TIM SORT SARALASH ALGORITMI

Onarkulov Maksadjon Karimberdiyevich
Farg'ona davlat univarsiteti Amaliy matematika va informatika kafedrasi dotsenti (PhD)
maxmaqsad@gmail.com
Ismoilov Javohir Ulug'bek o'g'li
Farg'ona davlat universiteti 2-bosqich talabasi
Ismoilov Javohir Ulug'bek o'g'li

Anatatsiya:

Tim Sort algoritmi zamonaviy dasturlash tillarida ishlatiladigan samarali saralash algoritmlaridan biridir. U Python va Java kabi tillarning ichki sortlash funksiyalarida asosiy mexanizm sifatida qo'llaniladi. Tim Sort bir necha algoritmlarning afzalliklarini birlashtirgan bo'lib, xususan, Merge Sort va Insertion Sort yondashuvlarini kombinatsiyalab ishlaydi. Bu yondashuv tartiblangan qismlar (run'lar) bilan ishlashga asoslanadi va haqiqiy dunyo ma'lumotlarining tuzilmasiga yaxshi mos keladi. Ushbu maqolada Tim Sort algoritmining ishlash prinsipi, asosiy bosqichlari, afzalliklari va boshqa saralash algoritmlari bilan solishtirilgan tahlili beriladi.

Kalit so'zlar: *Tim Sort, saralash algoritmi, Merge Sort, Insertion Sort, run, adaptiv algoritm, Python sort, Java sort, murakkablik, samaradorlik, algoritm tahlili, optimal saralash, real ma'lumotlar, hibrid algoritm, dasturlash tillari*

Anatation:

Tim Sort is one of the most efficient and widely used sorting algorithms in modern programming languages. It is the default sorting mechanism in languages such as Python and Java. Tim Sort is a hybrid algorithm that combines the principles of Merge Sort and Insertion Sort to optimize performance on real-world datasets. It works by identifying ordered subsequences called "runs" and efficiently merging them. This paper presents the working principle of Tim Sort, its key stages, advantages, and comparison with other sorting algorithms.

Keywords: *Tim Sort, sorting algorithm, Merge Sort, Insertion Sort, run, adaptive algorithm, Python sort, Java sort, complexity, efficiency, algorithm analysis, optimal sorting, real-world data, hybrid algorithm, programming languages*

Аннотация:

Алгоритм Tim Sort — один из самых эффективных и широко используемых алгоритмов сортировки в современных языках программирования. Он используется по умолчанию в таких языках, как Python и Java. Tim Sort является гибридным алгоритмом, объединяющим принципы сортировки слиянием (Merge Sort) и сортировки вставками (Insertion Sort), что обеспечивает высокую производительность при работе с реальными данными. В данной статье рассматриваются принципы работы Tim Sort, его основные этапы, преимущества и сравнение с другими алгоритмами сортировки.

Ключевые слова: *Tim Sort, алгоритм сортировки, Merge Sort, Insertion Sort, run, адаптивный алгоритм, сортировка в Python, сортировка в Java, сложность, эффективность, анализ алгоритма, оптимальная сортировка, реальные данные, гибридный алгоритм, языки программирования*

Zamonaviy axborot texnologiyalari rivojlanib borishi bilan bir qatorda katta hajmdagi ma’lumotlar ustida samarali va tez ishlash zarurati ham ortib bormoqda. Shunday vaziyatlarda ma’lumotlarni tartibga keltirish, ya’ni saralash (sortlash) algoritmlarining roli nihoyatda muhim hisoblanadi. Saralash algoritmlari dasturlash va ma’lumotlar tuzilmalari fanining eng muhim yo‘nalishlaridan biri bo‘lib, ularning yordami bilan ma’lumotlar ustida tezkor qidiruv, tahlil, filtrlash va boshqa amallarni bajarish imkoniyati yaratiladi. Saralash algoritmlarining ko‘plab turlari mavjud bo‘lib, ularning har biri turli xil vaziyatlar va ma’lumotlar to‘plami uchun samarali yoki samarasiz bo‘lishi mumkin. Ushbu algoritmlar orasida Tim Sort algoritmi alohida o‘rin egallaydi.

Tim Sort algoritmi dastlab 2002-yilda Python dasturlash tilining ichki saralash mexanizmi sifatida Tim Peters tomonidan ishlab chiqilgan. Bu algoritm

oddiy ma’lumotlar to‘plamlarida ham, katta hajmdagi murakkab real ma’lumotlarda ham yuqori samaradorlikni ta’minlashga mo‘ljallangan. Tim Sort — bu hibrid saralash algoritmi bo‘lib, u ikkita mashhur algoritm: Insertion Sort va Merge Sort yondashuvlarini o‘zida mujassamlashtiradi. Shu jihati bilan u adaptiv xususiyatga ega bo‘lib, ya’ni ma’lumotlar allaqachon qisman tartiblangan holatda bo‘lsa, algoritm bu holatdan foydalanib, saralash vaqtini sezilarli darajada qisqartiradi. Aynan mana shu adaptivlik Tim Sort’ni boshqa ko‘plab klassik saralash algoritmlaridan ajratib turadi.

Tim Sort algoritmining ishlash asosida "run" deb nomlanuvchi tartiblangan kichik qatorlarni aniqlash yotadi. Algoritm ma’lumotlar to‘plamida avvaldan mavjud bo‘lgan tartibni aniqlab, bu qatorlarni aniqlik bilan birlashtiradi. Bu yondashuv klassik Merge Sort algoritmidan samaraliroq bo‘lib, kamroq xotira va kamroq vaqt talab qiladi. Shu sababli Tim Sort real hayotdagi ilovalar, ayniqsa, foydalanuvchilarning kundalik ma’lumotlar bilan ishlashiga asoslangan tizimlar uchun juda mos keladi. Tim Sort bugungi kunda nafaqat Python, balki Java kabi boshqa yirik dasturlash tillarida ham asosiy saralash mexanizmi sifatida foydalanilmoqda. Bu esa uning amaliy qiymatini yanada oshiradi.

Saralash algoritmlarining samaradorligi ko‘pincha ularning eng yomon (worst-case), o‘rtacha (average-case) va eng yaxshi (best-case) holatlaridagi murakkablik darajalari bilan baholanadi. Tim Sort algoritmi barcha holatlarda n*log(n) darajali murakkablikni ta’minlashga intiladi. Bu uni Quick Sort kabi tez ishlaydigan, ammo ba’zi hollarda n^2 murakkablikka tushib qoluvchi algoritmlarga nisbatan ishonchliroq qiladi. Aynan mana shu omillar Tim Sort’ni zamonaviy dasturlash tillarining ichki kutubxonalarida asosiy algoritm sifatida tanlanishiga sabab bo‘lgan.

Ushbu maqolada Tim Sort algoritmining ishlash mexanizmi, asosiy texnik bosqichlari, uning boshqa algoritmlar bilan taqqoslanishi va real tizimlarda qo‘llanilishi chuqur tahlil qilinadi. Shuningdek, bu algoritmning amaliy dasturlarda qanday afzalliklarga ega ekani, foydalanuvchilarga qanday qulayliklar yaratishi va uning kelgusidagi tadbiqot imkoniyatlari ham yoritiladi. Maqola

orqali o‘quvchi Tim Sort algoritmini nafaqat nazariy jihatdan, balki amaliy nuqtai nazardan ham chuqur tushunib oladi. Shu bilan birga, maqola dasturlash bilan shug‘ullanayotgan talabalar, dasturchilar va ilmiy izlanish olib borayotgan mutaxassislar uchun muhim qo‘llanma bo‘lib xizmat qilishi mumkin.

Tim Sort algoritmining ishlash prinsipi uning asosida yotgan hibrid yondashuvga asoslanadi. Bu algoritm ikki mashhur saralash usuli – Insertion Sort va Merge Sort algoritmlarining kuchli jihatlarini birlashtiradi. Avvalo, Tim Sort kiruvchi ma’lumotlar to‘plamida allaqachon mavjud bo‘lgan tartiblangan kichik qism segmentlarni, ya’ni “run” deb ataluvchi bo‘laklarni aniqlaydi. “Run”lar – bu ketma-ket tartiblangan (yoki teskari tartiblangan bo‘lsa, teskari aylantirib olingan) elementlar to‘plamidir. Tim Sort algoritmi ma’lumotlar ichida shunday “run”larni topadi va ularni minimal uzunlikka yetgunicha Insertion Sort yordamida to‘liq tartiblaydi. Insertion Sort kichik hajmdagi ma’lumotlarda tez va samarali ishlashi bilan tanilgan algoritm bo‘lib, ayni shu bosqichda u optimal natija beradi.

Keyingi bosqichda esa tartiblangan “run”lar o‘zaro birlashtiriladi. Bu bosqichda Merge Sort algoritmiga xos bo‘lgan yondashuv ishlatiladi, ya’ni ikkita tartiblangan “run” qiyoslanadi va bitta yangi tartiblangan qatorga birlashtiriladi. Tim Sort ushbu birlashtirish (merge) bosqichida muvozanatni saqlashga harakat qiladi, ya’ni “run”lar o‘lchamlarini hisobga olib, algoritm o‘zining “merge strategy” qoidalari asosida qaysi “run”larni birinchi birlashtirishni hal qiladi. Bu orqali Tim Sort saralash jarayonida vaqt va xotirani optimal tarzda boshqaradi.

Tim Sort algoritmi o‘zining ishlash jarayonida adaptiv xususiyatga ega, ya’ni kiruvchi ma’lumotlar qanchalik tartiblangan bo‘lsa, saralash jarayoni shunchalik tez amalga oshiriladi. Ayniqsa, real hayotdagi dasturlar uchun bu juda muhim, chunki amaliyotda ma’lumotlar ko‘pincha to‘liq aralashmasdan, qisman tartiblangan bo‘ladi. Tim Sort ana shu mavjud tartibdan maksimal foydalanib, saralash tezligini oshiradi. Shu bilan birga, algoritm barqaror (stable) hisoblanadi, ya’ni teng elementlar saralanganda ularning dastlabki joylashuvi saqlanib qoladi. Bu esa ba’zi hollarda, masalan, kompleks obyektlar bilan ishlaganda muhim ahamiyatga ega bo‘ladi.

Yuqoridagi bosqichlarning uyg‘unlashuvi orqali Tim Sort algoritmi yuqori tezlik, ishonchlilik va samaradorlikni ta’minlaydi. Bu algoritm nafaqat nazariy jihatdan, balki amaliy dasturlashda ham juda foydali va keng qo‘llaniladi. Aynan shuning uchun u ko‘plab dasturlash tillarining (masalan, Python, Java) standart kutubxonalarida asosiy saralash algoritmi sifatida tanlangan. Tim Sortning ishlash prinsipi uning aniqlik, mukammallik va real muhitda ishlashga mo‘ljallanganligi bilan boshqa algoritmlardan ajralib turadi.

Tim Sort algoritmi zamonaviy saralash algoritmlari ichida o‘zining afzalliklari bilan alohida ajralib turadi. Avvalo, uning asosiy ustunligidan biri – bu **adaptiv** algoritm ekanligidir. Ya’ni, Tim Sort kiritilgan ma’lumotlar to‘plamining tartib darajasini avtomatik aniqlab, unga mos strategiya asosida ishlaydi. Agar ma’lumotlar allaqachon qisman tartiblangan bo‘lsa, algoritm bu tartibdan maksimal darajada foydalanadi va saralash tezligini sezilarli darajada oshiradi. Bu jihat ayniqsa real hayotdagi dasturlarda juda foydali, chunki ko‘plab holatlarda foydalanuvchilarning ma’lumotlari to‘liq aralash bo‘lmaydi, balki qisman tartiblangan bo‘ladi. Shu sababli, Tim Sort ushbu turdagi ma’lumotlarda yuqori samaradorlik ko‘rsatadi.

Bundan tashqari, Tim Sort **barqaror (stable)** algoritmdir. Ya’ni, teng qiymatga ega elementlar saralanganda ularning dastlabki tartibi saqlanib qoladi. Bu holat ba’zi hollarda juda muhim bo‘lishi mumkin, masalan, murakkab obyektlarni (masalan, xodimlar ro‘yxatini ism bo‘yicha, keyin lavozim bo‘yicha saralash) tartiblashda. Yana bir muhim afzalligi – Tim Sort algoritmining **eng yaxshi, o‘rtacha va eng yomon holatdagi ishlash murakkabligi** barqaror ravishda O(n log n) bo‘lib, bu uni Quick Sort kabi tez, lekin eng yomon holatda sust ishlovchi algoritmlardan ishonchliroq qiladi. Aynan shu barqaror ishlash tezligi tufayli Tim Sort ko‘plab dasturlash tillarining standart kutubxonalarida asosiy saralash mexanizmi sifatida tanlangan. Misol uchun, Python va Java tillarida massivlarni saralashda Tim Sort algoritmi ishlatiladi.

Ammo Tim Sort algoritmining ayrim **kamchiliklari** ham mavjud. Uning ishlashi uchun qo‘shimcha **xotira sarfi** talab qilinadi, ayniqsa Merge bosqichida,

chunki bu bosqichda ikkita “run” birlashtirilganda yangi massivlar ajratiladi. Bu holat resurslar cheklangan muhitda muammo tug‘dirishi mumkin. Shuningdek, algoritmning ichki ishlash mexanizmi murakkab bo‘lgani sababli uni noldan yozish va tushunish Insertion Sort yoki Bubble Sort kabi oddiy algoritmlarga nisbatan ko‘proq bilim va vaqt talab qiladi. Shuningdek, kichik hajmdagi ma’lumotlar uchun oddiy Insertion Sort yoki boshqa engil algoritmlar Tim Sortdan ko‘ra tezroq ishlashi mumkin. Chunki Tim Sortda "run"larni aniqlash, ularni tartiblash va birlashtirish kabi qo‘shimcha amallar mavjud bo‘lib, bu kichik massivlar uchun ortiqcha murakkablik keltirib chiqaradi.

Shu bilan birga, algoritmda ishlatiladigan minimal "run" uzunligi dinamik hisoblab chiqiladi, lekin noto‘g‘ri tanlov qilingan taqdirda saralashning samaradorligi pasayishi mumkin. Shu sababli, Tim Sort faqat yirik va murakkab real hayotdagi ma’lumotlar bilan ishlaganda optimal natijani beradi. Biroq, umumiy jihatdan olganda, Tim Sort algoritmi o‘zining barqarorligi, adaptivligi, yuqori samaradorligi va real dasturlashda keng qo‘llanilishi bilan eng foydali saralash algoritmlaridan biri hisoblanadi. Ayniqsa, foydalanuvchi tajribasini oshirish va dastur tezligini ta’minlash zarur bo‘lgan ilovalarda Tim Sort ishonchli yechim bo‘la oladi.

Tim Sort algoritmi zamonaviy dasturlashda keng qo‘llaniladigan, amaliy jihatdan samaradorligi isbotlangan saralash algoritmlaridan biridir. Uning amaliy qo‘llanilishi real hayotdagi katta va murakkab ma’lumotlar to‘plamlarini tez va ishonchli tarzda tartiblash zarur bo‘lgan vaziyatlarda ayniqsa dolzarb hisoblanadi. Tim Sort algoritmi yuqori samaradorlik, barqarorlik va adaptivlikka ega bo‘lgani sababli ko‘plab sanoat darajasidagi ilovalarda qo‘llaniladi. Masalan, elektron tijorat saytlarida mahsulotlar ro‘yxatini narx yoki reyting bo‘yicha saralashda, katta hajmdagi log fayllarni vaqt tamg‘asi bo‘yicha tartiblashda, moliyaviy ilovalarda real vaqtli tranzaktsiyalarni to‘g‘ri ketma-ketlikda ko‘rsatishda Tim Sort algoritmi keng qo‘llanadi. Shuningdek, foydalanuvchi interfeysida jadval, ro‘yxat yoki boshqa ko‘rinishlarda dinamik saralash funksiyalarini amalga oshirishda ham ushbu algoritmning ishonchliligi muhim ahamiyatga ega bo‘ladi.

Tim Sort algoritmining dasturlash tillarida qo‘llanilishiga to‘xtaladigan bo‘lsak, u bugungi kunda bir nechta mashhur dasturlash tillarining standart kutubxonalarida asosiy saralash algoritmi sifatida tanlangan. Masalan, Python dasturlash tilida sort() funksiyasi yoki sorted() funksiyasi chaqirilganda, aslida Tim Sort algoritmi ishlatiladi. Bu esa Python dasturchilariga katta hajmdagi ro‘yxatlarni samarali saralash imkonini beradi. Java dasturlash tilida esa Arrays.sort() metodi, ayniqsa Object[] tiplari bilan ishlaganda, Tim Sort algoritmiga asoslangan. Bu Java ilovalarida massivlar va ro‘yxatlarni saralashda yuqori aniqlik va barqarorlikni ta’minlaydi. Shuningdek, Android ilovalari ishlab chiqishda ham Tim Sort algoritmiga asoslangan saralash funksiyalari ishlatiladi, bu esa foydalanuvchilarga tezkor javob va silliq interfeys taqdim etishga yordam beradi.

Shuni alohida ta’kidlash lozimki, Tim Sort algoritmi dasturlash tilining standart kutubxonasiga integratsiyalashganligi sababli, dasturchilar bu algoritmning murakkab mexanizmini batafsil bilmasdan turib ham undan samarali foydalanishlari mumkin. Bu esa ishlab chiqish jarayonini soddalashtirib, samaradorlikni oshiradi. Shu bilan birga, agar kerak bo‘lsa, Tim Sort algoritmini foydalanuvchi o‘zi ham Python, Java yoki C++ kabi tillarda amalga oshirishi mumkin, chunki uning ishlash prinsipi aniq ta’riflangan va o‘rganilgan. Natijada, Tim Sort real ilovalarda yuqori tezlik, aniqlik va barqarorlikka erishish uchun ideal yechim sifatida tan olinmoqda. Bu algoritmni amaliy dasturlashda tanlash – bu nafaqat samarali, balki ishonchli natijalarni ta’minlashga qaratilgan ongli qarordir.

Tim Sort algoritmi hozirgi zamonaviy dasturlash va kompyuter fanida eng samarali va ishonchli saralash algoritmlaridan biri sifatida alohida o‘rin egallaydi. Uning yuqori samaradorligi, barqarorligi, va real hayotdagi ma’lumotlarga moslashuvchanligi uni ko‘plab amaliy ilovalarda qo‘llashga imkon beradi. Ayniqsa, kiritilgan ma’lumotlar qisman tartiblangan holatda bo‘lsa, Tim Sort ushbu tartibni aniqlab, Insertion Sort va Merge Sort kombinatsiyasi orqali eng optimal natijani ta’minlaydi. Uning bu xususiyati nafaqat saralash tezligini

oshiradi, balki tizim resurslaridan oqilona foydalanish imkonini ham yaratadi. Shuningdek, barqaror (stable) saralash imkoniyati Tim Sort algoritmini murakkab ma’lumotlar strukturasi bilan ishlashda afzal tanlovga aylantiradi. Masalan, bir nechta mezonlar bo‘yicha saralash talab etilgan hollarda, birinchi mezon bo‘yicha saralangandan so‘ng ikkinchi mezon bilan ishlaganda elementlar tartibi saqlanib qolishi muhim bo‘ladi.

Taklif sifatida, dasturchilar va kompyuter fanlari bilan shug‘ullanuvchilar ushbu algoritmning ishlash prinsiplarini chuqur o‘rganishlari va uni amaliyotda qanday ishlashini real misollar orqali tahlil qilishlari zarur. Shuningdek, algoritmning ishlash samaradorligini boshqa saralash algoritmlari bilan solishtirib ko‘rish, uning kuchli va zaif tomonlarini aniqlashga yordam beradi. Dasturlash tillarining ichki kutubxonalarida Tim Sort keng qo‘llanilayotgan bo‘lsa-da, uni noldan yozish orqali algoritmik tafakkurni rivojlantirish, murakkabliklarni tahlil qilish, xotira boshqaruvi va optimizatsiya usullarini tushunishda foydali bo‘ladi. Kelgusida algoritmni yanada takomillashtirish yoki uni o‘zgartirilgan variantlarini yaratish uchun yangi yondashuvlarni ishlab chiqish mumkin. Xulosa qilib aytganda, Tim Sort nafaqat samarali saralash algoritmi, balki algoritmlarni tahlil qilish va ularni real hayotga tatbiq etishda mustahkam nazariy va amaliy asos bo‘lib xizmat qiladi. Shu sababli, bu algoritm ustida tadqiqotlar olib borish, uni amaliy tizimlarda yanada samarali qo‘llash yo‘llarini izlash doimo dolzarb bo‘lib qoladi.

Foydalanilgan adabiyotlar

1. Tim Peters. *Timsort Description*. Python Software Foundation.
2. Oracle Docs. *Sorting Algorithms in Java*. Oracle Corporation.
3. Thomas H. Cormen et al. *Introduction to Algorithms*. MIT Press, 3rd Edition.
4. Robert Sedgewick, Kevin Wayne. *Algorithms*. Addison-Wesley, 4th Edition.
5. Python Developers Guide. *Sorting HOW TO*. https://docs.python.org

6. Java SE Documentation. *Arrays.sort() Implementation*.

7. Wikipedia. *Timsort*. https://en.wikipedia.org/wiki/Timsort

8. GeeksforGeeks. *TimSort Algorithm*. https://www.geeksforgeeks.org

9. HackerRank. *TimSort Efficiency Analysis*.

10. JavaWorld. *Sorting in Java: TimSort Deep Dive*.

11. Stack Overflow Discussions. *Why Timsort is better for real-world data*.

12. ACM Digital Library. *TimSort Performance on Large Datasets*.

13. Medium Articles. *Understanding TimSort Algorithm*.

14. Dev.to. *Python's Sorting Secrets: TimSort Explained*.

15. Lecture Notes on Algorithms. *Hybrid Sorting Techniques*. Computer Science Department, Stanford University.