KĀRTOŠANAS ALGORITMUS REALIZĀCIJA UN TO ĪPATNĪBAS
SORTING ALGORITHMS REALIZATION AND THEIR FEATURES

Autors: **Ivars CABUĻEVS**, e-pasts: 19ivars98@gmail.com
Zinātniskā darba vadītājs: Mg.sc.ing., **Aleksejs ZORINS**
Rēzeknes Tehnoloģiju akadēmija, Rēzekne, Atbrīvošanas aleja 115

Abstract. *The performance of computers and programs is the most important thing for a user nowadays. Sorting algorithms appeared in the 19th century, but nowadays, developers often forget about the effectiveness of these algorithms and always use only a couple of algorithms, which are not always the best solution for certain tasks. This slows down the performance of certain important applications for professionals as well as for ordinary users. In this work was told and implemented different numerical sorting algorithms. The development environment here was "Microsoft Visual Studio 2017". Was used the C++ programming language. Knowledge of sorting algorithms will always help you to optimize your program (if there used sorting), which will have a positive impact on user feedback about your application also this knowledge has a positive effect on the thought processes, allowing you to make the right decisions in the shortest time.*

Keywords: *Sorting algorithm, efficiency of the algorithm, numeric sorting.*

Ievads

Jau no paša sākuma (20.gs.) tika pievērsta daudz uzmanību kārtošanas problēmām. Kā piemērs, burbuļa kārtošana tika apskatīta 1956. gadā. Neskatoties uz to, ka jau ir daudz kārtošanas algoritmus, cilvēki izgudro vēl jauno kārtošanas veidus. Kārtošanas algoritmi ir plaši izplatīti iesācējiem IT, kā arī tur, kur cilvēki tiks iepazīstināti ar algoritmiem.[1]

Kā reāls piemērs, Twitter, pirmajās dienās pēc izveides saskarās ar ražīguma problēmām. Tās bija saistītas ar kārtošanas algoritmiem, jo algoritms bija paredzēts tikai nelielam datu apjomam. [2]

Ir uzskats, ka lai kļūtu par programmēšanas inženieri, jāsāk mācīt algoritmus un datu struktūras. Un kad būs saprotami to pamati, tie būs redzami visur, un palīdzēs tev visas dzīves garumā, īpaši darbā IT nozarē.[3]

Mūsdienās arvien vairāk cilvēku ir ieinteresēti informācijas tehnoloģijām, īpaši programmēšanās. Internetā visbiežāk parādās neapstiprināta informācija. Līdz ar to visbiežāk rodas problēmas, tas viss nes sev līdzi negatīvas sekas, kas ietekmē programmu un ierīci veiktspēju.

Darba mērķis ir realizēt dažus kārtošanas algoritmus un novērtēt to efektivitāti savā starpā.

Materiāli un metodes

Pētījuma mērķis ir pierādīt **hipotēzi**, ka "burbuļa" kārtošanas algoritms ir lēnāk nekā "ātrās" kārtošanas algoritms uz datu apjomu no 300 tūkstošiem veseliem skaitļiem.

Šajā darbā tika izskatīti un īstenotie vairāki skaitļu kārtošanas algoritmi. izmantota izstrādes vide šeit ir "Microsoft Visual Studio 2017". Tika izmantota programmēšanas valoda C++.

Darba uzdevumi:

1. Pārskatīt visplašāk un vismazāk izplatītos kārtošanas algoritmus;
2. Realizēt vairākus kārtošanas algoritmus;
3. Novērtēt iegūtos kārtošanas algoritmus rezultātus.

Pētījumā ir pielietotas vairākas **metodes:**

- praktiskās pētījuma metodes
- eksperiments;
- salīdzināšanas metodes;

https://doi.org/10.17770/het2020.24.6744

- tabulas un grafiskās metodes;
- zinātnisko rakstu analīzes metodes;

Kārtošanas algoritmi

Kārtošanas algoritms ieņem nozīmīgu vietu programmēšanas jomā. Kārtošana nozīme noteikta saraksta elementus izvietošana noteiktā secībā. Lai optimizēti izmantotu tādus algoritmus, kā meklēšanas un apvienošanās algoritmus, kārtošanas algoritmiem jābūt efektīvam. Viss ir savstarpēji saistīts. Kārtošanas algoritmi tiek pārveidotas un parādās līdz pat šai dienai, tas ir saistīts ar tos svarīgumu IT nozarē. [4]

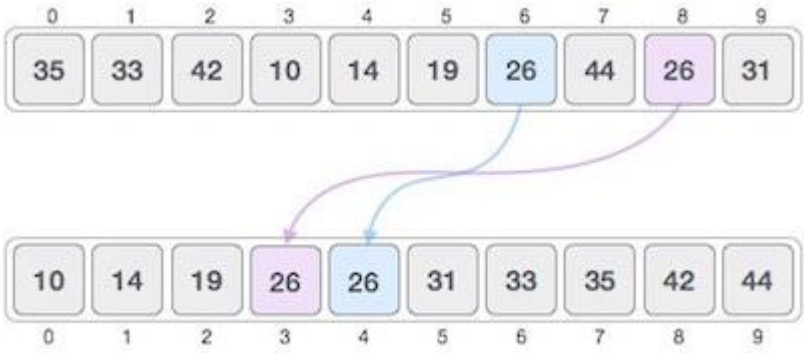
Visi kārtošanas algoritmi tiek klasificēti pēc:

- algoritma sarežģītība (labāka, sliktāka, vidēja), kas ir atkarīgi no saraksta izmēra (skat. 1.1.1. attēlu);
- atmiņas izmantošana;
- rekursijas pielietošana (vai ir rekursija, vai nē) (kad funkcija izraisa sevi);
- stabilitāte (ilgtspējīga kārtošana nemaina savstarpējo izvietojumu vienādiem elementiem. Šāda īpašība var būt ļoti noderīga, ja elementi sastāv no vairākiem laukiem) (piemēram, stabila ir burbuļa, ievietošanas, sapludināšanas kārtošana) (sk. 1.1.2., 1.1.3. att.);
- kārtošanas metode (apvienošana, maiņa, ievietošana);
- pielāgošana (vai ir efektīvi, apstrādājot jau sakārtoti (daļēji) sakārtoti dati)

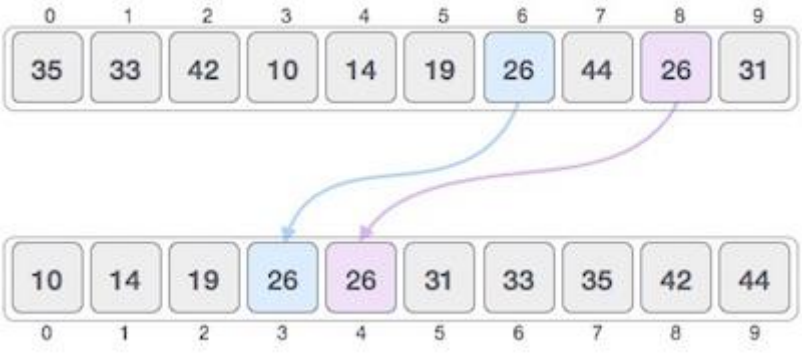
Klasifikācija sastāv

Algorithm	Time Complexity		
	Best	Average	Worst
Selection Sort	Ω(n^2)	θ(n^2)	O(n^2)
Bubble Sort	Ω(n)	θ(n^2)	O(n^2)
Insertion Sort	Ω(n)	θ(n^2)	O(n^2)
Heap Sort	Ω(n log(n))	θ(n log(n))	O(n log(n))
Quick Sort	Ω(n log(n))	θ(n log(n))	O(n^2)
Merge Sort	Ω(n log(n))	θ(n log(n))	O(n log(n))
Bucket Sort	Ω(n+k)	θ(n+k)	O(n^2)
Radix Sort	Ω(nk)	θ(nk)	O(nk)

1. att. Algoritma sarežģītība



2. att. Stabilā kārtošana



3. att. Nestabilā kārtošana

Apskatīsim sarežģītības:

O (O-notācija). O(f(n)) – no zīme, ka algoritma darba laika pieaugšana ir atkarīga no datu apjoma ne ātrāk par konstanti, kas reizināta ar f(n).

Θ (Θ – teta). f(x)=Θ(g(n)), nozīmē, ka f, tāpat kā g, aug tad, kad n tiecas uz bezgalību.

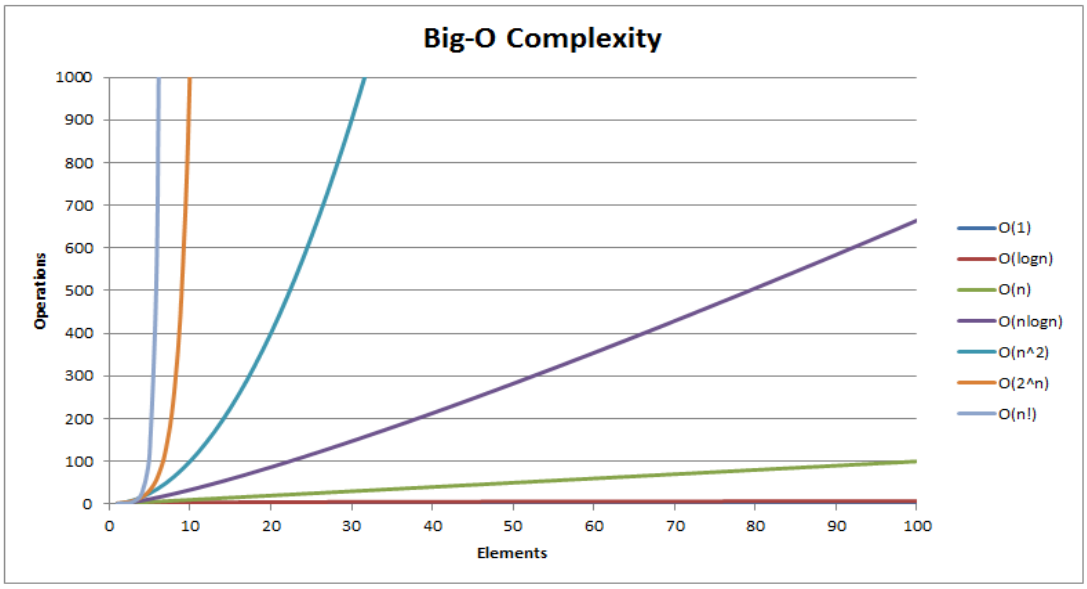
Ω (Ω – omega. Ω (n log n) pieprasa ne mazāk kā (n log n) laika, bet augšējā robeža nav zināms.

O(n) - ir lineāra sarežģītība. Tas nozīme, ka darba laiks pieaug lineāri, palielinoties izmēra ievadīto datu.

O(log n) - ir logaritmiskā sarežģītība. Ja masīvs ir sakārtots, mēs varam pārbaudīt, vai tajā ir kāda īpaša vērtība, to sadalot uz pusi. Mēs pārbaudīsim vidējo elementu, ja tas ir vairāk nekā tas, ko mēs meklējam, tad varam izmest masīva otro pusi - tur tas nav tieši tur. Ja tas ir mazāks, tad otrādi – varam izmest sākotnējo pusi. Tāpēc mēs turpinām sadalīt to pa pusēm, galu galā mēs pārbaudām (log n) elementus.

O(n^2) - ir kvadrātiskā sarežģītība. Algoritms ar šo sarežģītību sastāv no divām ciklam: viens, kas iet caur visu masīvu, un otrais, lai atrastu vietu nākamajam elementam jau nošķirtajā daļā. Tādējādi operāciju skaits būs atkarīgs no masīva lieluma n * n, tas ir, n^2.

O(n log n) – šī sarežģītība ir līdzīga ar O(n), jo viņa aizņem aptuveni divreiz vairāk laiku divreiz vairākām problēmām. Viņš ir efektīvs, jo logaritmi aug ļoti lēni.[10] (sk. 4. att.)



4. att. Sarežģītības atkarība no elementa skaitu

Rezultāti

Katrs algoritms savā veidā un kaut kādā vidē ir labs, bet ņemt pirmo ieraudzīto kļūdains risinājums. Tas nozīme, ka nepieciešams sīkāk iepazīties ar to, kāds ir gadījums un kādu algoritmu efektīvāk izvēlēt.

Iegūtie eksperimentālā daļā rezultāti (pēc 5 reizes vidēji pārbaudes) tika parādīti tabulā, kur "Sort" – pats algoritms, "CPU"- viss kods milisekundēs (1 sekunde = 1000 milisekundes). Var secināt, ka ātrās kārtošanas algoritms intervālā no 10000 līdz 5000000 elementiem neapšaubāmi labāk nekā burbuļa kārtošana, bet intervālā no 100 (kā arī no 1) līdz 10000 nav kritiskas atšķirības izvēlētajā algoritmā. Sapludināšanas kārtošana, salīdzinot ar ātrās kārtošanas metodēm, parādīja vispār nav daudz sliktāko rezultātu. Šis algoritms arī var tikt lietots. Ievietošanas kārtošana, tāpat kā burbuļa kārtošana, bija efektīva intervālā no 1 līdz 10000. (sk. 1. tab.)

1. tab.

Kārtošanas rezultāti

Name\Elements	100	1000	10000	100000	1000000	5000000
Bubble sort	0ms	1,8ms	238,8ms	28501,4ms	**too long**	**too long**
Bubble(CPU)	3,8ms	7,4ms	248,2ms	28830,8ms	**too long**	**too long**
Insertion sort	0ms	0ms	73,2ms	7106,2ms	**too long**	**too long**
Insertion(CPU)	4,2ms	4,6ms	80,6ms	7128,4ms	**too long**	**too long**
Quick sort	0ms	0ms	1ms	12ms	120,8ms	617,8ms
Quick(CPU)	4,6ms	5,4ms	10,6ms	75,8ms	683,6ms	3589,6ms
Merge sort	0ms	0,4ms	3ms	29,2ms	311,2ms	1628,8ms
Merge(CPU)	4,8ms	4,6ms	13,8ms	93,8ms	904,2ms	4446,2ms

Secinājumi

1. Darba hipotēze tiek apstiprināta, ātrās kārtošanas algoritms parādīja sevi labāk nekā burbuļa kārtošanas algoritms, bet reāli nozīmīga atšķirība ir pēc 10000 elementiem. Līdz 10000 elementiem burbuļa kārtošana var būt izmantota.

2. Izmantota programmatūra (eksperimentālā daļā) ļauj veiksmīgi veikt algoritma dziļu testēšanu (ja tās ir nepieciešams).
3. Pati kārtošanas algoritmi internetā ir korekti strādājošie, bet datu (skaitļu) sagatavošana pirms kārtošanas procesa ir jābūt pārbaudīta, jo tur dažos gadījumos var būt atrasta kļūda, kas neļaus kārtošanai strādāt.
4. Algoritma sarežģītība, mūsu gadījumā, ļoti stipri ietekmē uz kārtošanas laiku tikai tad, kad elementu skaits pārsniedz 10000.
5. Var pateikt, ka darbs ir veiksmīgs, jo mērķis tika sasniegts, daži algoritmi tika realizēti un novērtēti, darba uzdevumi tika izpildīti veiksmīgi.
6. Neskatoties uz to, ka kārtošanas algoritmiem ir tikai viens mērķis, viņi atšķiras pēc daudzām īpašībām, kā piemērs, stabilitāte, izpildes ātrums un laiks, rekursijas pieejamība, algoritma sarežģītības.

Summary

The document discusses and implemented popular sorting algorithms and talked about non popular algorithms that can be found on the internet. In this work no in-depth study of some sorting algorithms, but they're available for each explained principle work and some of their implementation in the programming language with summing up of the results. Sorting algorithms are relevant today. More and more people are buying new mobile phones and computers with the internet, must be sorted and stored a lot of information, but unreliable and unverified algorithms lead to countless problems with the performance of the application or some site. In work all tasks completed, the main goal is successfully achieved. Software, that has been used for testing is available on the internet to everyone for free, this means, that everyone after reading this work, the reader will understand the meaning of sorting algorithms and will be able to implement them.

Bibliography

1. C. A. Russell, Aaron Rotenberg, Ben Standeven. Sorting algorithm. Sk. internetā (21.06.2018) https://www.saylor.org/site/wp-content/uploads/2011/06/Sorting-Algorithm.pdf
2. Jordan Hudgens. Why are Sorting Algorithms Important? Sk. internetā (21.06.2018) https://www.crondose.com/2016/07/sorting-algorithms-important/
3. Jason Roell. Top Algorithms and Data Structures You Really Need To Know. Sk. internetā (23.06.2018) https://towardsdatascience.com/top-algorithms-and-data-structures-you-really-need-to-know-ab9a2a91c7b5
4. Bucket Sort.Sk. internetā (21.06.2018) http://www.growingwiththeweb.com/2015/06/bucket-sort.html