Laborants Dāvis Kalvāns

02.04.2022

Paralēlā skaitļošana

Latvijas Zinātnes padome

FLPP
FUNDAMENTĀLO UN LIETIŠĶO PĒTĪJUMU PROJEKTI

Saturs

1. Kas ir paralēlā skaitļošana?
2. Mazs piemērs no dzīves.
3. Motivācija paralēlai skaitļošanai - datora procesora uzbūve un Mūra likums.
4. Apkaunojoši jeb patīkami paralelizējamas problēmas.
5. Laukums zem līknes.
6. Pirmskaitļu meklēšana.

Kādas asociācijas nāk prātā dzirdot vārdus paralēlā skaitļošana?

Paralēlās skaitļošanas definīcija

Paralēlā skaitļošana ir aprēķināšanas veids, kad datoram tiek uzdots uzdevumus, kurā tas veic vairākus aprēķinus vienlaicīgi. [1]

[1] Gottlieb, Allan; Almasi, George S. (1989). Highly parallel computing. Redwood City, Calif.: Benjamin/Cummings. ISBN 978-0-8053-0177-9.

Piemērs no dzīves– Anniņa, Jānītis, Zane un Toms izdomā paātrināt mājasdarba izpildi

a) $4x^{\frac{1}{2}} \cdot (x-4)$, ja $x = 16$

b) $a^{\frac{1}{4}} + 3a^{\frac{1}{2}}$, ja $a = 64$

c) $y^{\frac{1}{5}} - 2y^{\frac{1}{2}} + 5$, ja $y = 32$

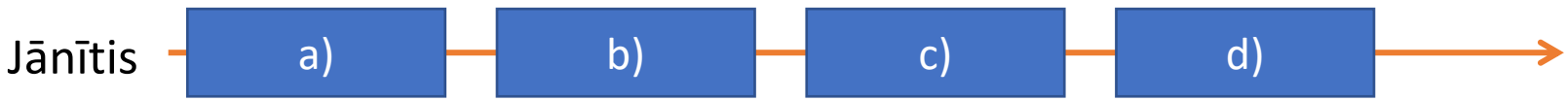
d) $a^{\frac{1}{3}} - 2a^{\frac{2}{3}} - 1$, ja $a = -8$

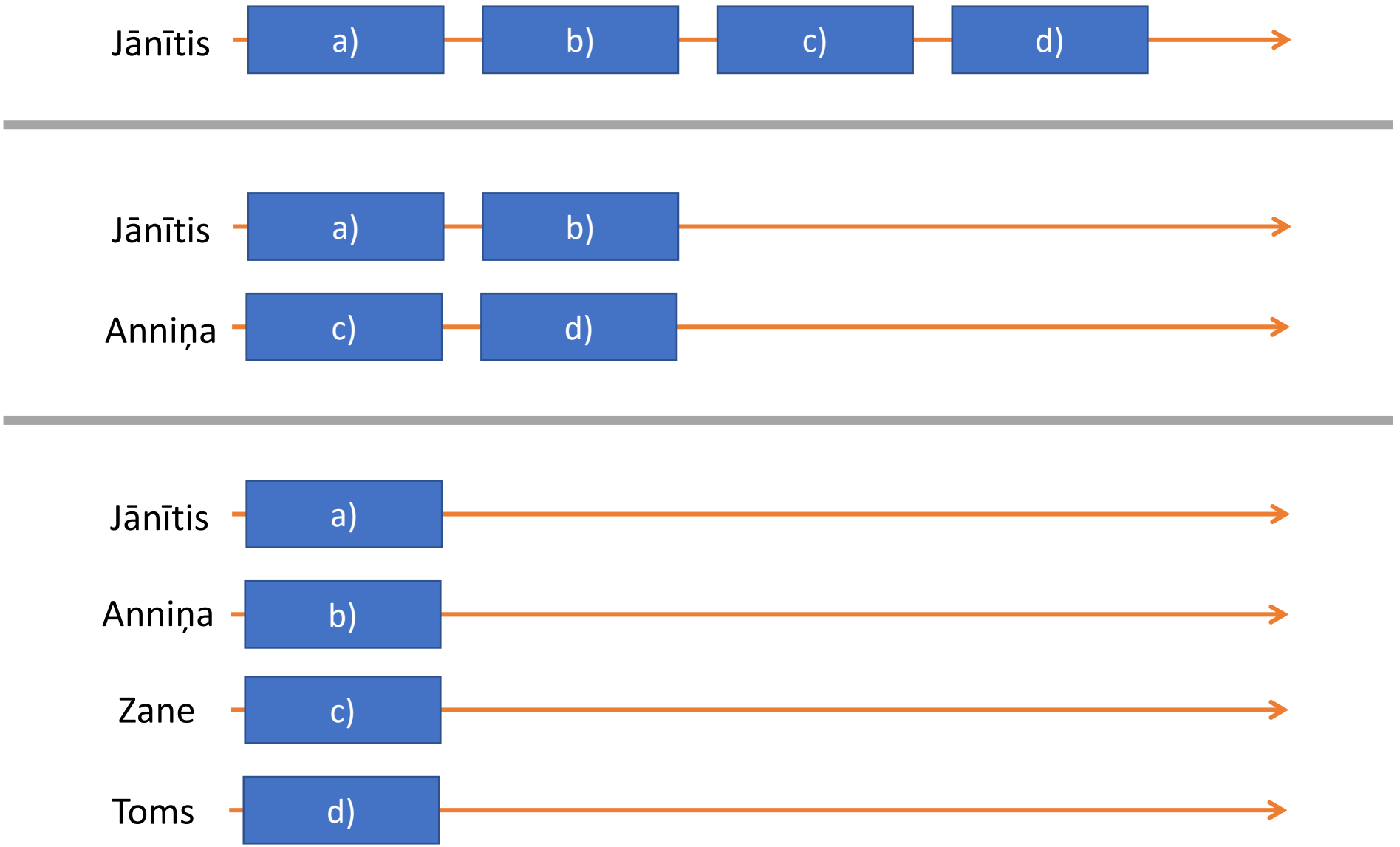
Kā notika paralelizācija (kādu algoritmu izmantoja)

1. Četri klases biedri sadalīja darbus (katrs pieteicās izpildīt vienu no piemēriem).
2. Katrs izpildīja savu piemēru.
3. Dalījās ar savu atbildi ar pārējiem.

Grafiski attēlosim izpildes laiku

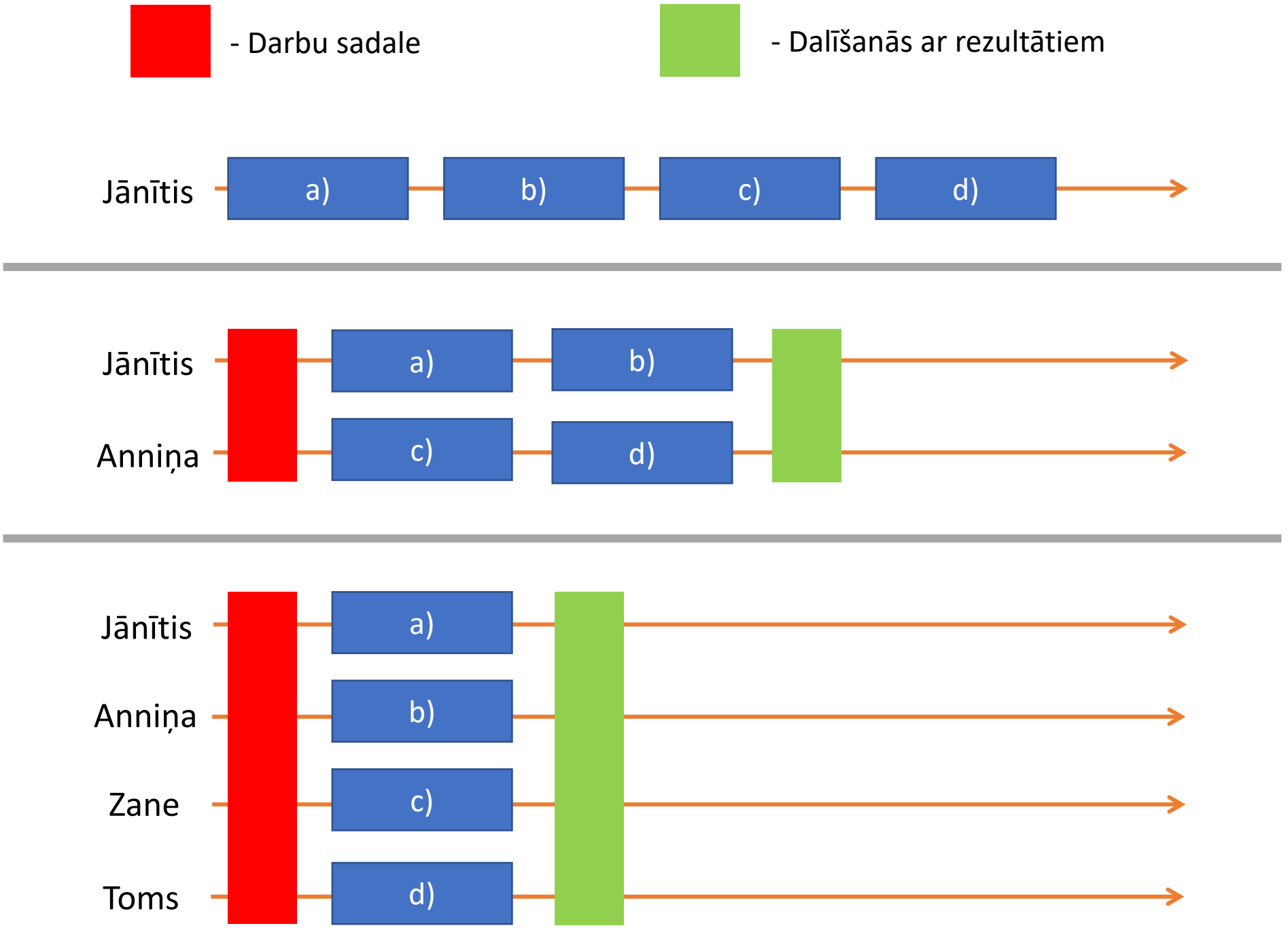
- Vienkāršības pēc, pieņemam, ka katram piemēram vajadzīgs vienāds izpildes laiks.
- Grafiski attēlojam uz taisnes, ar taisnstūriem norādot veikto darbību.
- Piemēram, šeit attēlots izpildes laiks Jānītim, ja viņš visu uzdevumu risinātu viens pats.




Jānītis
a)
b)
c)
d)
Jānītis
a)
b)
Anniņa
c)
d)
Jānītis
a)
Anniņa
b)
Zane
c)
Toms
d)

Vai es neesmu kaut ko piemirsis?

Darba sadale un dalīšanās ar rezultātiem arī prasa laiku

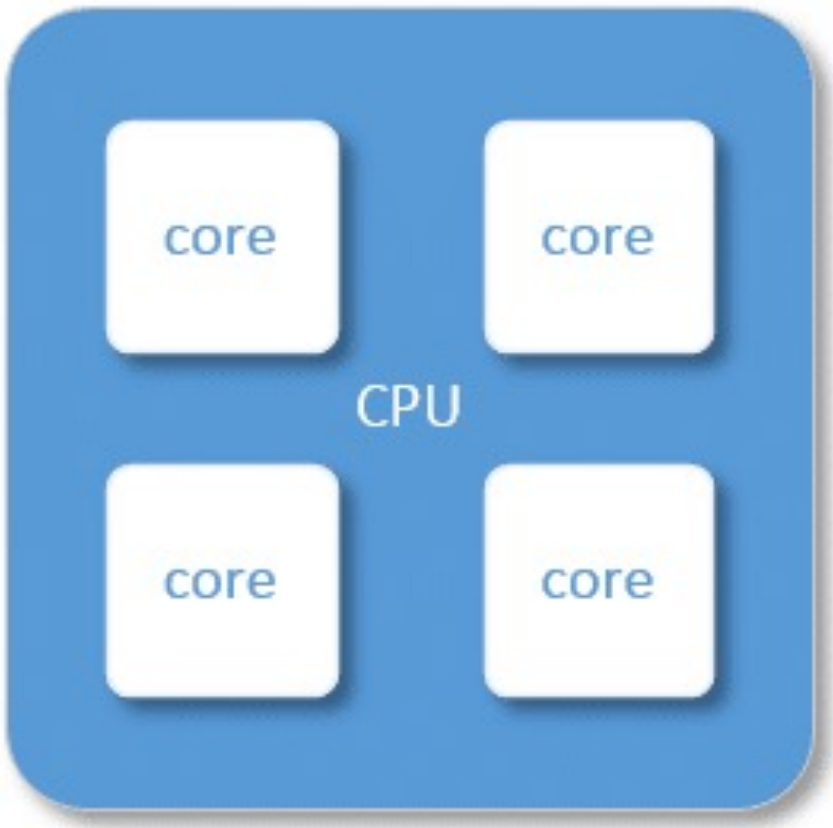

- Darbu sadale
- Dalīšanās ar rezultātiem
Jānītis
a)
b)
c)
d)
Jānītis
Anniņa
a)
b)
c)
d)
Jānītis
Anniņa
Zane
Toms
a)
b)
c)
d)

Kā piemērs saistīts ar datora aprēķiniem?

- Izklausās, ka šāda paralēla skaitļošana strādātu tikai ar vairākiem datoriem.
- Tomēr tā tas nav.

Datora procesors (CPU – central processing unit)

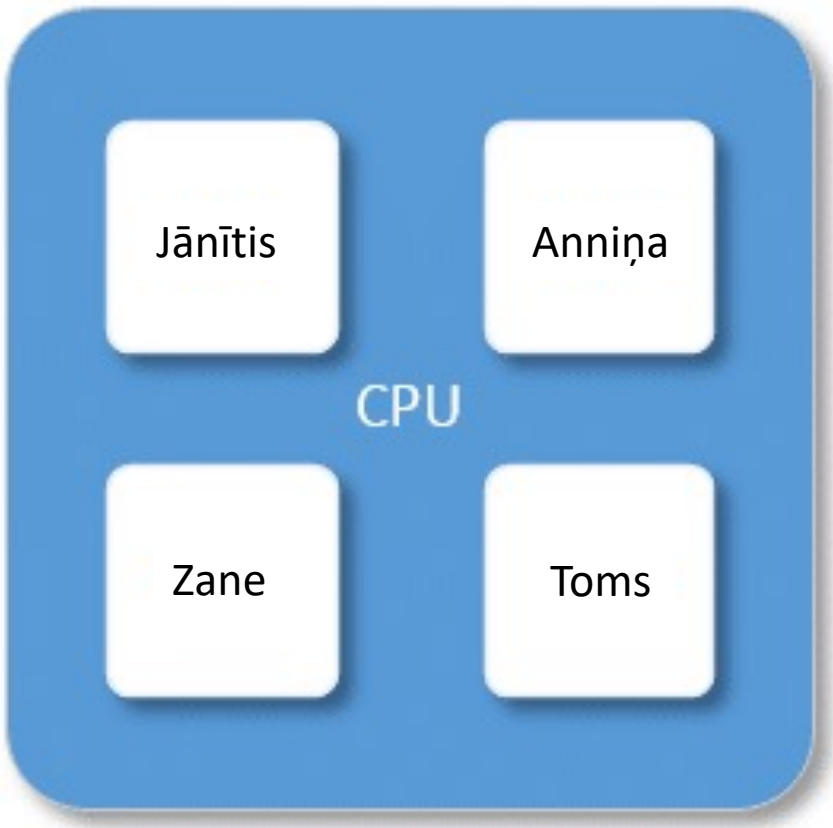
- Mūsdienās datoru procesori sastāv no vairākiem kodoliem, kas katrs var veikt neatkarīgas darbības/aprēķinus.
- Bildē procesors ar 4 kodoliem.



Quad-core CPU

Datora procesors (CPU – central processing unit)

- Mūsdienās datoru procesori sastāv no vairākiem kodoliem, kas katrs var veikt neatkarīgas darbības/aprēķinus.
- Bildē procesors ar 4 kodoliem.



Quad-core CPU

Mūra likums

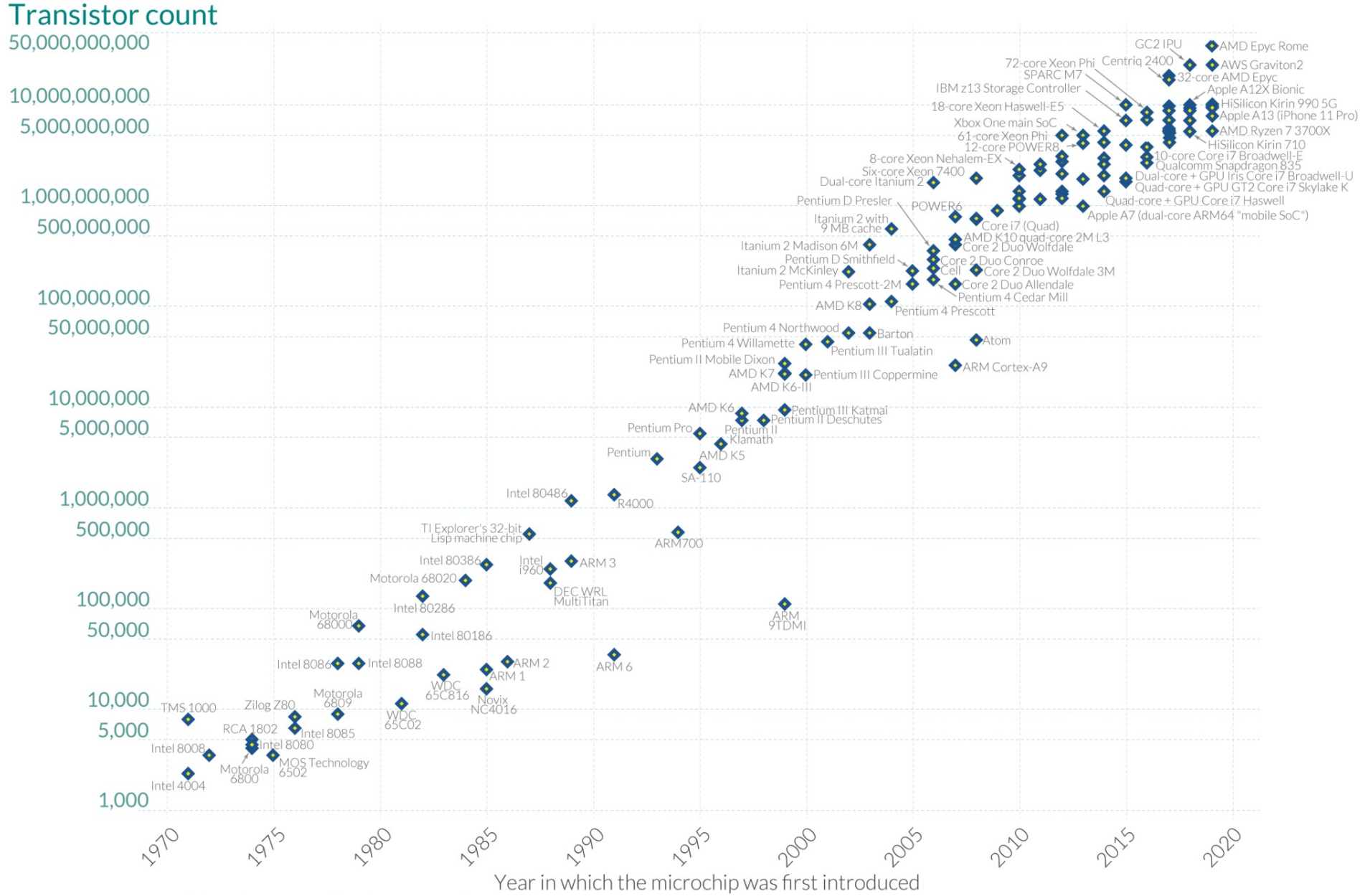
Mūra likums ir novērojums, ka skaitļošanas aparatūras vēstures gaitā tranzistoru skaits integrālajās shēmās dubultojas aptuveni divu gadu laikā. Likums nosaukts Intel līdzdibinātāja **Gordona Mūra vārdā**, kurš aprakstīja šo tendenci savā 1965. gada publikācijā. [2]

[2] Gordon E. Moore. «Cramming more components onto integrated circuits». Electronics Magazine, 1965

Moore's Law: The number of transistors on microchips has doubled every two years

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

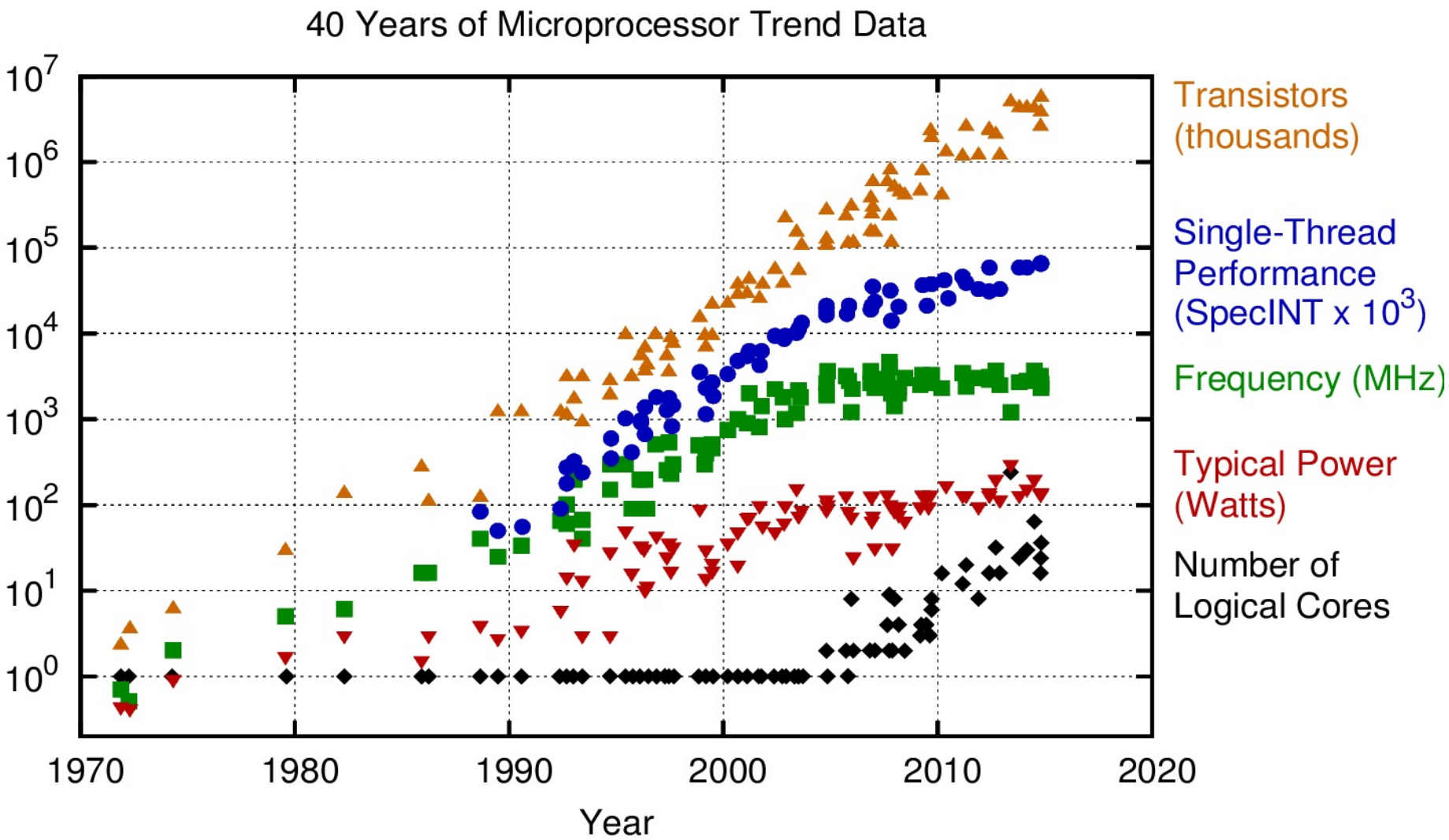
Our World in Data



https://ourworldindata.org/technological-progress

Bet kāpēc paralēli uz vairākiem kodoliem?

Izveidojam «super procesoru» ar vienu kodolu un nekādus paralēlus algoritmus nevajadzēs, vai ne?



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/

Apkaunojoši jeb patīkami paralelizējamas problēmas.

Paralēlā skaitļošanā apkaunojoši paralelizējama problēma (arī saukta par patīkami paralelizējamu problēmu) ir tāda, kura bez vai ar minimālām grūtībām ir sadalāma vairākos paralēlos uzdevumos. [3]

[3] Herlihy, Maurice; Shavit, Nir (2012). The Art of Multiprocessor Programming, Revised Reprint (revised ed.). Elsevier. p. 14. ISBN 9780123977953.

Nav pārāk veiksmīga definīcija

Paralēlā skaitļošanā apkaunojoši paralelizējama problēma (arī saukta par patīkami paralelizējamu problēmu) ir tāda, kura bez vai ar minimālām grūtībām ir sadalāma vairākos paralēlos uzdevumos. [3]

[3] Herlihy, Maurice; Shavit, Nir (2012). The Art of Multiprocessor Programming, Revised Reprint (revised ed.). Elsevier. p. 14. ISBN 9780123977953.

Piedāvāju savu definīciju

Paralēlā skaitļošanā apkaunojoši paralelizējams algoritms (arī saukts par patīkami paralelizējamu algoritmu) ir tāds, kuram sekvenciālā versija ir sadalāma vairākos neatkarīgos paralēlos uzdevumos.

Iepriekšējais piemērs ir patīkami paralelizējams

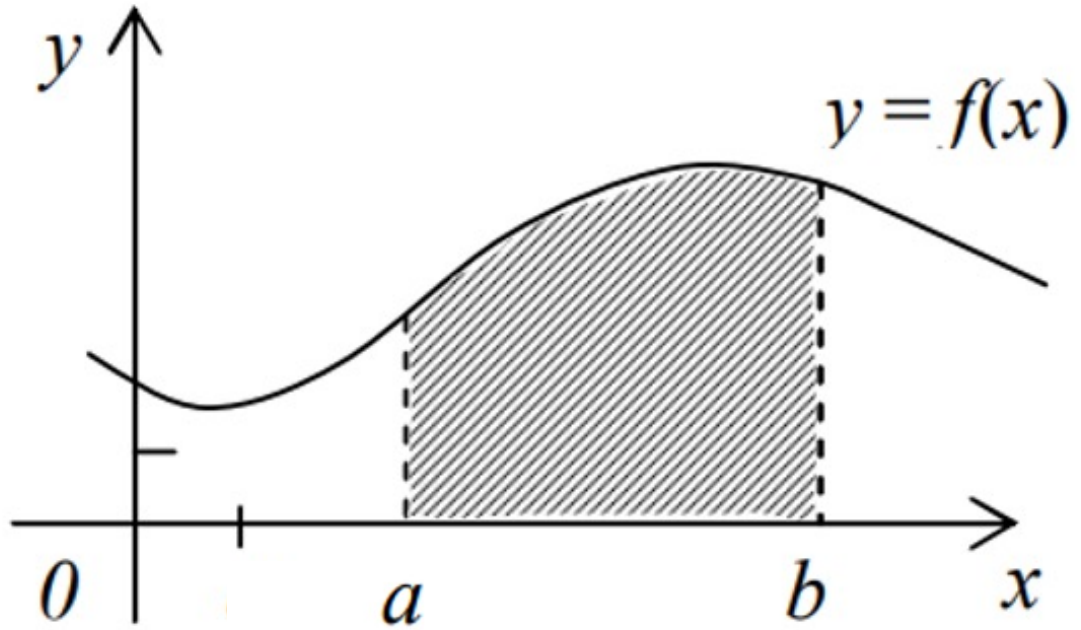
a) $4x^{\frac{1}{2}} \cdot (x-4)$, ja $x = 16$

b) $a^{\frac{1}{4}} + 3a^{\frac{1}{2}}$, ja $a = 64$

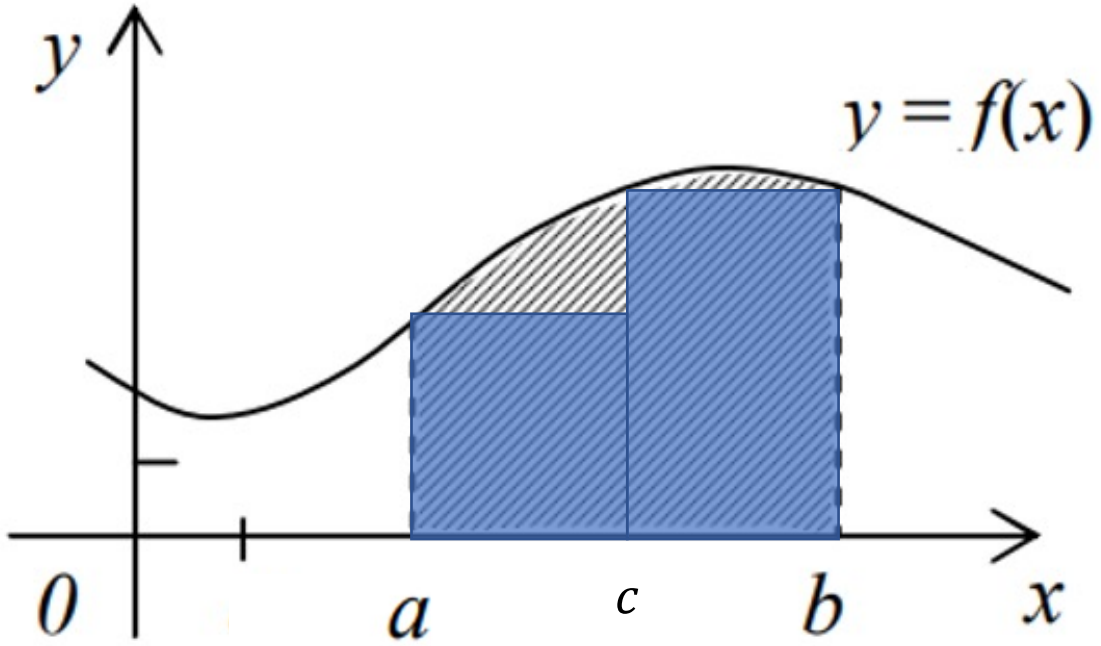
c) $y^{\frac{1}{5}} - 2y^{\frac{1}{2}} + 5$, ja $y = 32$

d) $a^{\frac{1}{3}} - 2a^{\frac{2}{3}} - 1$, ja $a = -8$

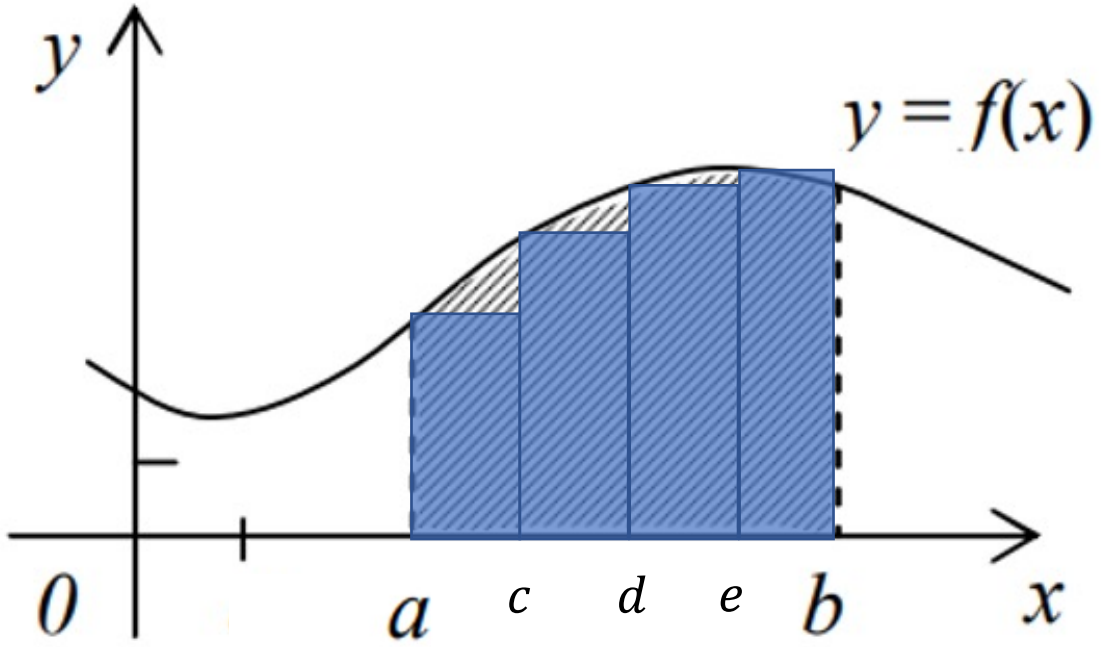
Cits piemērs – laukums zem līknes



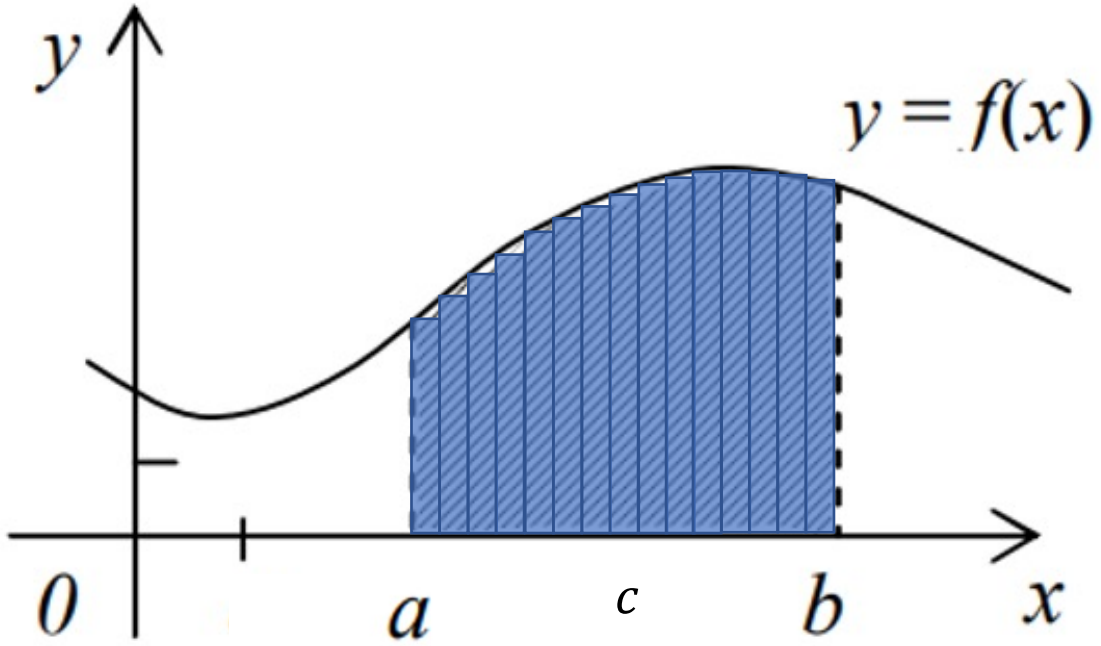
Sadalām intervālu uz diviem



Sadalām vēl smalkāk



Vēl smalkāk



Ar šādu ideju var aprēķināt laukum zem līknes

- Sadala bezgalīgi daudz starppunktos (jeb attālums starp tiem tiecās uz nulli).
- Un tad saskaita kopā laukumus.
- To sauc par noteikto integrāli un apzīmē ar:

$$\int_a^b f(x)dx$$

Datora aptuveno aprēķinu algoritms

- Sadala intervālu $[a, b]$ n starppunktos ($a = x_0, x_1, x_2, \dots, x_n, x_{n+1} = b$).
- Aprēķina funkcijas vērtības šajos punktos un sareizina ar attālumu līdz nākamajam punktam (platumu h).
- Sasummē kopā:

$$\int_a^b f(x) \approx hf(x_0) + hf(x_1) + \dots + hf(x_n) =$$

$$= h(f(x_0) + f(x_1) + \dots + f(x_n) =$$

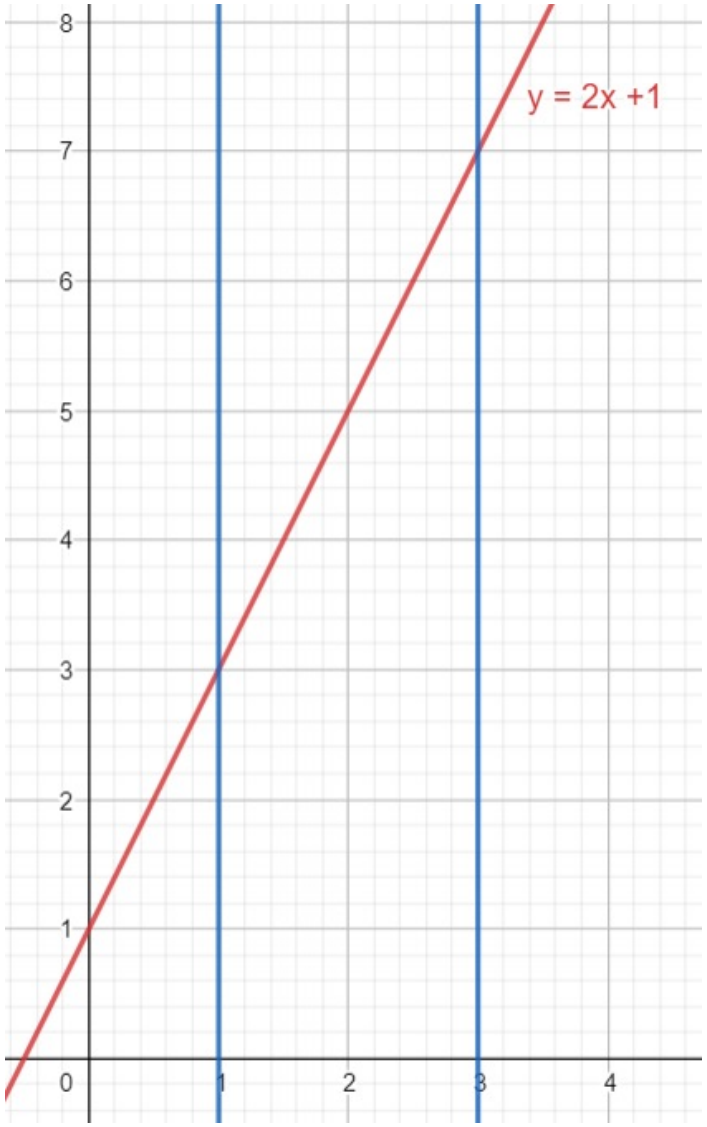
$$= h\sum_{i=0}^{n} f(x_i)$$

Pamēģiniet vienkāršai funkcijai (bez šī paņēmiena)

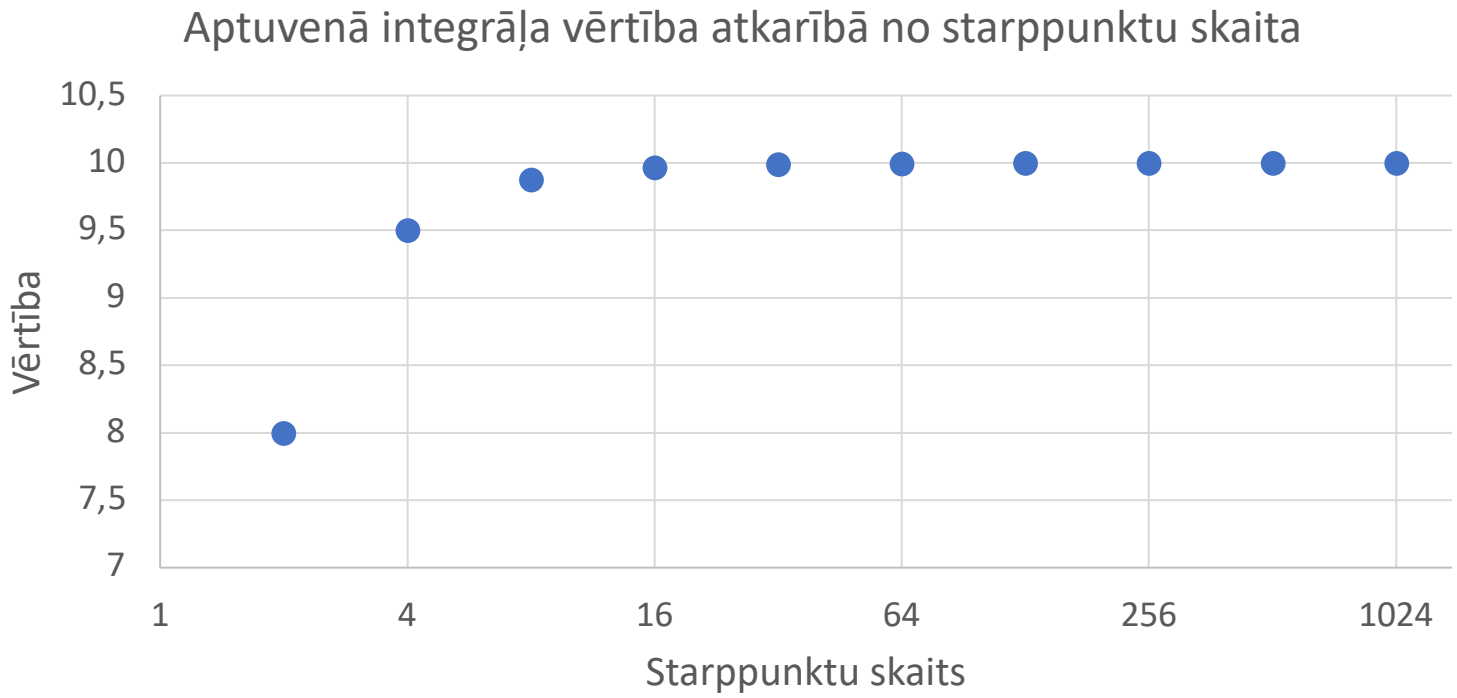
$$f(x) = 2x + 1, \qquad \int_1^3 f(x)dx = ?$$

Funkcijas $f(x) = 2x + 1$ grafiks

Varam aprēķināt trapeces laukumu un iegūt $5 \cdot 2 = 10$



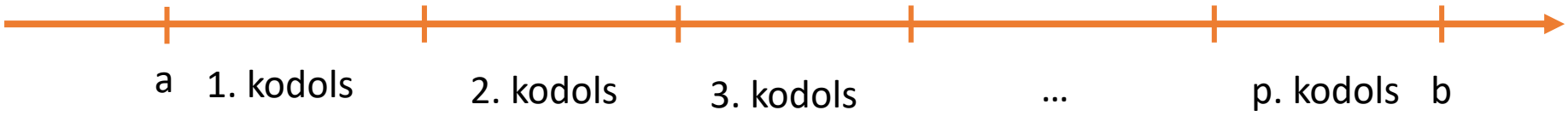
Kādu rezultātu dod dators



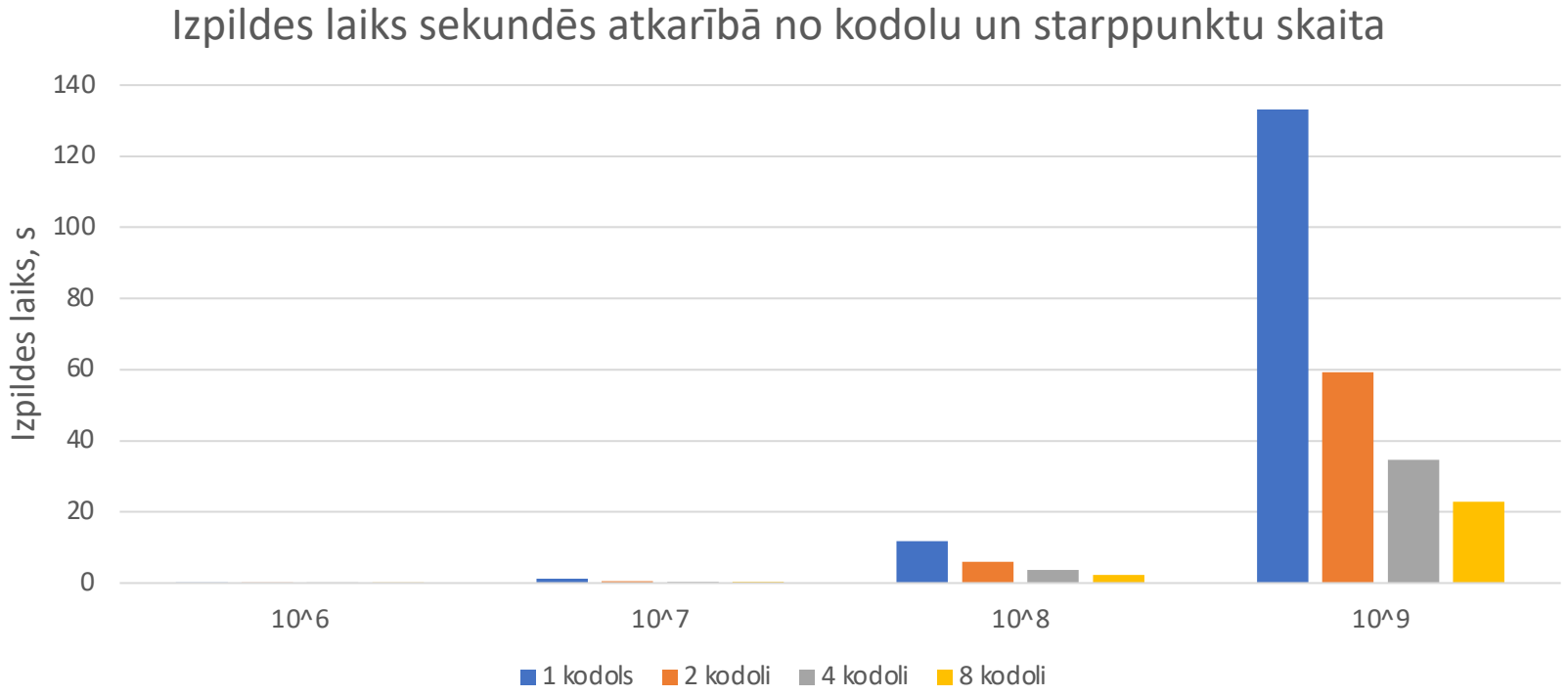
Idejas, ka šo algoritmu paralelizēt?

Sadala šo problēmu mazākās problēmās

- p – kodolu skaits
- N – kopējais starppunktu skaits
- Intervālu $[a, b]$ sadala mazākos ar vienādu garumu intervālos un iedod katram kodolam, lai rēķina pēc taisnstūru metodes ar $n = \frac{N}{p}$ starppunktiem.



Kādu paātrinājumu iegūstam, izmantojot vairākus kodolus



Kādu paātrinājumu iegūstam, izmantojot vairākus kodolus

	Kodolu skaits			
Starppunkti	1 kodols	2 kodoli	4 kodoli	8 kodoli
10^6	0.142	0.063	0.031	0.031
10^7	1.192	0.597	0.313	0.266
10^8	11.786	5.945	3.635	2.308
10^9	133.076	59.239	34.600	22.874

Smagāka problēma – pirmskaitļu meklēšana

Apsvērumi no matemātikas, kas var palīdzēt

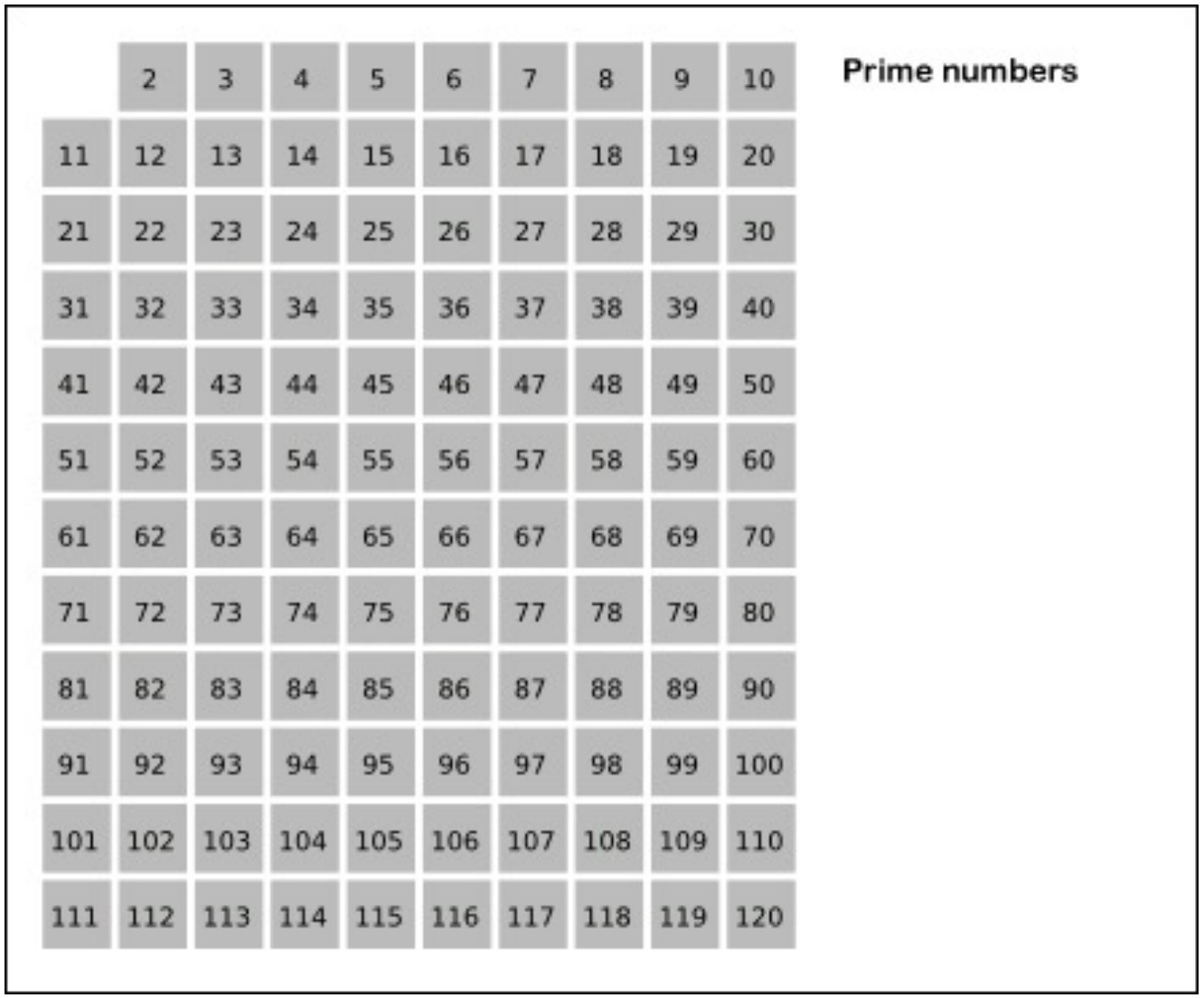
- Lai pārbaudītu vai skaitlis x ir pirmskaitlis, mēģina to dalīt ar visiem skaitļiem naturāliem skaitļiem no 2 līdz $\sqrt{x}$. Ja ar nevienu no tiem nedalās bez atlikuma, tad skaitlis x arī ir pirmskaitlis.

- Ja jau zināmi visi pirmskaitļi līdz skaitlim x (x neieskaitot), tad pietiek dalīt tikai jau ar zināmajiem pirmskaitļiem.

Sieti – Eratostena siets

- Uzraksta visus naturālos skaitļus no 2 līdz n. Sāk ar 2 – to pasludina par pirmskaitli un visus divnieka reizinājumus nosvītro.
- Dodas uz nākamo nenosvītroto skaitli un procesu atkārto. Beigās atlikušie nenosvītrotie skaitļi būs pirmskaitļi.

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, …



	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes

Vispirms izveidojam sekvenciālu jeb neparalēlu algoritmu

- Pārbaudīs visus skaitļus no 3 līdz n.
- Sāk darbu jau zinot, ka 2 ir pirmskaitlis.
- Pārbauda pirmo skaitļi 3 to dalot ar atrastajiem pirmskaitļiem. 3 nedalās ar 2 bez atlikuma, tādēļ 3 pievieno atrasto pirmskaitļu sarakstam.
- Tā atkārto līdz pārbaudīti visi skaitļi.

Kādas idejas paralelizācijai?

«Naivais» algoritms

- Pārbaudīs visus skaitļus no 3 līdz n.
- Līdzīgi kā laukuma zem līknes atrašanā, vienādā skaitā kodoliem izdala skaitļus, ko pārbaudīt
- Lai katra kodola darbs būtu neatkarīgs, tad lai pārbaudītu skaitli x to dala ar visiem skaitļiem no 3 līdz $\sqrt{x}$.
- Tā atkārto līdz pārbaudīti visi skaitļi un beigās visi kodoli padalās ar saviem atrastajiem pirmskaitļiem.

Vai varam izdomāt efektīvāku algoritmu?

- Ir p kodoli. Izdala katram pārbaudīt vienu skaitli, jau sākumā zinot, ka 2 ir pirmskaitlis.
- Katrs kodols pārbauda savu skaitli to dalot ar atrastajiem pirmskaitļiem.
- Kodoli apkopo informāciju, papildinot atrasto pirmskaitļu sarakstu un sāk pārbaudīt nākamos skaitļus.
- Atkārto līdz pārbaudīti visi skaitļi.

Kaut kas nav labi...

```
Aprekini tika veikti ar 7 kodoliem.
Lai aprekinatu visus pirmskaitlus no 1 lidz 20 bija nepieciesamas 0.0 sekund
es.
Tika atrasti 10 pirmskaitli.
[2, 9, 3, 5, 7, 11, 13, 23, 17, 19]
```

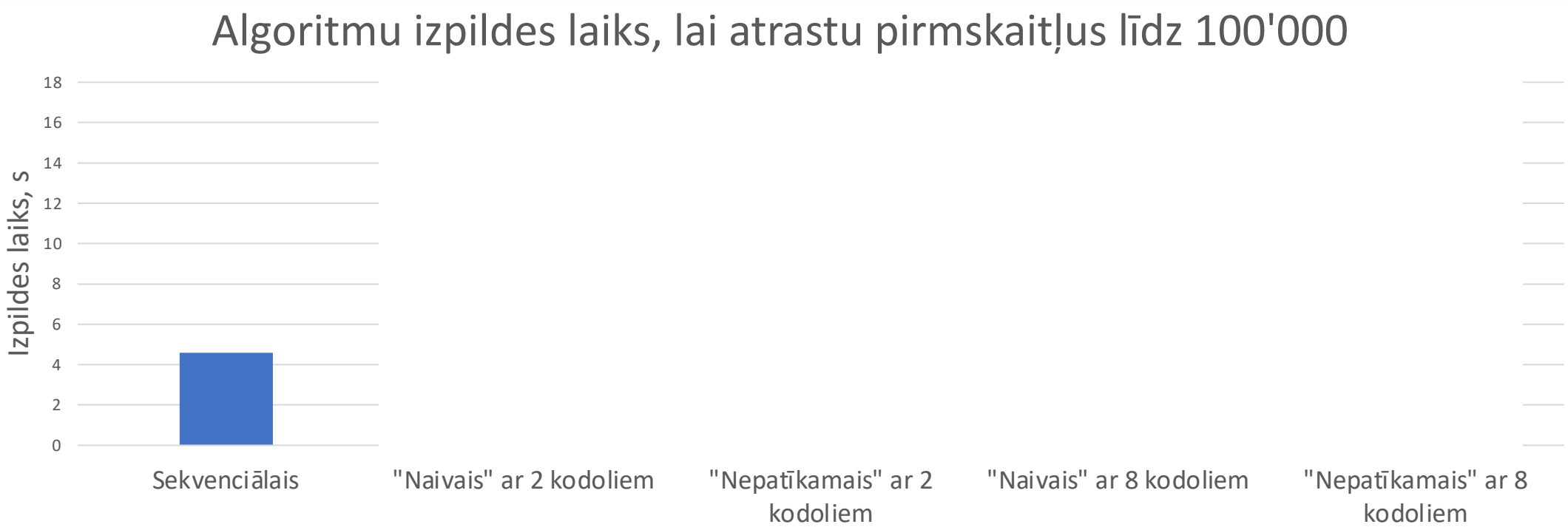
Kas nav labi algoritmam

- Pārējie kodoli nezina citu atrastos pirmskaitļus, līdz visi nav pabeiguši pārbaudīt savu skaitli. Piemēram, izskaidrosim situāciju un problēmu ar 9.
- 1. kodols pārbauda 3, to dalot ar atrastajiem pirmskaitļiem [2]
- 2. kodols pārbauda 4, to dalot ar atrastajiem pirmskaitļiem [2]
- …
- 8. kodols pārbauda 9, to dalot ar atrastajiem pirmskaitļiem [2], nezinot, ka 1. kodols jau nonāca pie secinājuma, ka 3 ir pirmskaitlis un arī ar to būtu jādala. Tādēļ 9 kļūdaini uzskatīs par pirmskaitli.

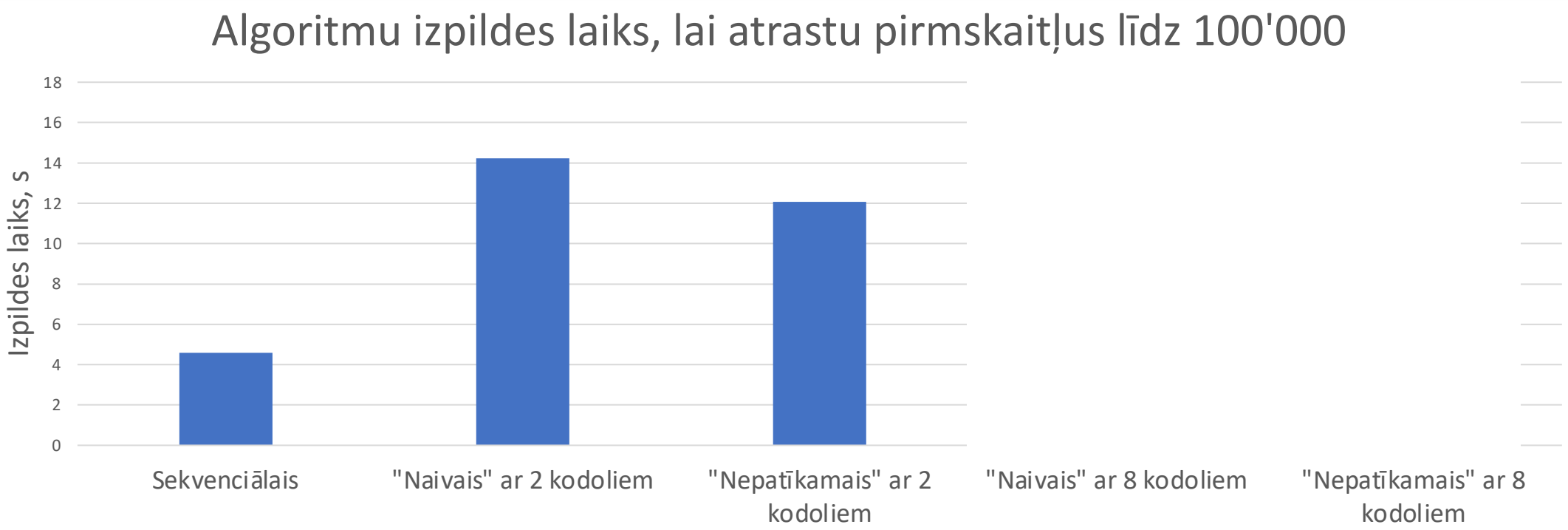
Risinājums

- Lai kodoliem darbi tiešām būtu neatkarīgi, tad atrastajiem pirmskaitļiem papildus pārbauda arī nākamos pāris skaitļus (gadījumā, ja cits kodols ir atradis pirmskaitli)

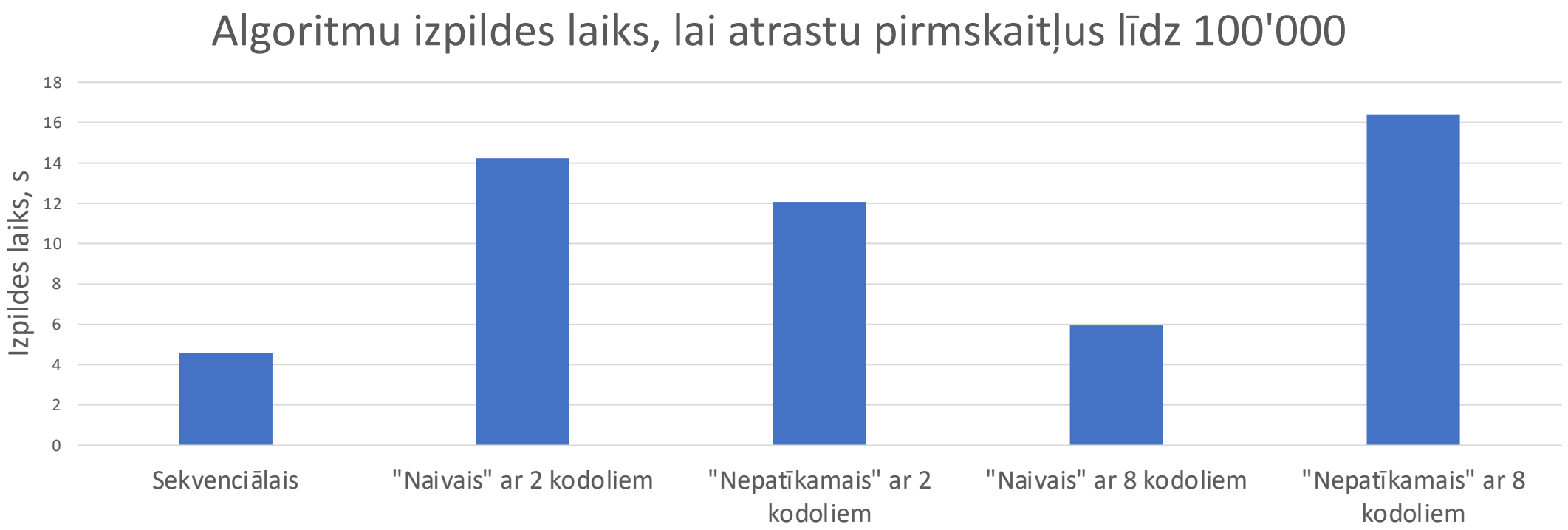
Algoritmu sacīkstes

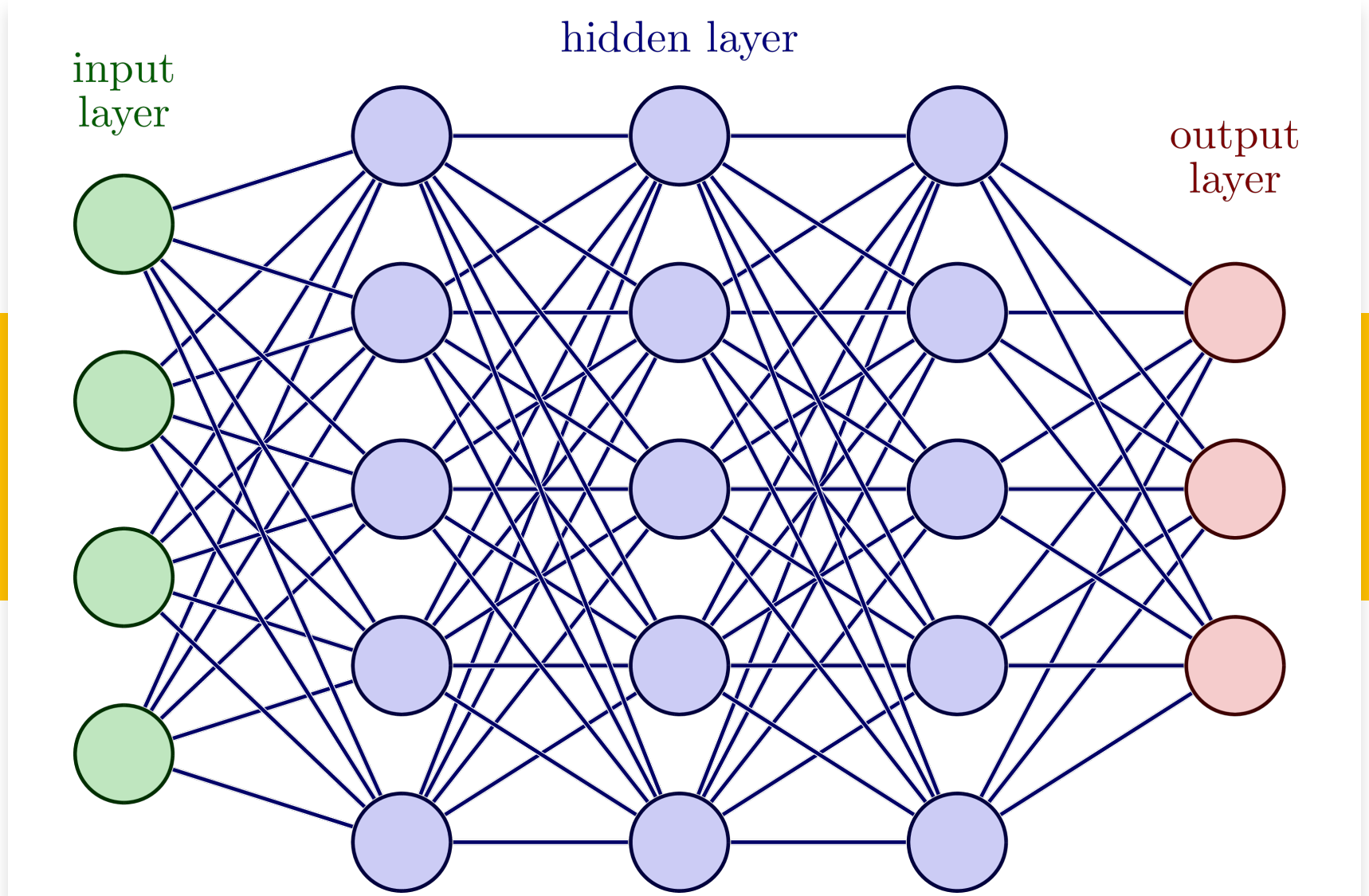


Algoritmu sacīkstes



Algoritmu sacīkstes

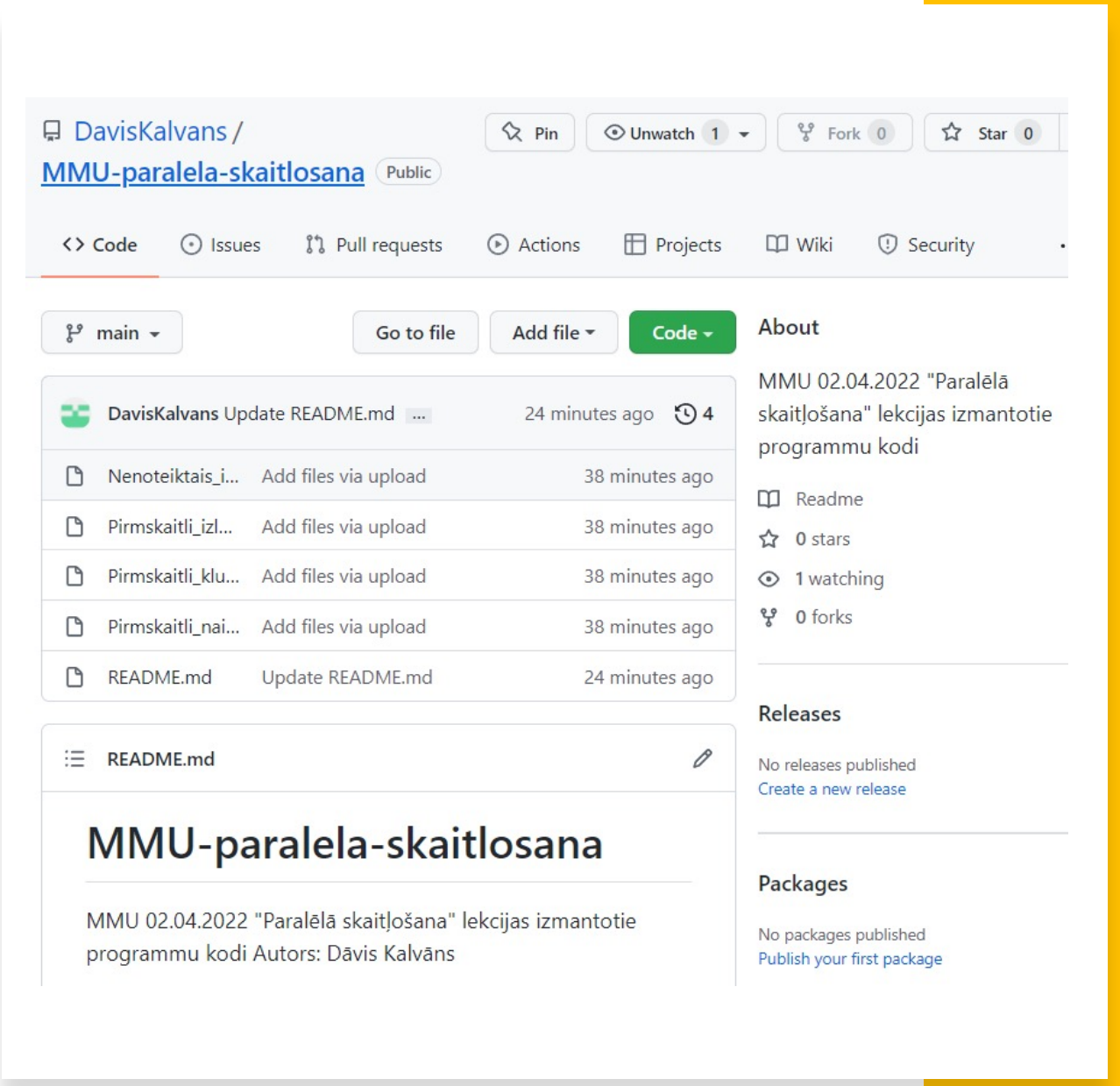




Paralēlās skaitļošanas viens no mūsdienu piemēriem - mašīnmācīšanās algoritmos

Visi aplūkoto algoritmu kodi pieejami GitHub

https://github.com/DavisKalvans/MMU-paralela-skaitlosana



Paldies par uzmanību!
Vai ir kādi jautājumi?