

# ***Algoritmai ir duomenų struktūros***

4 paskaita

2021-03-03

# ***Papildomos užduoties sprendimai***

## ***Problem C. Card Game***

- Olimpiada:  
<https://neerc.ifmo.ru/archive/2013/western/standings.html>
- Užduotys:  
<http://fpmi.bsu.by/ImgFpmi/Cache/46233.pdf>
- C užduoties sprendimai:
  - Audrius Mardosas: <https://onlinegdb.com/ldtXmX6pA> (C++)
  - Tomas Šinkūnas: <https://jsfiddle.net/renderTom/aydsmne0> (JavaScript)
  - Dėstytojas: <https://www.onlinegdb.com/FPpmGa1iQ> (Python)

# ***Duomenų rūšiavimo poreikis***

- Rūšiavimas yra viena pagrindinių kompiuterio atliekamų operacijų (vidutiniškai 25% viso skaičiavimo laiko kompiuteris skiria rūšiavimui).
- Rūšiavimo procesą sąlygoja:
  - Duomenų tvarkymo apibrėžimas.
  - Duomenų struktūra.
  - Įvairūs prioritetai.
  - Atminties panaudojimas:
    - Vidinės atminties panaudojimas.
    - Išorinės atminties panaudojimas.
  - Lygiagretus rūšiavimas ir t. t.

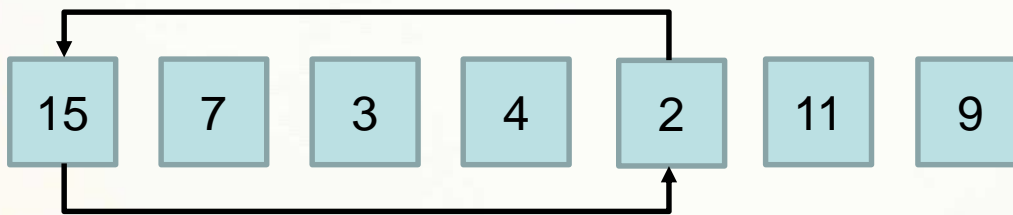
# ***Rūšiavimo (rikiavimo) algoritmai***

1. Išrinkimo algoritmas (*angl.* Selection sort)
2. Burbuliuko algoritmas (*angl.* Bubble sort)
3. Įterpimo algoritmas (*angl.* Insertion sort)
4. Rikiavimas Šelo metodu (*angl.* Shell sort)
5. Spartaus rikiavimo algoritmas (*angl.* Quick sort)
6. Sąlajos rikiavimas (*angl.* Merge sort)
7. Išorinis rikiavimas (*angl.* External sorting)
8. Piramidės rikiavimas (*angl.* Heap sort)
9. Skaitmeninis rikiavimas (*angl.* Radix sort)

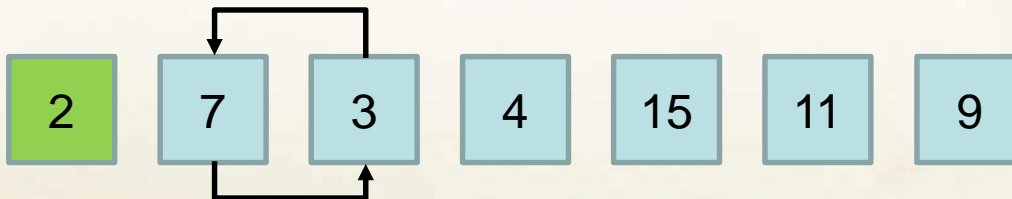
# Išrinkimo algoritmas (angl. Selection sort)

Rikiavimo taisyklės:

1) Iš turimo duomenų sąrašo išrenkamas mažiausias elementas ir sukeičiamas (*swap*) su pirmuoju elementu:



2) Principas kartojamas sąrašui be pirmojo elemento ir t. t. kol gaunamas išrikiuotas sąrašas didėjimo tvarka:



# Burbuliuko algoritmas (angl. Bubble sort)

Algoritmo idėja – nuosekliai iš kairės į dešinę lyginti gretimų elementų (a, b) poras, jei  $a > b$ , sukeisti juos vietomis (*swap*).

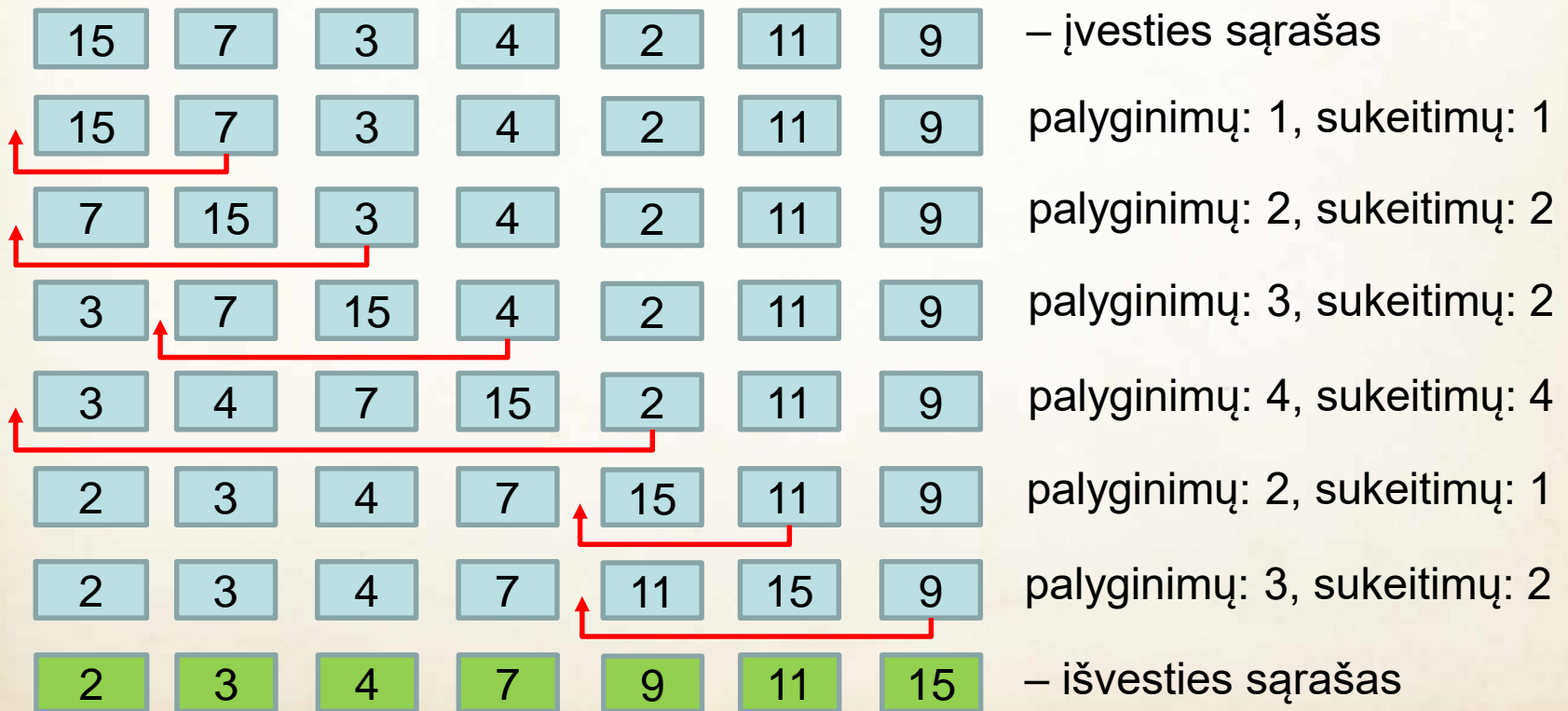
Taip elgiantis, randamas didžiausias sąrašo elementas.

Po to principas kartojamas sąrašui be paskutiniojo elemento ir t. t.:

15	7	3	4	2	11	9	– įvesties sąrašas
7	3	4	2	11	9	15	palyginimų: 6, sukeitimų: 6
3	4	2	7	9	11	15	palyginimų: 5, sukeitimų: 4
3	2	4	7	9	11	15	palyginimų: 4, sukeitimų: 1
2	3	4	7	9	11	15	palyginimų: 3, sukeitimų: 1
2	3	4	7	9	11	15	palyginimų: 2, sukeitimų: 0
2	3	4	7	9	11	15	palyginimų: 1, sukeitimų: 0
2	3	4	7	9	11	15	– išvesties sąrašas

# Įterpimo algoritmas (angl. Insertion sort)

Algoritmo idėja: elementas  $x_i$  yra palyginamas su prieš jį esančiu  $x_{i-1}$ , jei neatitinka tvarka, t. y.  $x_{i-1} > x_i$ ,  $x_i$  lyginamas su dar ankstesniu  $x_{i-2}$  ir t. t. kol tvarka atitinka, t. y.  $x_j < x_i$ . Tada  $x_i$  perkeliamas po  $x_j$  (poromis elementus sukeičiant). Tokiu principu pereinami visi sąrašo elementai, kol gaunamas išrikiuotas sąrašas didėjimo tvarka:



# Rikiavimas Šelo metodu (angl. Shell sort)

Algoritmas sukurtas 1959 m. Donaldo L. Šelo.



Šelo rikiavimo metodo idėja:

1. Padalinti sąrašą į  $n \leftarrow \lfloor n / 2 \rfloor$  posąrašų.
2. Kiekvieną posąrašį išrikiuoti didėjimo tvarka pagal taisykles (tai ekvivalentu insertion sort algoritmui):
  1. Posąrašio gretimi elementai  $x_i$  ir  $x_{i+1}$  sukeičiami vietomis jei  $x_i > x_{i+1}$ .
  2. Jei  $x_i$  ir  $x_{i+1}$  sukeičiami vietomis, reikia patikrinti, ar reikia sukeisti vietomis  $x_{i-1}$  ir  $x_i$ .
  3. 2 žingsnis kartojamas kol nebereikia keisti gretimų elementų ir prieinama prie 1 žingsnio tikrinant  $x_{i+1}$  ir  $x_{i+2}$  porą.
  4. Posąrašis tampa išrikiuotas 1 žingsniu perėjus visas elementų poras iš kairės į dešinę.
3. Posąrašus atgal sujungti į sąrašą.
4. Kartoti 2 ir 3 žingsnius parinkus  $n \leftarrow \lfloor n / 2 \rfloor$  posąrašų ir t. t. kol  $n = 1$ .



# ***Rikiavimas Šelo metodu (angl. Shell sort)***

Pavyzdys. Tegu turime sąrašą ( $n=11$ ):

[7, 8, 99, 2, 5, 12, 34, 54, 2, 3, 1].

$n \leftarrow \lfloor n / 2 \rfloor = 5$ , todėl sudaromi 5 posąrašiai pagal [1,2,3,4,5,1,2,3,4,5,1]:

[7, 12, 1], [8, 34], [99, 54], [2, 2], [5, 3].

Posąrašiai išrikiuojami didėjimo tvarka: [1,7,12], [8,34], [54,99], [2,2], [3,5].

Tai atitinka sąrašą: [1, 8, 54, 2, 3, 7, 34, 99, 2, 5, 12].

$n \leftarrow \lfloor n / 2 \rfloor = 2$ , todėl sudaromi 2 posąrašiai pagal [1,2,1,2,1,2,1,2,1,2,1]:

[1, 54, 3, 34, 2, 12], [8, 2, 7, 99, 5].

Posąrašiai išrikiuojami didėjimo tvarka: [1,2,3,12,34,54], [2, 5, 7, 8, 99].

Tai atitinka sąrašą: [1, 2, 2, 5, 3, 7, 12, 8, 34, 99, 54].

$n \leftarrow \lfloor n / 2 \rfloor = 1$ , todėl posąrašis lygus sąrašui, kurį belieka išrikiuoti:

[1, 2, 2, 3, 5, 7, 8, 12, 34, 54, 99].

## ***Spartaus rikiavimo algoritmas (angl. Quick sort)***

**Algoritmo idėja.** Tarkime turime sąrašą ir vidurinį jo elementą Pivot (7):

[1, 8, 54, 2, 3, **7**, 34, 99, 2, 5, 12].

Visus elementus, mažesnius už 7 perrašome kairėje pusėje, visus didesnius – dešinėje:

[1, 2, 3, 2, 5, **7**, 8, 54, 34, 99, 12].

Tą patį metodą taikome posarašiams [1, 2, **3**, 2, 5] ir [8, 54, **34**, 99, 12]:

[1, 2, 2, **3**, 5, **7**, 8, 12, **34**, 54, 99].

Ir galiausiai posarašiams [1, **2**, 2], [8, **12**], [54, **99**]:

[1, **2**, 2, **3**, 5, **7**, 8, **12**, **34**, 54, **99**].

Iki kol sąrašas tampa išrikiuotas (posarašiai tik iš 1 elemento):

[1, **2**, 2, **3**, 5, **7**, 8, **12**, **34**, 54, **99**].

## *Spartaus rikiavimo algoritmo pavyzdys*

Pivot  
[ 7, 13, 5, 11, 4, 1, 6 ]  
↑

7 > Pivot, todėl baltoji rodyklė pastumiama dešinėn, juodoji lieka savo vietoje.

Pivot  
[ 7, 13, 5, 11, 4, 1, 6 ]  
↑ ↑

13 > Pivot, todėl baltoji rodyklė pastumiama dešinėn, juodoji lieka savo vietoje.

## *Spartaus rikiavimo algoritmo pavyzdys*

Pivot  
[ 7, 13, 5, 11, 4, 1, 6 ]  
↑           ↑

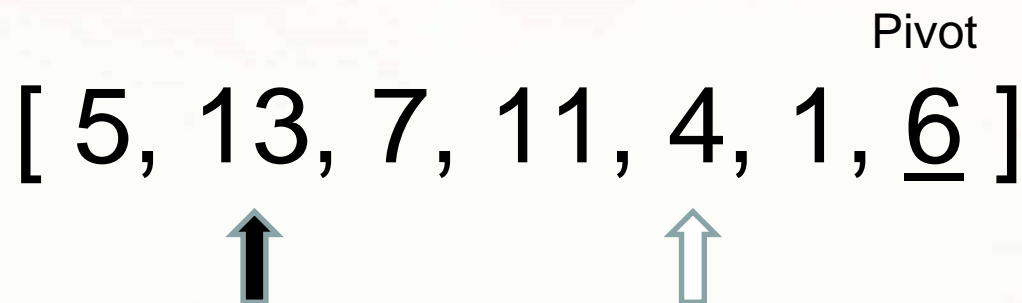
5 < Pivot, todėl sukeičiamos rodyklių reikšmės ir abi rodyklės pastumiamos dešinėn.

Pivot  
[ 5, 13, 7, 11, 4, 1, 6 ]  
          ↑           ↑

11 > Pivot, todėl baltoji rodyklė pastumiama dešinėn, juodoji lieka savo vietoje.

## *Spartaus rikiavimo algoritmo pavyzdys*

Pivot  
[ 5, 13, 7, 11, 4, 1, 6 ]



4 < Pivot, todėl sukeičiamos rodyklių reikšmės ir abi rodyklės pastumiamos dešinėn.

Pivot  
[ 5, 4, 7, 11, 13, 1, 6 ]



1 < Pivot, todėl sukeičiamos rodyklių reikšmės ir abi rodyklės pastumiamos dešinėn.

## Spartaus rikiavimo algoritmo pavyzdys

Pivot

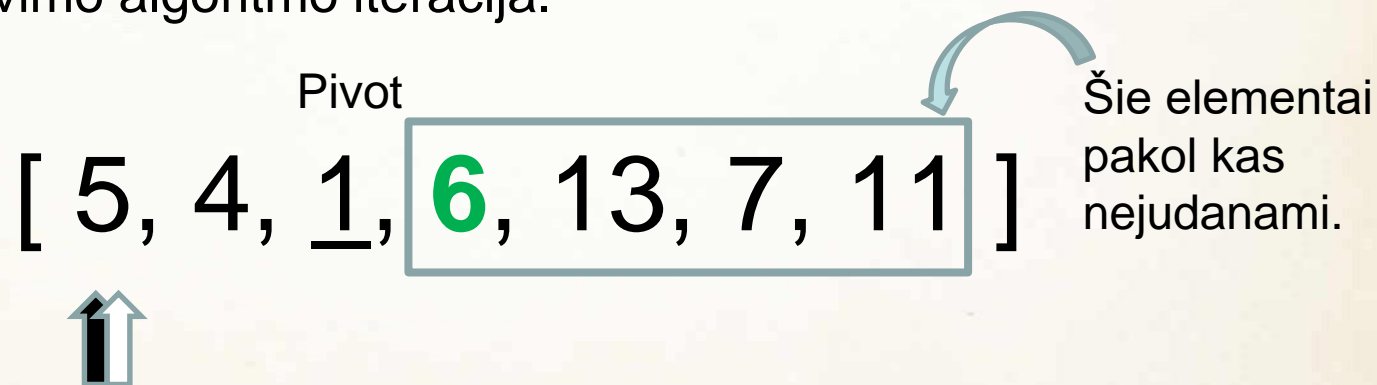
[ 5, 4, 1, 11, 13, 7, 6 ]



11 > pivot, todėl sukeičiamos rodyklių reikšmės ir užbaigiama rekursyvi spartaus rikiavimo algoritmo iteracija.

Pivot

[ 5, 4, 1, 6, 13, 7, 11 ]



Šie elementai pakol kas nejudanami.

Analogiška iteracija kartojama sąrašui [ 5, 4, 1 ].

5 > Pivot, todėl baltoji rodyklė pastumiama dešinėn, juodoji lieka savo vietoje.

## *Spartaus rikiavimo algoritmo pavyzdys*

Pivot  
[ 5, 4, 1, 6, 13, 7, 11 ]  
↑ ↑

4 > Pivot, todėl baltoji rodyklė pastumiama dešinėn, juodoji lieka savo vietoje.

[ 5, 4, 1, 6, 13, 7, 11 ]  
↑ ↑

5 > pivot, todėl sukeičiamos rodyklių reikšmės ir užbaigiama rekursyvi spartaus rikiavimo algoritmo iteracija.

## *Spartaus rikiavimo algoritmo pavyzdys*

Pivot  
[ **1**, 4, 5, **6**, 13, 7, 11 ]  
↑

4 < Pivot, todėl abi rodyklės pastumiamos dešinėn.

Pivot  
[ **1**, 4, 5, **6**, 13, 7, 11 ]  
↑

Pivot lieka savo vietoje ir vykdoma kita rekursyvi iteracija.

Pivot  
[ **1**, 4, **5**, **6**, 13, 7, 11 ]  
↑

Pivot lieka savo vietoje ir vykdoma kita rekursyvi iteracija.



## *Spartaus rikiavimo algoritmo pavyzdys*

Pivot  
[ **1**, **4**, **5**, **6**, 13, 7, 11 ]



13 > Pivot, todėl baltoji rodyklė pastumiama dešinėn, juodoji lieka savo vietoje.

Pivot  
[ **1**, **4**, **5**, **6**, 13, 7, 11 ]



7 < Pivot, todėl sukeičiamos rodyklių reikšmės ir abi rodyklės pastumiamos dešinėn.



## *Spartaus rikiavimo algoritmo pavyzdys*

Pivot  
[ 1, 4, 5, 6, 7, 11, 13 ]  
↑

Pivot lieka savo vietoje, užbaigiamas spartus rikiavimas, išvedami rezultatai:

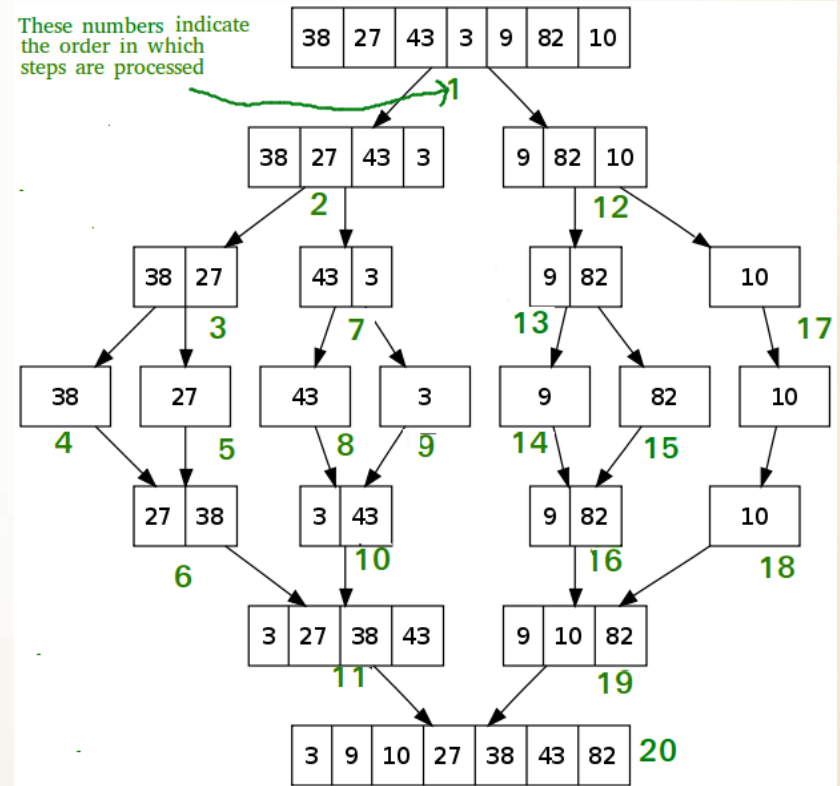
[ 1, 4, 5, 6, 7, 11, 13 ]

# Sąlajos rikiavimas (angl. Merge sort)

Algoritmo idėja: sąrašą rekursyviai išskaidyti į posąrašius, iki 1 elemento, po posąrašius rekursyviai apjungti išrikiuojant jų elementus didėjimo tvarka. Algoritmo pavyzdys:

Sąlajos rikiavimas dar vadinamas:

- Rikiavimu sujungiant
- Suliejimo metodu
- Vidiniu suliejimo metodu



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

Šis metodas taikomas rikiuoti didelės apimties failams ar rinkiniams, kurie netelpa kompiuterio vidinėje atmintyje (pavyzdžiui, disko defragmentavimas). Rikiavimo pavyzdys:

**Ivesties duomenys:** Failas\_1: [15, 1, 3, 11, 5, 7, 9, 5, 4]

**Skirstymas:** Failas\_2: [15, 3, 5, 9, 4], Failas\_3: [1, 11, 7, 5]

**Jungimas:** Failas\_1: [(1, 15), (3, 11), (5, 7), (5, 9), (4)]

**Skirstymas:** Failas\_2: [(1, 15), (5, 7), (4)], Failas\_3: [(3, 11), (5, 9)]

**Jungimas:** Failas\_1: [(1, 3, 11, 15), (5, 5, 7, 9), (4)]

**Skirstymas:** Failas\_2: [(1, 3, 11, 15), (4)], Failas\_3: [(5, 5, 7, 9)]

**Jungimas:** Failas\_1: [(1, 3, 5, 5, 7, 9, 11, 15), (4)]

**Skirstymas:** Failas\_2: [1, 3, 5, 5, 7, 9, 11, 15], Failas\_3: [4]

**Jungimas (išvestis):** Failas\_1: [1, 3, 4, 5, 5, 7, 9, 11, 15]

Neprigiję žodžio „failas“ sinonimai: byla, rinkmena, tvarkmena ☺.

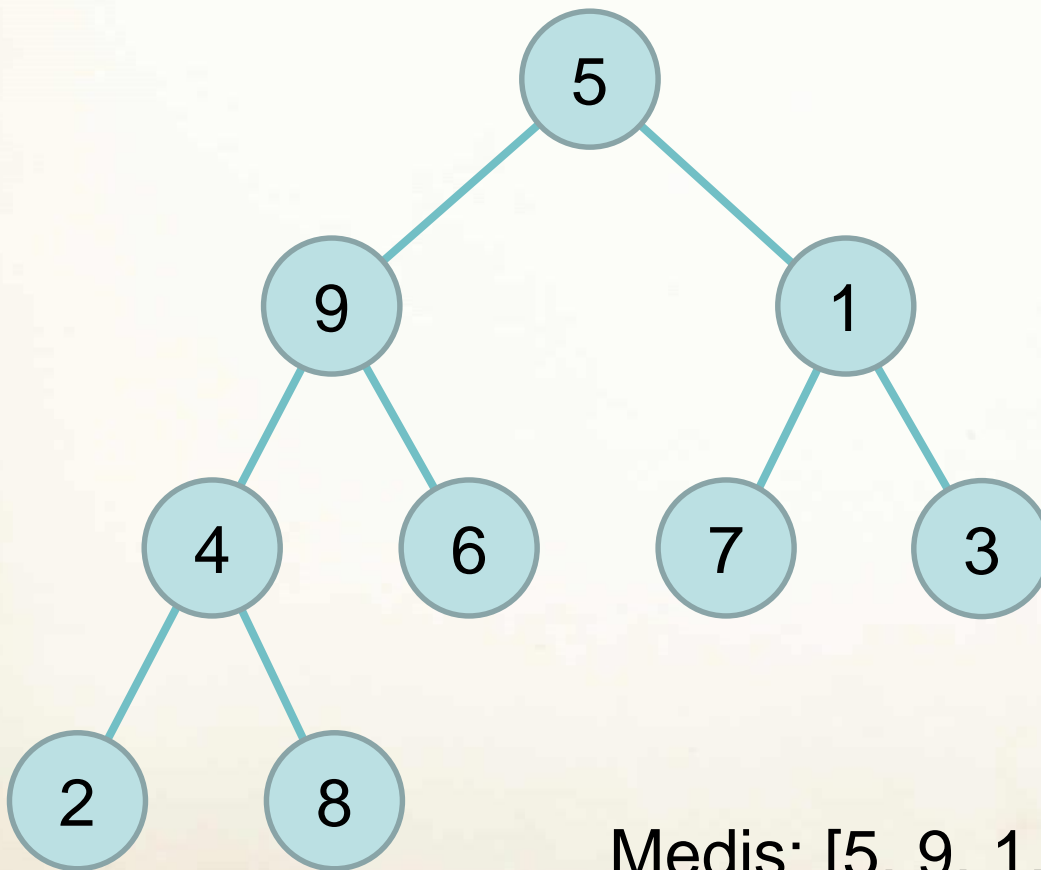
# ***Piramidės rikiavimas (angl. Heap sort)***

Piramidės rikiavimo idėja:

1. Pagal duoto sąrašo masyvo indeksus sudaromas pilnas dvejetainis medis.
2. Kylant iš apačios į viršų po to leidžiantis iš viršaus į apačią, medyje, jei reikia, sukeičiamos gretimų viršūnių reikšmės kol sudaroma *max Heap* struktūra (tokia struktūra kur tėvų reikšmės nemažesnės už vaikų).
3. Paskutinis sąrašo elementas sukeičiamas su *max Heap* struktūros medžio šaknies reikšme (ir vėl suardoma *max Heap* struktūra).
4. Leidžiantis iš viršaus į apačią, medyje, jei reikia, sukeičiamos gretimų viršūnių reikšmės, kol atstatoma *max Heap* struktūra.
5. Toliau kartojami 3 ir 4 žingsniai sąrašė kaskart eliminuojant paskutinįjį elementą iki kol atbuline eiga sąrašas išrikiuojamas didėjimo tvarka.

## *Piramidės rikiavimo pavyzdys*

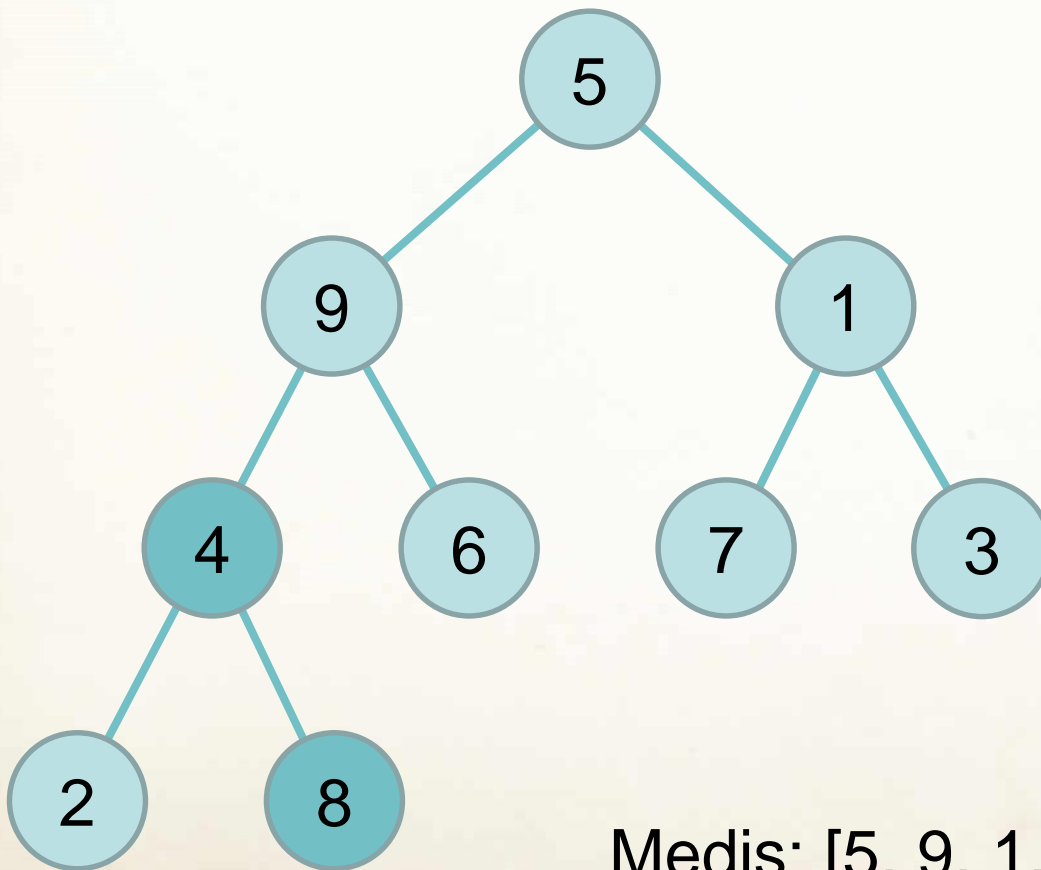
Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Medis: [5, 9, 1, 4, 6, 7, 3, 2, 8].

## *Piramidės rikiavimo pavyzdys*

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].

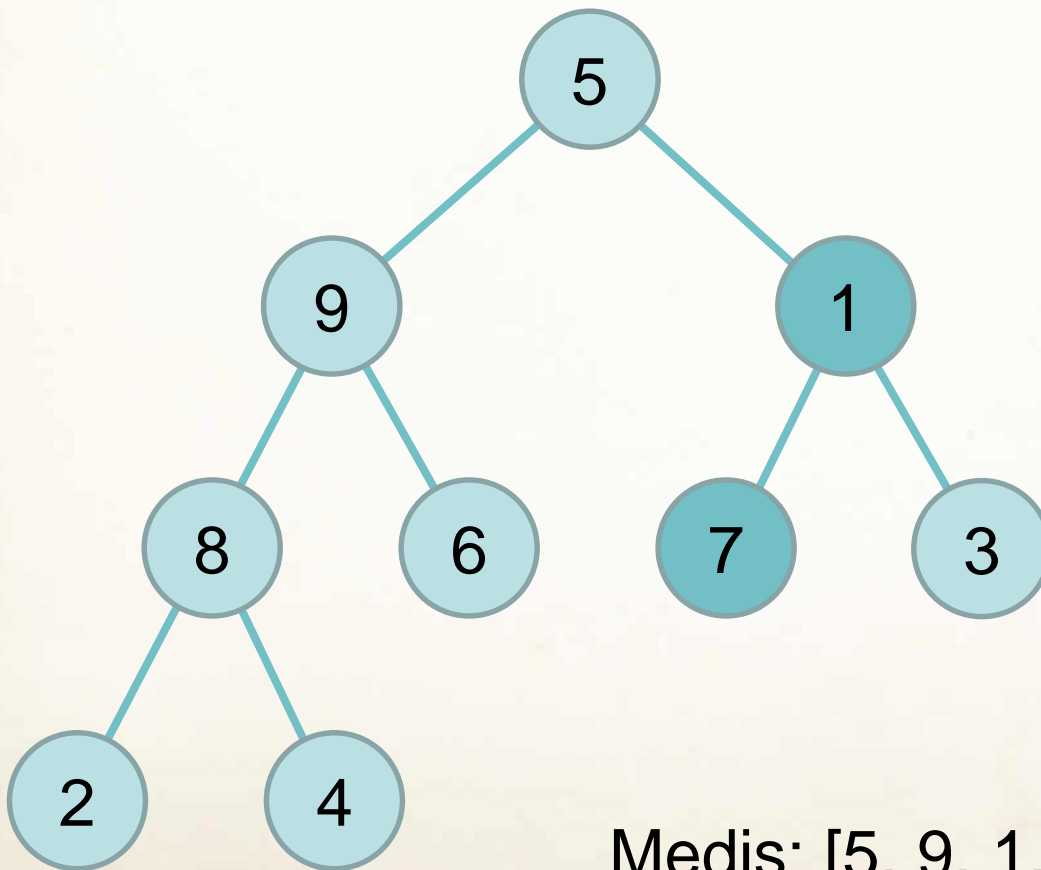


Medis: [5, 9, 1, 4, 6, 7, 3, 2, 8].



## *Piramidės rikiavimo pavyzdys*

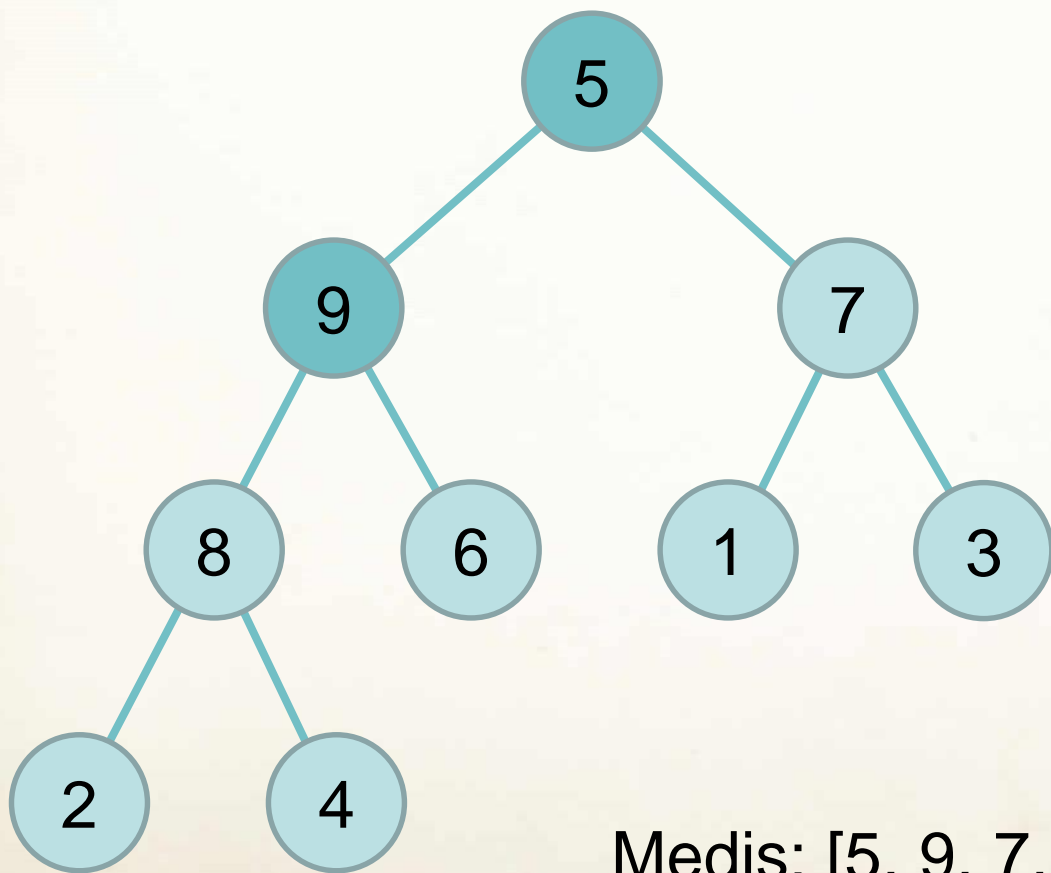
Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Medis: [5, 9, 1, 8, 6, 7, 3, 2, 4].

## *Piramidės rikiavimo pavyzdys*

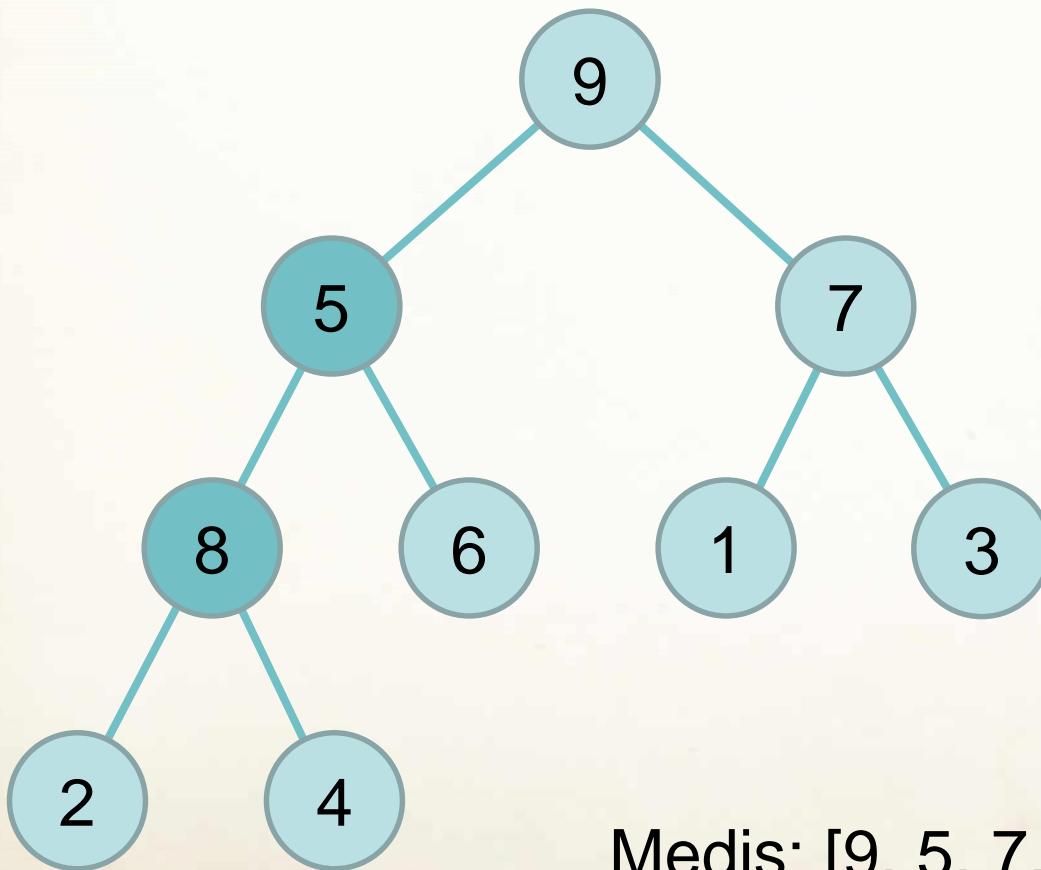
Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Medis: [5, 9, 7, 8, 6, 1, 3, 2, 4].

## *Piramidės rikiavimo pavyzdys*

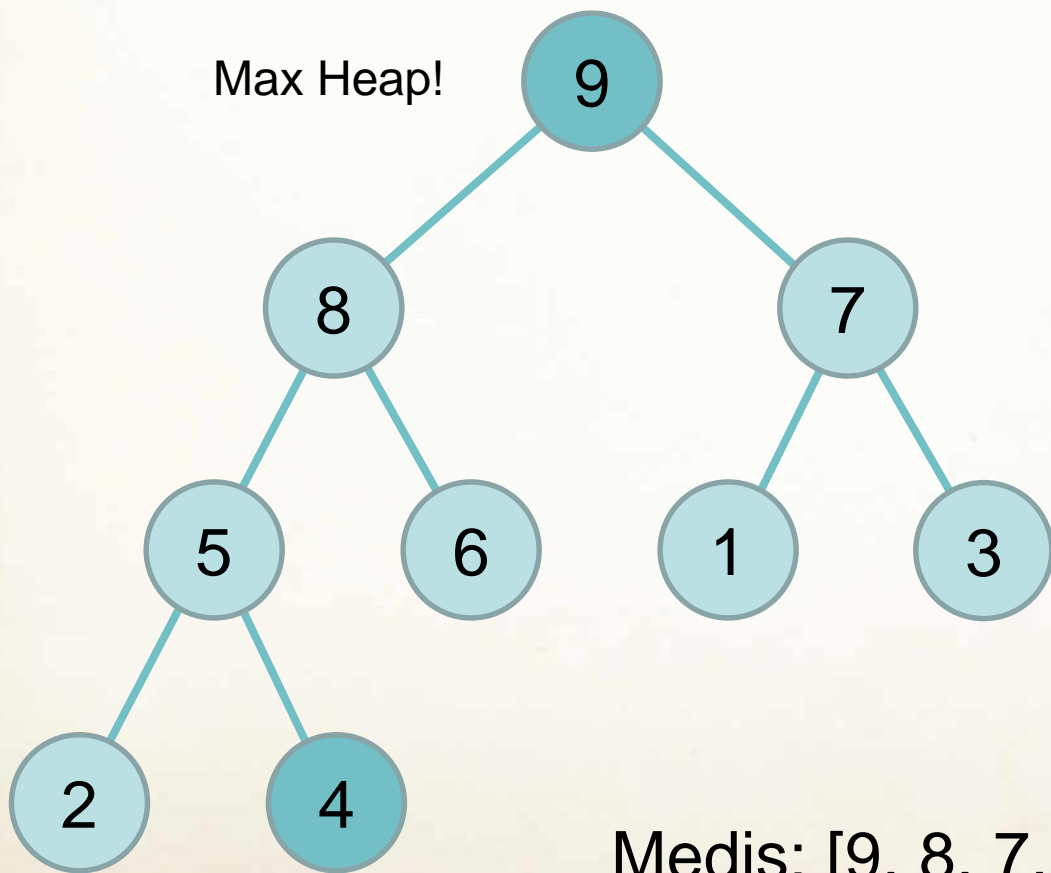
Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Medis: [9, 5, 7, 8, 6, 1, 3, 2, 4].

## *Piramidės rikiavimo pavyzdys*

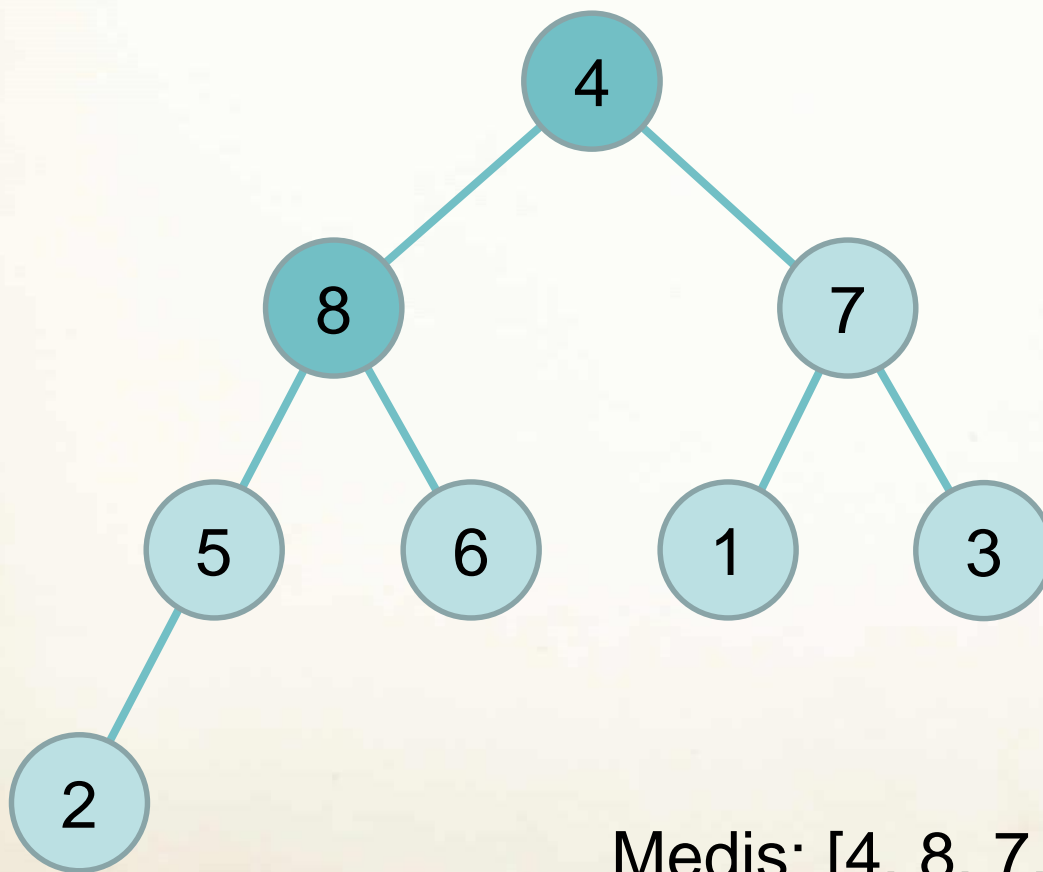
Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Medis: [9, 8, 7, 5, 6, 1, 3, 2, 4].

# *Piramidės rikiavimo pavyzdys*

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].

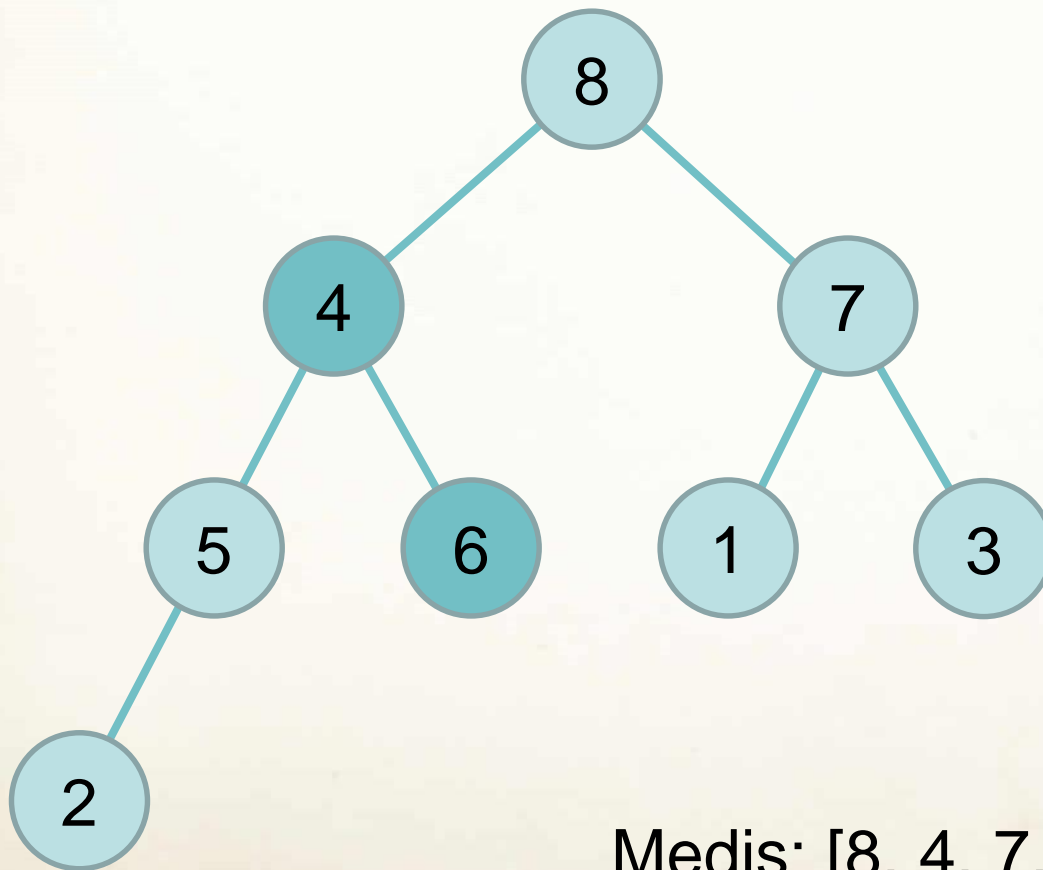


Rikiavimo eiga:  
[4, 8, 7, 5, 6, 1, 3, 2, 9].

Medis: [4, 8, 7, 5, 6, 1, 3, 2].

# *Piramidės rikiavimo pavyzdys*

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



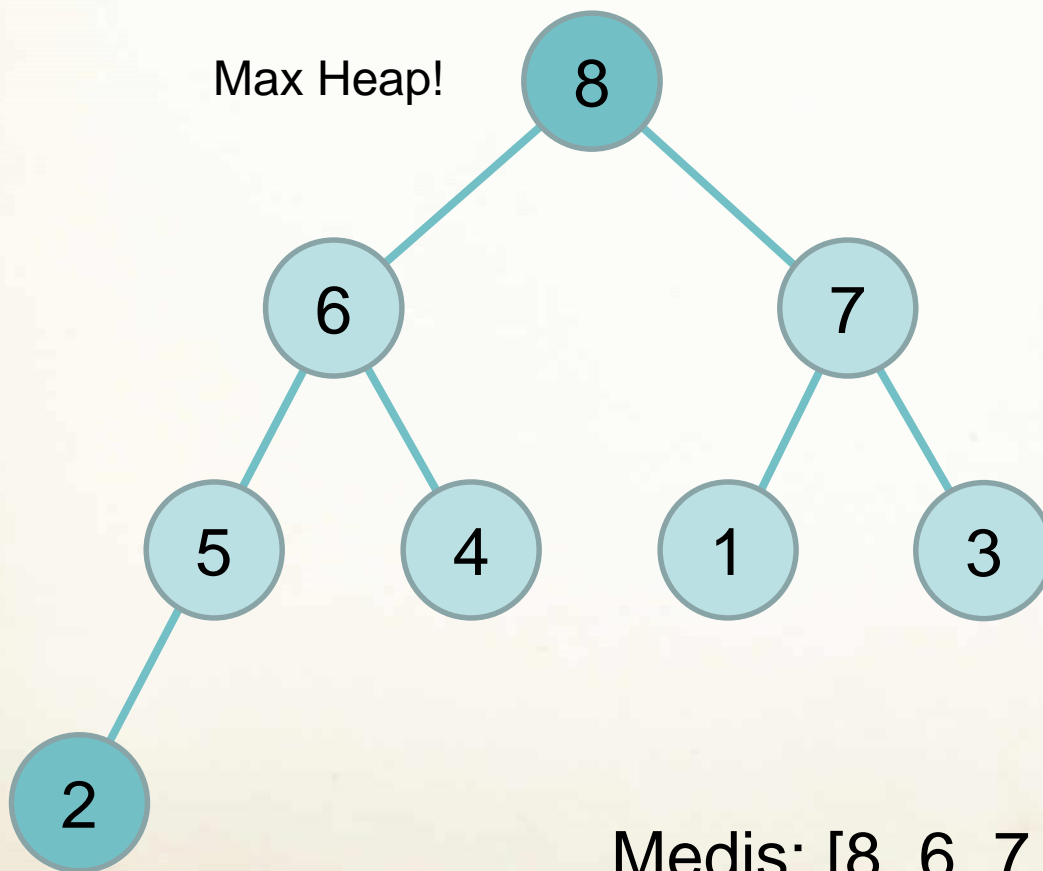
Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9].

Medis: [8, 4, 7, 5, 6, 1, 3, 2].

# *Piramidės rikiavimo pavyzdys*

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].

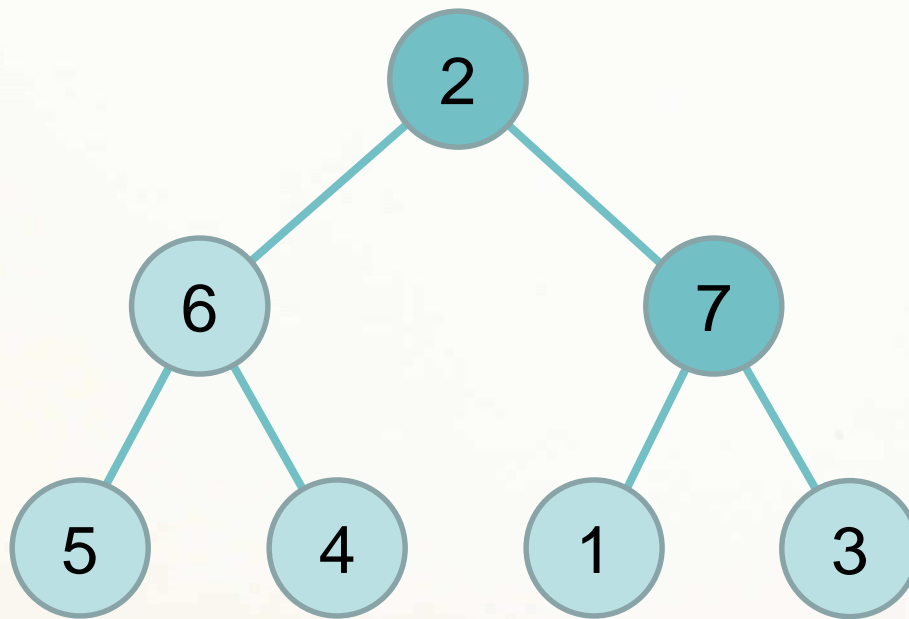


Rikiavimo eiga:  
[4, 8, 7, 5, 6, 1, 3, 2, 9].

Medis: [8, 6, 7, 5, 4, 1, 3, 2].

## *Piramidės rikiavimo pavyzdys*

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],

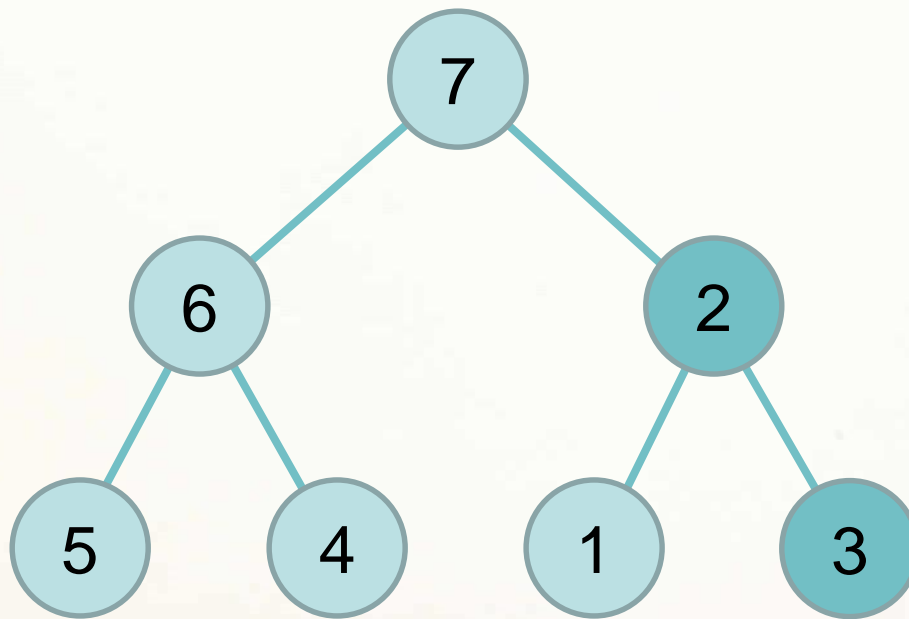
[2, 6, 7, 5, 4, 1, 3, 8, 9].

Medis: [2, 6, 7, 5, 4, 1, 3].



## *Piramidės rikiavimo pavyzdys*

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

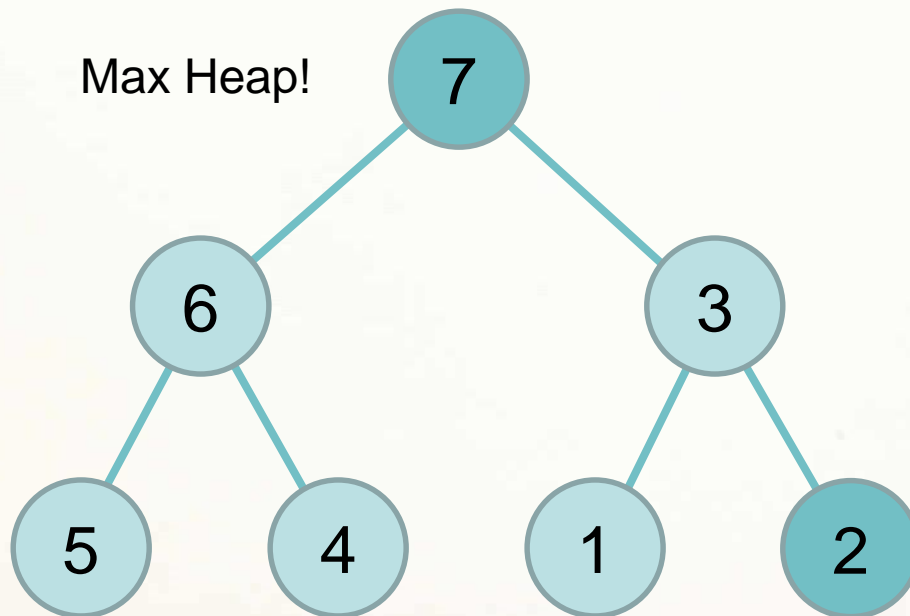
[4, 8, 7, 5, 6, 1, 3, 2, 9],

[2, 6, 7, 5, 4, 1, 3, 8, 9].

Medis: [7, 6, 2, 5, 4, 1, 3].

# Piramidės rikiavimo pavyzdys

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

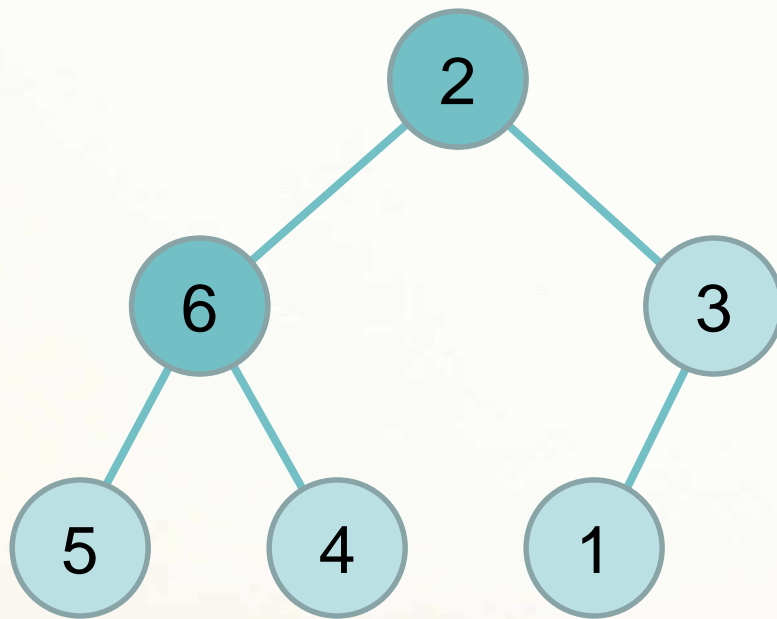
[4, 8, 7, 5, 6, 1, 3, 2, 9],

[2, 6, 7, 5, 4, 1, 3, 8, 9].

Medis: [7, 6, 3, 5, 4, 1, 2].

# Piramidės rikiavimo pavyzdys

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],

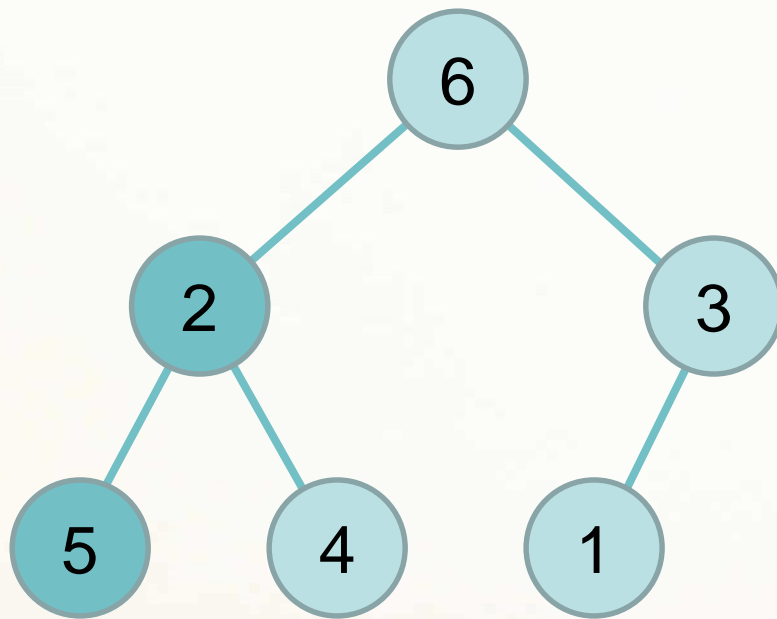
[2, 6, 7, 5, 4, 1, 3, 8, 9],

[2, 6, 3, 5, 4, 1, 7, 8, 9].

Medis: [2, 6, 3, 5, 4, 1].

# Piramidės rikiavimo pavyzdys

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],

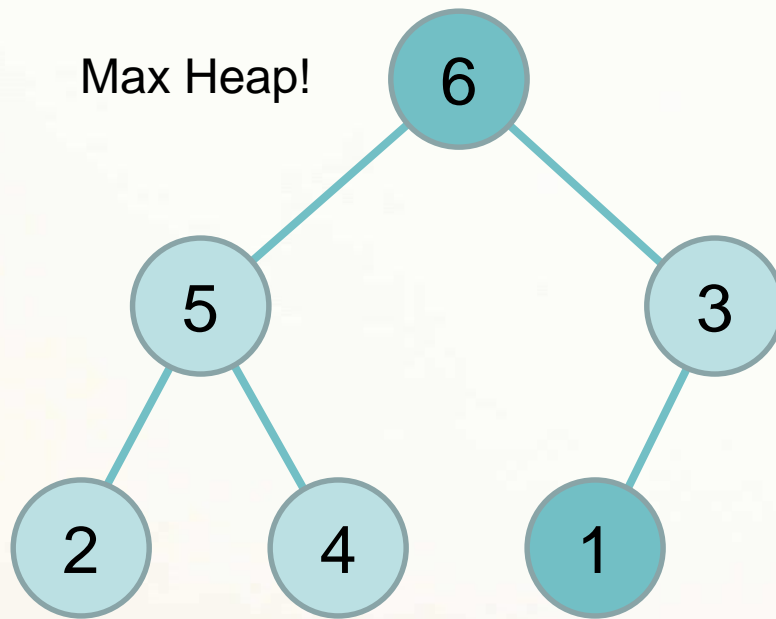
[2, 6, 7, 5, 4, 1, 3, 8, 9],

[2, 6, 3, 5, 4, 1, 7, 8, 9].

Medis: [6, 2, 3, 5, 4, 1].

# Piramidės rikiavimo pavyzdys

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],

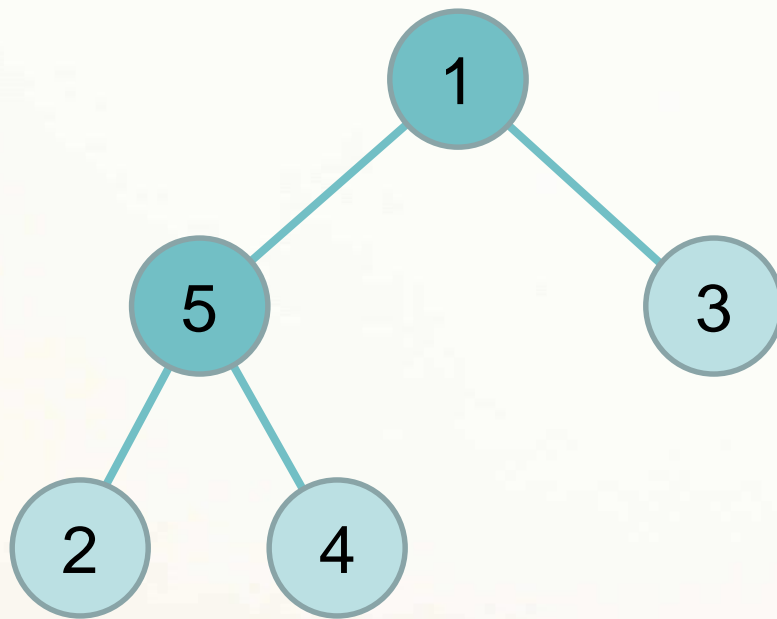
[2, 6, 7, 5, 4, 1, 3, 8, 9],

[2, 6, 3, 5, 4, 1, 7, 8, 9].

Medis: [6, 5, 3, 2, 4, 1].

# Piramidės rikiavimo pavyzdys

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],

[2, 6, 7, 5, 4, 1, 3, 8, 9],

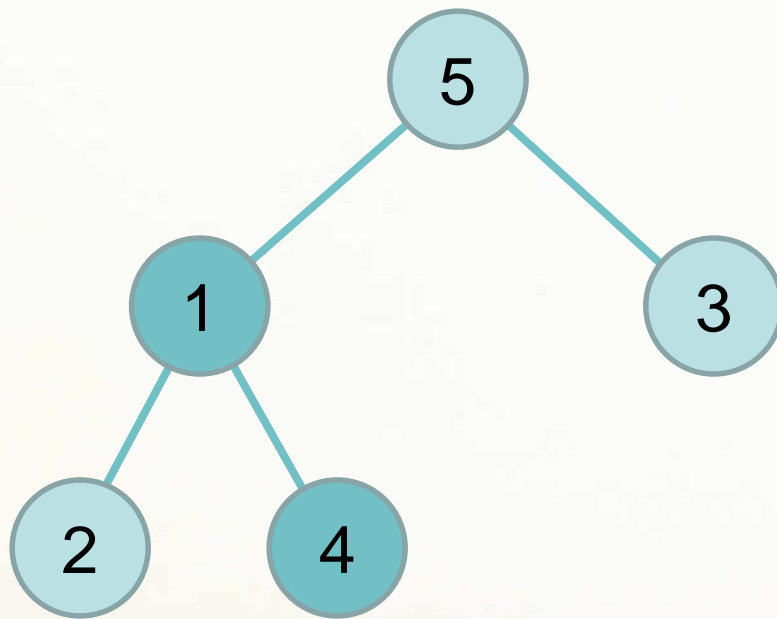
[2, 6, 3, 5, 4, 1, 7, 8, 9],

[1, 5, 3, 2, 4, 6, 7, 8, 9].

Medis: [1, 5, 3, 2, 4].

# *Piramidės rikiavimo pavyzdys*

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],

[2, 6, 7, 5, 4, 1, 3, 8, 9],

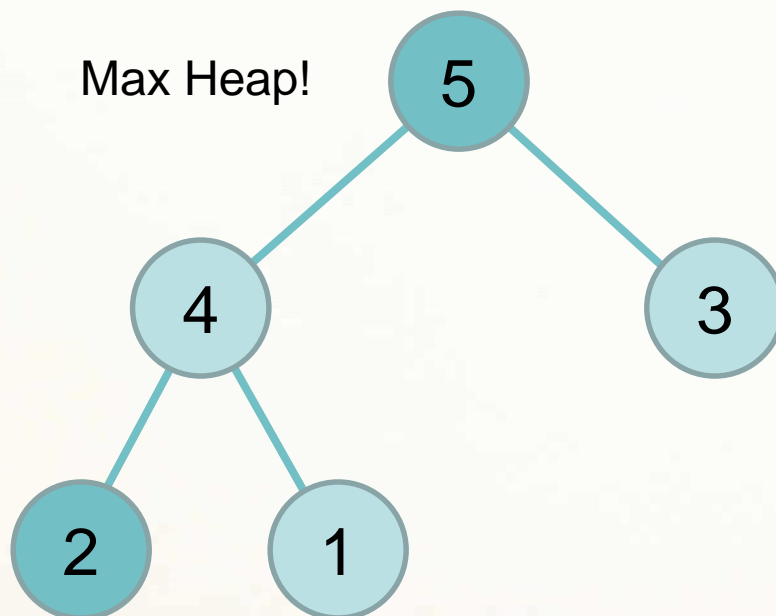
[2, 6, 3, 5, 4, 1, 7, 8, 9],

[1, 5, 3, 2, 4, 6, 7, 8, 9].

Medis: [5, 1, 3, 2, 4].

# Piramidės rikiavimo pavyzdys

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],

[2, 6, 7, 5, 4, 1, 3, 8, 9],

[2, 6, 3, 5, 4, 1, 7, 8, 9],

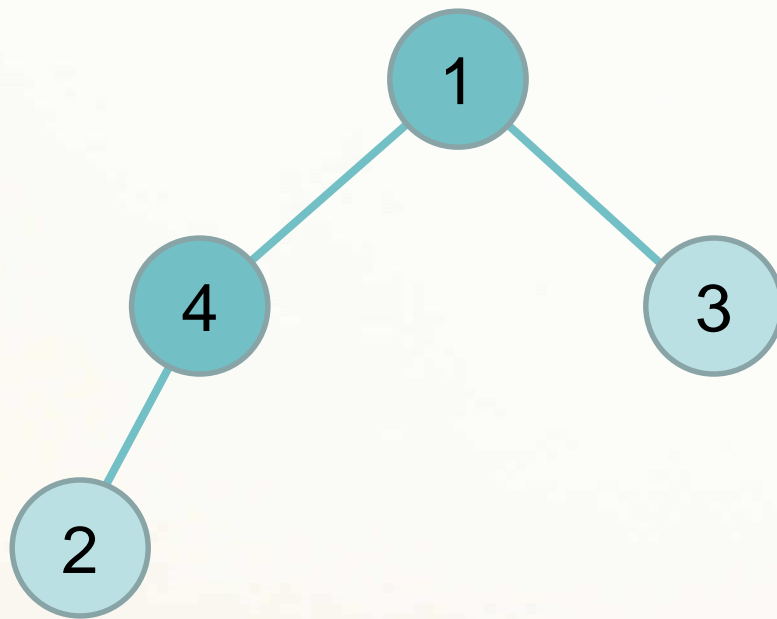
[1, 5, 3, 2, 4, 6, 7, 8, 9].

Medis: [5, 4, 3, 2, 1].



# Piramidės rikiavimo pavyzdys

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],

[2, 6, 7, 5, 4, 1, 3, 8, 9],

[2, 6, 3, 5, 4, 1, 7, 8, 9],

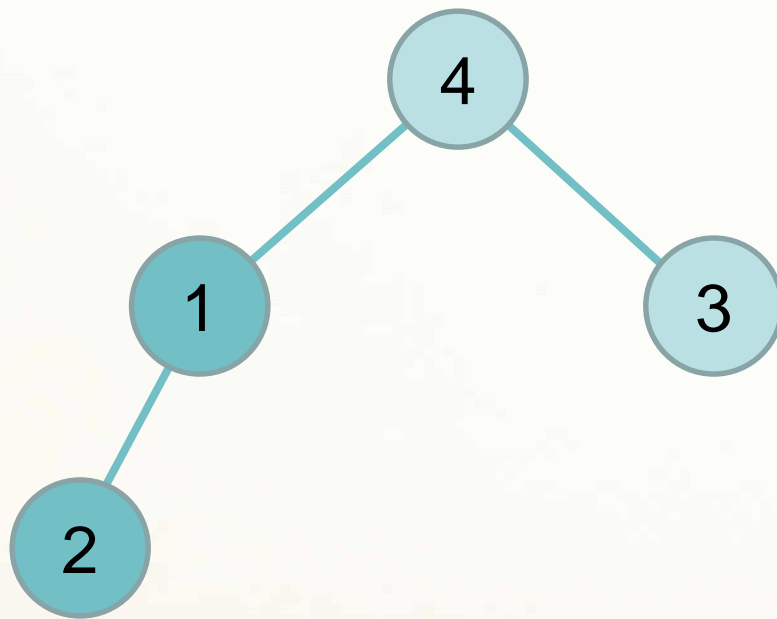
[1, 5, 3, 2, 4, 6, 7, 8, 9],

[1, 4, 3, 2, 5, 6, 7, 8, 9].

Medis: [1, 4, 3, 2].

# Piramidės rikiavimo pavyzdys

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],

[2, 6, 7, 5, 4, 1, 3, 8, 9],

[2, 6, 3, 5, 4, 1, 7, 8, 9],

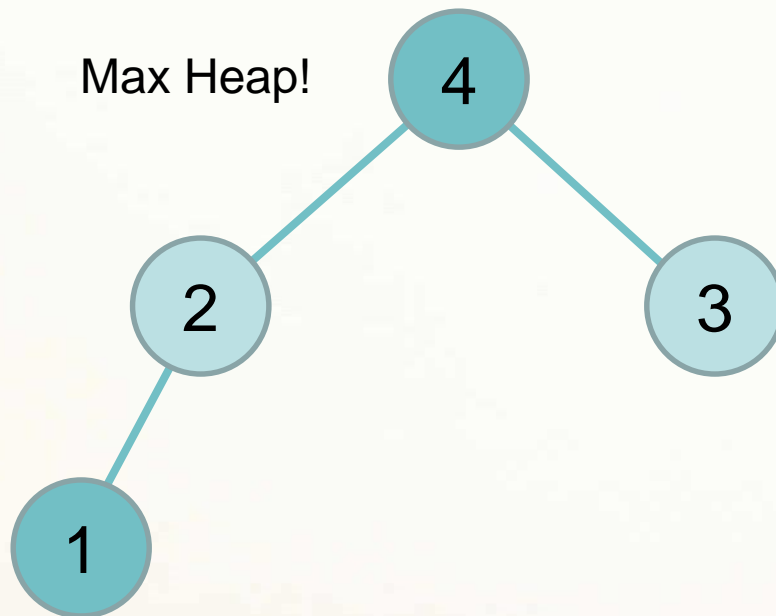
[1, 5, 3, 2, 4, 6, 7, 8, 9],

[1, 4, 3, 2, 5, 6, 7, 8, 9].

Medis: [4, 1, 3, 2].

# Piramidės rikiavimo pavyzdys

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],

[2, 6, 7, 5, 4, 1, 3, 8, 9],

[2, 6, 3, 5, 4, 1, 7, 8, 9],

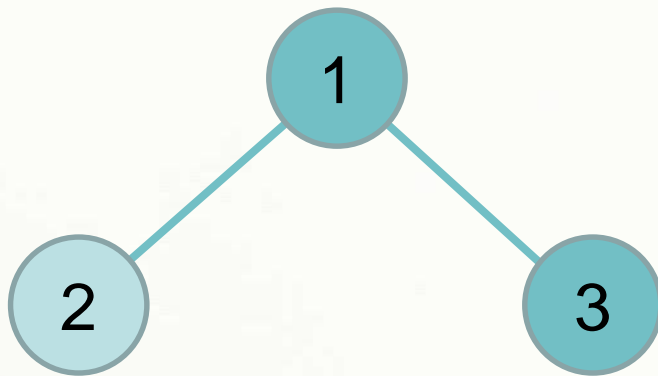
[1, 5, 3, 2, 4, 6, 7, 8, 9],

[1, 4, 3, 2, 5, 6, 7, 8, 9].

Medis: [4, 2, 3, 1].

# *Piramidės rikiavimo pavyzdys*

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],

[2, 6, 7, 5, 4, 1, 3, 8, 9],

[2, 6, 3, 5, 4, 1, 7, 8, 9],

[1, 5, 3, 2, 4, 6, 7, 8, 9],

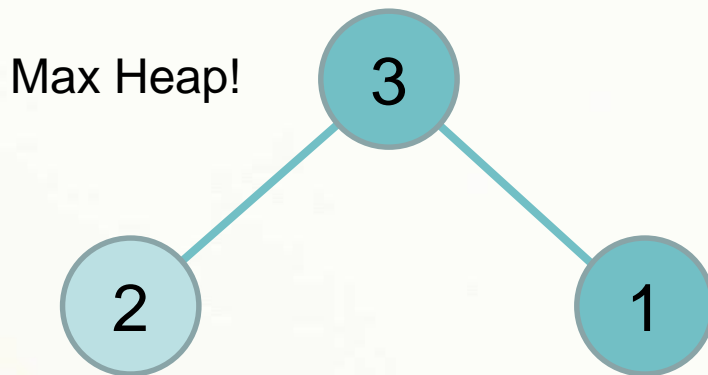
[1, 4, 3, 2, 5, 6, 7, 8, 9],

[1, 2, 3, 4, 5, 6, 7, 8, 9].

Medis: [1, 2, 3].

# Piramidės rikiavimo pavyzdys

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



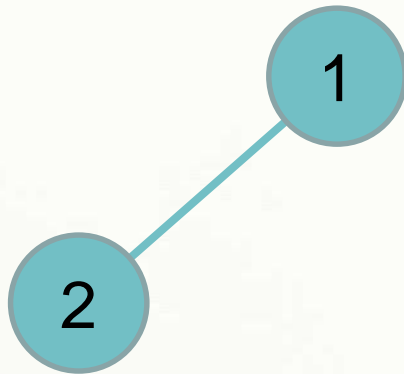
Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],  
[2, 6, 7, 5, 4, 1, 3, 8, 9],  
[2, 6, 3, 5, 4, 1, 7, 8, 9],  
[1, 5, 3, 2, 4, 6, 7, 8, 9],  
[1, 4, 3, 2, 5, 6, 7, 8, 9],  
[1, 2, 3, 4, 5, 6, 7, 8, 9].

Medis: [3, 2, 1].

# *Piramidės rikiavimo pavyzdys*

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



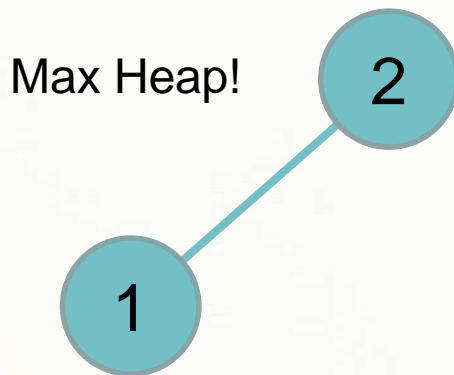
Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],  
[2, 6, 7, 5, 4, 1, 3, 8, 9],  
[2, 6, 3, 5, 4, 1, 7, 8, 9],  
[1, 5, 3, 2, 4, 6, 7, 8, 9],  
[1, 4, 3, 2, 5, 6, 7, 8, 9],  
[1, 2, 3, 4, 5, 6, 7, 8, 9],  
[1, 2, 3, 4, 5, 6, 7, 8, 9].

Medis: [1, 2].

# Piramidės rikiavimo pavyzdys

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].



Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],  
[2, 6, 7, 5, 4, 1, 3, 8, 9],  
[2, 6, 3, 5, 4, 1, 7, 8, 9],  
[1, 5, 3, 2, 4, 6, 7, 8, 9],  
[1, 4, 3, 2, 5, 6, 7, 8, 9],  
[1, 2, 3, 4, 5, 6, 7, 8, 9],  
[1, 2, 3, 4, 5, 6, 7, 8, 9].

Medis: [2, 1].

# *Piramidės rikiavimo pavyzdys*

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].

1

Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],

[2, 6, 7, 5, 4, 1, 3, 8, 9],

[2, 6, 3, 5, 4, 1, 7, 8, 9],

[1, 5, 3, 2, 4, 6, 7, 8, 9],

[1, 4, 3, 2, 5, 6, 7, 8, 9],

[1, 2, 3, 4, 5, 6, 7, 8, 9],

[1, 2, 3, 4, 5, 6, 7, 8, 9],

[1, 2, 3, 4, 5, 6, 7, 8, 9].

Medis: [1].



# *Piramidės rikiavimo pavyzdys*

Išrikiuokime sąrašą: [5, 9, 1, 4, 6, 7, 3, 2, 8].

Rikiavimo eiga:

[4, 8, 7, 5, 6, 1, 3, 2, 9],

[2, 6, 7, 5, 4, 1, 3, 8, 9],

[2, 6, 3, 5, 4, 1, 7, 8, 9],

[1, 5, 3, 2, 4, 6, 7, 8, 9],

[1, 4, 3, 2, 5, 6, 7, 8, 9],

[1, 2, 3, 4, 5, 6, 7, 8, 9],

[1, 2, 3, 4, 5, 6, 7, 8, 9],

[1, 2, 3, 4, 5, 6, 7, 8, 9],

[1, 2, 3, 4, 5, 6, 7, 8, 9].

# **Skaitmeninis rikiavimas (angl. Radix sort)**

Šis rikiavimas skirtas sudėtingiems duomenims:

- įrašams duomenų bazėse,
- telefonų sąrašams,
- bibliotekų katalogams ir t. t.

Skaitmeninio rikiavimo algoritmuose duomenų reikšmės interpretuojamos kaip skaičiai  $m$ -tainėje skaičiavimo sistemoje.

Pavyzdys: išrikiuokime septyntainės skaičiavimo sistemos skaičius: 1125, 1043, 112, 34, 222, 6, 662, 51, 2040, 1000, 513, 101, 103, 5, 43, 11.

# Skaitmeninio rikiavimo pavyzdys

1125, 1043, 112, 34, 222, 6, 662, 51, 2040, 1000, 513, 101, 103, 5, 43, 11.

0: 2040, 1000

1: 51, 101, 11

2: 112, 222, 662

3: 1043, 513, 103, 43

4: 34

5: 1125, 5

6: 6

Sąrašas po pirmos skaitmeninio rikiavimo iteracijos:

2040, 1000, 51, 101, 11, 112, 222, 662, 1043, 513, 103, 43, 34, 1125, 5, 6.

# Skaitmeninio rikiavimo pavyzdys

2040, 1000, 51, 101, 11, 112, 222, 662, 1043, 513, 103, 43, 34, 1125, 5, 6.

0: 1000, 101, 103, 05, 06

1: 11, 112, 513

2: 222, 1125

3: 34

4: 2040, 1043, 43

5: 51

6: 662

Sąrašas po antros skaitmeninio rikiavimo iteracijos:

1000, 101, 103, 5, 6, 11, 112, 513, 222, 1125, 34, 2040, 1043, 43, 51, 662.

# Skaitmeninio rikiavimo pavyzdys

1000, 101, 103, 5, 6, 11, 112, 513, 222, 1125, 34, 2040, 1043, 43, 51, 662.

0: 1000, 005, 006, 011, 034, 2040, 1043, 043, 051

1: 101, 103, 112, 1125

2: 222

3:

4:

5: 513

6: 662

Sąrašas po trečios skaitmeninio rikiavimo iteracijos:

1000, 5, 6, 11, 34, 2040, 1043, 43, 51, 101, 103, 112, 1125, 222, 513, 662.

# Skaitmeninio rikiavimo pavyzdys

1000, 5, 6, 11, 34, 2040, 1043, 43, 51, 101, 103, 112, 1125, 222, 513, 662.

0: 0005, 0006, 0011, 0034, 0043, 0051,

0101, 0103, 0112, 0222, 0513, 0662

1: 1000, 1043, 1125

2: 2040

3:

4:

5:

6:

Sąrašas po ketvirtos skaitmeninio rikiavimo iteracijos:

5, 6, 11, 34, 43, 51, 101, 103, 112, 222, 513, 662, 1000, 1043, 1125, 2040.

# Rikiavimo algoritmų sudėtingumai

Algoritmas	Sudėtingumas blogiausių atveju	Vidutinis sudėtingumas	Sudėtingumas geriausių atveju
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Bubble sort	$O(n^2)$	$O(n^2)$	* $O(n)$
Insertion sort	$O(n^2)$	$O(n^2)$	$O(n)$
Shell sort	$\sim O(n \log n)$	$\sim O(n^{1,2})$	$O(n \log^2 n)$
Quick sort	$O(n^2)$	$O(n \log n)$	$O(n \log n)$
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
External sorting	$\sim O(n \log n)$	$\sim O(n \log n)$	$\sim O(n \log n)$
Heap sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Radix sort	$O(k*n)$	$O(k*n)$	$O(k*n)$

# ***Rikiavimo algoritmų greitaveikos palyginimas***

Eksperimentiniu būdu gauti algoritmų sudėtingumai:

- *Quick sort*:  $Q(n) = 11.667 (n+1) \ln(n) - 1.74n - 18.74$ ,
- *Merge sort*:  $M(n) = 12.5 n \ln(n)$ ,
- *Heap sort*:  $H(n) = 16n \ln(n) + 0.01n$ ,
- *Insertion sort*:  $I(n) = 2.25n^2 + 7.75n - 3\ln(n)$ .

n atitinka įvesties duomenų apimtį,

$Q(n)$ ,  $M(n)$ ,  $H(n)$ ,  $I(n)$  – atliekamų operacijų skaičių.

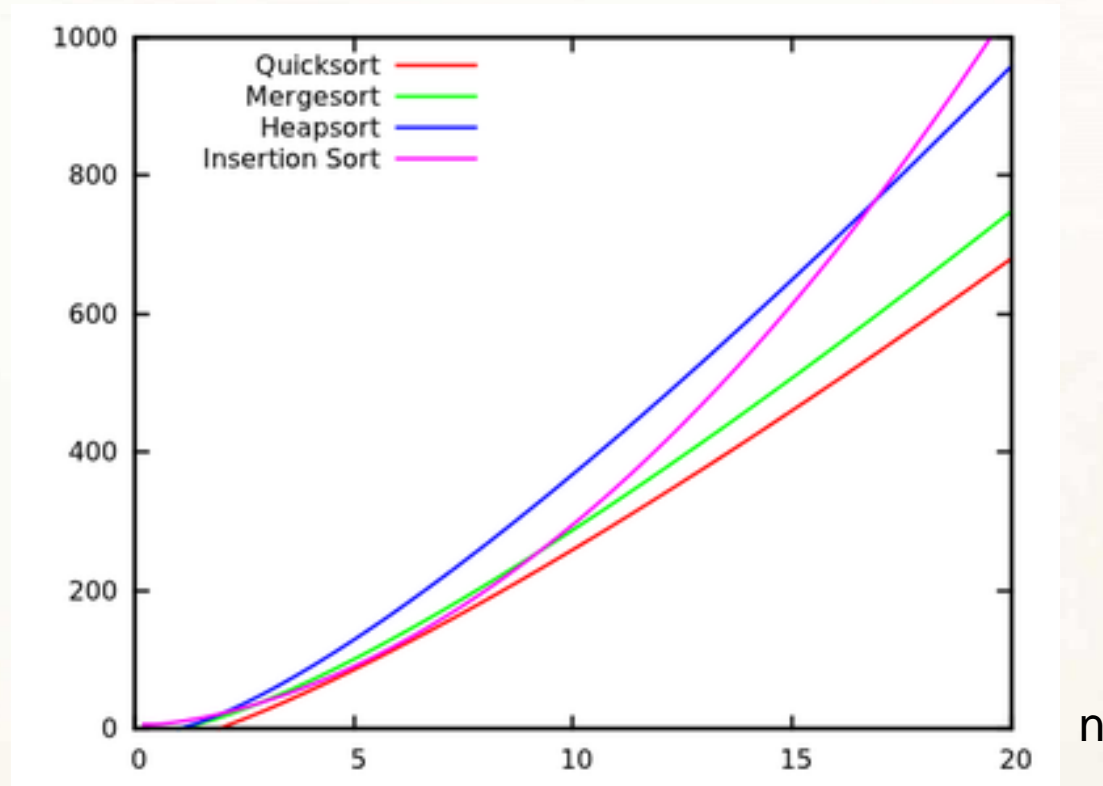
**Šaltinis:** Don Knuth's book series "The Art of Computer Programming".

<https://www-cs-faculty.stanford.edu/~knuth/taocp.html>



# Rikiavimo algoritmų palyginimas (1)

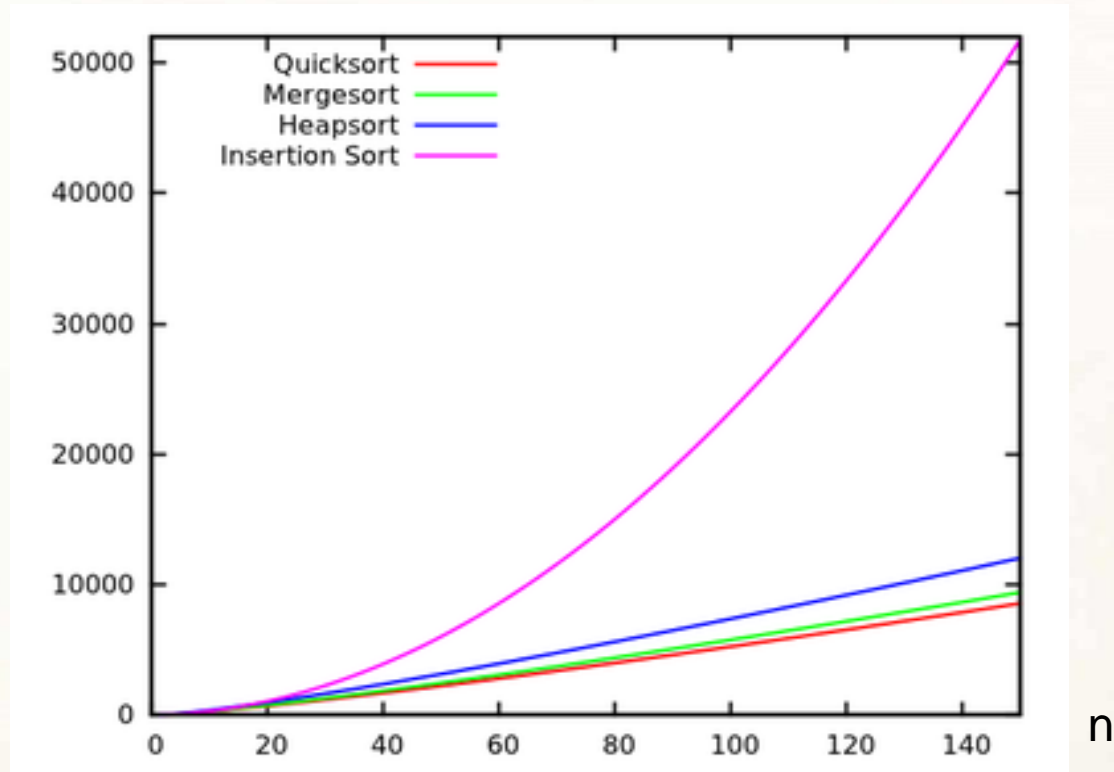
$f(n)$



Šaltinis: <https://cs.stackexchange.com/questions/3/why-is-quicksort-better-than-other-sorting-algorithms-in-practice>

# Rikiavimo algoritmų palyginimas (2)

$f(n)$



Šaltinis: <https://cs.stackexchange.com/questions/3/why-is-quicksort-better-than-other-sorting-algorithms-in-practice>

***Ačiū už dėmesį.***

***Klausimai?***