6$:

$\mathrm{F3} = (\,1, 2, 4, 4, 5, 8 \mid 2, 3, 4, 6, 8, 9 \mid 0, 1, 3, 5, 5\,)$

$\mathrm{F4} = (\,0, 2, 3, 3, 7, 9 \mid 0, 2, 4, 5, 6, 9\,)$

$M = 12$:

$\mathrm{F1} = (\,0, 1, 2, 2, 3, 3, 4, 4, 5, 7, 8, 9 \mid 0, 1, 3, 5, 5\,)$

$\mathrm{F2} = (\,0, 2, 2, 3, 4, 4, 5, 6, 6, 8, 9, 9\,)$

Įvertinsime, kiek kartų duomenys perrašomi iš rinkmenos į sparčiąją operatyviąją kompiuterio atmintį. Tegul $N = 2^k M$.

Įvertinsime, kiek kartų duomenys perrašomi iš rinkmenos į sparčiąją operatyviąją kompiuterio atmintį. Tegul $N = 2^k M$.

Priminsime, kad kaip tik šios operacijos vykdymas ir sudaro išorinio rūšiavimo algoritmo didžiąją sąnaudų dalį.

Įvertinsime, kiek kartų duomenys perrašomi iš rinkmenos į sparčiąją operatyviąją kompiuterio atmintį. Tegul $N = 2^k M$.

Priminsime, kad kaip tik šios operacijos vykdymas ir sudaro išorinio rūšiavimo algoritmo didžiąją sąnaudų dalį.

Kiekvienu etapu nuskaitomi ir įrašomi visi N elementai, bendrasis etapų skaičius yra $(k+1)$, todėl iš viso atliekame

$$N \log\left(\frac{N}{M}\right)$$

duomenų skaitymo veiksmų.