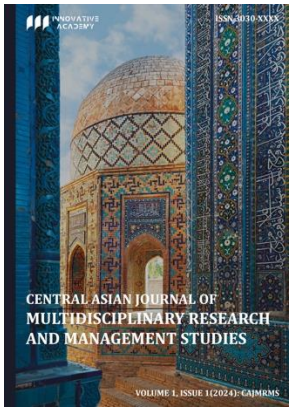

ALGORITM MURAKKABLIGI TAHLILI

Saidov Jasur Doniyor o‘g‘li
Guliston davlat universiteti o‘qituvchisi
Yo‘ldosheva Shohsanam Erkin qizi
Sayfullayeva Nafosat Jahongir qizi
Guliston davlat universiteti talabalari
https://doi.org/10.5281/zenodo.10916132

ARTICLE INFO

Received: 1st April 2024
Accepted: 2nd April 2024
Published: 4th April 2024

KEYWORDS

pragmatik dasturlash, big O, notatsiya, asimptotik xatti-harakatlar, eng yomon holatlar tahlili, assembler tili.

ABSTRACT

Ushbu maqolada algoritm murakkabligi va uning tahlilari organilgan. Olib borilgan ishlar taxlil qilingan. Muloxazalar va takliflar bildirilgan

Algoritmning murakkabligi dastur yoki algoritm qanchalik tez ishlashini rasman o'lchashning bir usuli bo'lib, bu juda pragmatik maqsaddir. Bilamizki, kod qanchalik tez ishlayotganini o'lchaydigan vositalar mavjud. Bular millisekundlarda bajarilish vaqtini aniqlaydigan, qiyinchiliklarni aniqlash va ularni optimallashtirishga yordam beradigan profilerlar deb ataladigan dasturlar. Ammo, bu foydali vosita bo'lsa-da, algoritmlarning murakkabligi bilan hech qanday aloqasi yo'q. Algoritmning murakkabligi ikkita algoritmni ideal darajada solishtirishga, dasturlash tilini amalga oshirish, dastur ishlayotgan apparat yoki berilgan protsessorda o'rnatilgan ko'rsatmalar kabi past darajadagi tafsilotlarga e'tibor bermaslikka asoslangan narsadir. Biz algoritmlarni ular aslida nima bo'lganligi nuqtai nazaridan solishtirmoqchimiz: hisoblash qanday ishlashi haqidagi g'oyalar. Bu erda millisekundlarni hisoblash juda oz yordam beradi. Ehtimol, past darajadagi tilda (masalan, assembler tilida) yozilgan yomon algoritm yuqori darajadagi dasturlash tilida (masalan, Python yoki Ruby) yozilgan yaxshi algoritmdan ancha tezroq bo'lishi mumkin. Shunday qilib, "eng yaxshi algoritm" nima ekanligini hal qilish vaqti keldi.

Algoritm - bu kompyuter tez-tez bajaradigan boshqa narsalarsiz, masalan, tarmoq vazifalari yoki foydalanuvchi kiritish/chiqishisiz, sof hisoblash dasturidir. Murakkablik tahlili ushbu dastur hisob-kitoblarni amalga oshirishda qanchalik tez ekanligini bilish imkonini beradi. Sof hisoblash operatsiyalariga misol sifatida suzuvchi nuqta operatsiyalari (qo‘shish va ko‘paytirish), operativ xotirada ma’lumotlar bazasidan berilgan qiymatni izlash, o‘yinning sun’iy intellektini (AI) uning xarakterini o‘yin dunyosi ichida faqat qisqa masofada harakatlanishi uchun harakatlantirishni aniqlash mumkin, yoki satrga nisbatan muntazam ifoda naqshini ishga tushirish. Shubhasiz, hisoblash kompyuter dasturlarida hamma joyda mavjud.

Murakkablik tahlili, shuningdek, kirish ma'lumotlari ko'payishi bilan algoritm qanday harakat qilishini tushuntirishga imkon beradi. Agar bizning algoritmimiz 1000 ta kirish bilan bir soniya davomida ishlayotgan bo'lsa, bu qiymatni ikki barobarga oshirsak, u qanday ishlaydi? Xuddi shunday tez ishlaydimi, bir yarim baravar tezroq yoki to'rt marta sekinroqmi? Dasturlash amaliyotida bunday bashoratlar nihoyatda muhimdir. Misol uchun, agar biz ming foydalanuvchisi bo'lgan veb-ilova uchun algoritm yaratgan bo'lsak va uning bajarilish vaqtini o'lchagan bo'lsak, unda murakkablik tahlilidan foydalanib, foydalanuvchilar soni ikki mingga etganida nima sodir bo'lishi haqida juda yaxshi tasavvurga ega bo'lamiz. Algoritm qurish musobaqalari uchun murakkablik tahlili, shuningdek, bizning kodimiz uning to'g'riligini tekshirish uchun testlarning eng uzunida qancha vaqt ishlashi haqida fikr beradi. Shunday qilib, agar biz oz miqdordagi kirish ma'lumotlari bo'yicha dasturimizning umumiy harakatini aniqlasak, katta ma'lumotlar oqimlari bilan nima sodir bo'lishi haqida yaxshi tasavvurga ega bo'lishimiz mumkin. Oddiy misoldan boshlaylik: massivdagi maksimal elementni topish.

Massivning maksimal elementini eng oddiy kod qismi yordamida topish mumkin. Misol uchun, biri Javascriptda yozilgan. n o‘lchamdagi A kirish massivi berilgan:

```
Var M = A[ 0 ];
For (var i = 0; i < n; ++i ) {
If ( A[ i ] >= M ) {
M = A[ i ];
}
}
```

Birinchidan, bu erda qancha fundamental ko'rsatmalar hisoblanganligini hisoblaylik. Biz buni faqat bir marta qilamiz – nazariyaga chuqurroq kirib borar ekanmiz, bu ehtiyoj yo'qoladi. Ammo hozircha bunga sarflagan vaqtimizga sabr qiling. Ushbu kodni tahlil qilish jarayonida uni oddiy ko'rsatmalarga - protsessor darhol yoki unga yaqin bajarishi mumkin bo'lgan vazifalarga bo'lish mantiqan to'g'ri keladi. Faraz qilaylik, bizning protsessorimiz quyidagi operatsiyalarni bitta buyruq sifatida bajarishga qodir:

O'zgaruvchiga qiymat belgilang
Massivdagi muayyan elementning qiymatini toping
Ikki qiymatni solishtiring
o'sish qiymati
Asosiy arifmetik amallar (qo‘shish va ko‘paytirish kabi)
Tarmoqlanish (if-shartini baholagandan so'ng kodning if va else qismlari o'rtasidagi tanlov) bir zumda sodir bo'ladi deb taxmin qilamiz va hisoblashda bu ko'rsatmani hisobga olmaymiz. Yuqoridagi kodning birinchi qatori uchun:

```
Var M = A[ 0 ];
```

ikkita ko'rsatma talab qilinadi: A[0] ni qidirish va M ga qiymat berish (biz n har doim kamida 1 deb hisoblaymiz). Ushbu ikkita ko'rsatmalar n qiymatidan qat'i nazar, algoritm tomonidan talab qilinadi. For tsikli ham har doim ishga tushiriladi, bu bizga yana ikkita buyruq beradi: tayinlash va taqqoslash.

```
i = 0;
i < n;
```

Bularning barchasi for ning birinchi yugurishidan oldin sodir bo'ladi. Har bir yangi iteratsiyadan so'ng bizda yana ikkita ko'rsatma bo'ladi: o'sish i va tsiklni to'xtatish vaqti kelganligini tekshirish uchun taqqoslash.

++i;

i < n;

Shunday qilib, agar biz halqa tanasining mazmunini e'tiborsiz qoldiradigan bo'lsak, unda ushbu algoritm uchun ko'rsatmalar soni 4 + 2n - for tsiklining boshida to'rtta va har bir iteratsiya uchun ikkitadan, bizda n ta bo'lak bor. Endi f(n) matematik funksiyasini shunday aniqlashimiz mumkinki, n ni bilib, algoritm talab qiladigan ko’rsatmalar sonini ham bilib olamiz. Bo'sh tanasi bo'lgan for tsikli uchun

f(n) = 4 + 2n.

Eng yomon vaziyatni tahlil qilish

Loopning tanasida bizda har doim sodir bo'ladigan massivlarni qidirish va taqqoslash operatsiyalari mavjud:

If (A[i] >= M) {

Lekin if ning tanasi massivdagi haqiqiy qiymatga qarab ishlashi yoki ishlamasligi mumkin. Agar A[i] >= M bo'lsa, biz ikkita qo'shimcha buyruqni bajaramiz: massivni qidirish va tayinlash:

M = A[i]

Biz endi f(n) ni bunchalik oson aniqlay olmaymiz, chunki endi ko'rsatmalar soni nafaqat n ga, balki ma'lum kirish qiymatlariga ham bog'liq. Masalan, A = [1, 2, 3, 4] uchun dastur A = [4, 3, 2, 1] ga qaraganda ko‘proq ko‘rsatmalarga muhtoj bo‘ladi.Algoritmlarni tahlil qilganda, biz ko'pincha eng yomon stsenariyni ko'rib chiqamiz. Bizning holatimizda bu qanday bo'ladi? Algoritmni bajarish uchun eng ko'p ko'rsatmalar qachon kerak bo'ladi? Javob, massiv A = [1, 2, 3, 4] kabi o'sish tartibida tartiblanganida. Keyin M har safar qayta tayinlanadi, bu eng ko'p buyruqlarni beradi. Nazariychilar buning uchun chiroyli nomga ega, eng yomon vaziyatni tahlil qilish, bu shunchaki eng yomon stsenariyni ko'rib chiqishdan boshqa narsa emas. Shunday qilib, eng yomon holatda, bizning kodimizdan tsiklning tanasida to'rtta ko'rsatmalar ishga tushiriladi va bizda f(n)=4+2n+4n=6n+4 mavjud.

Yuqoridagi funksiya yordamida biz algoritmimiz qanchalik tez ekanligi haqida juda yaxshi tasavvurga egamiz. Biroq, men va'da qilganimdek, dasturdagi ko'rsatmalarni hisoblash kabi zerikarli ish bilan doimiy shug'ullanishimiz shart emas. Bundan tashqari, foydalaniladigan dasturlash tilining har bir bandini amalga oshirish uchun ma'lum bir protsessor uchun ko'rsatmalar soni ushbu tilning kompilyatoriga va protsessor uchun mavjud bo'lgan ko'rsatmalar to'plamiga bog'liq (shaxsiy kompyuterda AMD yoki Intel Pentium, Playstation 2 da MIPS, va boshqalar.). Avvalroq biz bu kabi shartlarga e'tibor bermasligimiz haqida aytgan edik. Endi biz f funktsiyamizni nazariyotchilar e'tibor bermaslikni afzal ko'rgan mayda detallardan tozalash uchun "filtr" orqali o'tamiz.

Bizning 6n + 4 funksiyamiz ikkita elementdan iborat: 6n va 4. Murakkablikni tahlil qilishda n sezilarli darajada oshganda ko'rsatmalarni hisoblash funksiyasi bilan nima sodir bo'lishi muhim. Bu avvalgi "eng yomon holat" g'oyasiga to'g'ri keladi: biz qiyin ish qilishga majbur bo'lgan "yomon sharoitda" bo'lgan algoritmning xatti-harakati bilan qiziqamiz. E'tibor bering, bu algoritmlarni taqqoslashda juda foydali. Agar ulardan biri katta kirish ma'lumotlar oqimi bilan ikkinchisini mag'lub etsa, u engil, kichik oqimlarda tezroq qolishi mumkin. Shuning uchun biz funktsiyaning n o'sishi bilan sekin o'sadigan elementlarini yo'q qilamiz va faqat kuchli

o'sadiganlarini qoldiramiz. Shubhasiz, n qiymatidan qat'i nazar, 4 4 bo'lib qoladi, 6n esa, aksincha, ortadi. Shunday qilib, biz qiladigan birinchi narsa 4 ni tashlab, faqat f(n) = 6n ni qoldirishdir.

4 ni oddiygina "boshlash doimiysi" deb hisoblash mantiqan. Turli xil dasturlash tillarini sozlash har xil vaqt talab qilishi mumkin. Masalan, Java birinchi navbatda virtual mashinasini ishga tushirishi kerak. Va biz dasturlash tillaridagi farqlarga e'tibor bermaslikka rozi bo'lganimiz sababli, biz bu qiymatni bekor qilamiz.

E'tibor bermaslik kerak bo'lgan ikkinchi narsa - n ning oldidagi multiplikator. Shunday qilib, bizning funktsiyamiz f(n) = n bo'ladi. Ko'rib turganingizdek, bu juda ko'p narsani soddalashtiradi.

Xulosa qilib aytganda, algoritmlarni tahlil qilishning asosiy vazifasi kirish ma’lumotlari hajmining oshib borishi bilan resurslarga bo’lgan talabni (vaqt va xotira xarajatlari) o’lchash usullarini aniqlashdir.

FOYDALANILGAN ADABIYOTLAR:

1. Saidov, J., Matmusayeva, M., & To‘rayeva, Z. (2024). AXBOROT TIZIMLARI VA ULARNING RIVOJLANISHI OMILLARI. Центральноазиатский журнал междисциплинарных исследований и исследований в области управления, 1(4), 66-69.

2. Saidov, J. D. Study of the process of database and creation in higher education. Guliston. 2021.

3. Saidov, J., Irsaliyev, F., Temirxolova, B., & Ismoilova, C. (2024). TALABALARNING BILIM OLISHGA BO ‘LGAN QIZIQISHLARINI OSHIRISH MUAMMOLARI. Центральноазиатский журнал междисциплинарных исследований и исследований в области управления, 1(2), 134-137.

4. Saidov, J., Irsaliyev, F., Elmurodova, G., & Rustamova, M. (2024). TALABALARNING MA’LUMOTLAR BAZASINI YARATISH BO ‘YICHA BILIMLARINI BAHOLASH MEZONLARI. Центральноазиатский журнал междисциплинарных исследований и исследований в области управления, 1(2), 131-134.

5. Saidov, J., Nazarqulov, A., & Danaboyev, N. Z. (2024). ELEKTRON DIDAKTIK VOSITALAR YORDAMIDA BILIMLARNI SINASH MUAMMOLARI. Центральноазиатский журнал междисциплинарных исследований и исследований в области управления, 1(2), 143-147.

6. Saidov, J. D. O. G. L., Allayorov, S. P., & Islikov, S. X. (2021). MA’LUMOTLAR OMBORINI YARATISH BO ‘YICHA KASBIY KOMPETENTLIGINI BAHOLASH MEZONLARI. Scientific progress, 2(1), 1804-1807.

7. Islikov, S., Saidov, J., & Xolmuminov, D. (2023). MUSTAQIL TA’LIMNI SHARQ MUTAFAKKIRLARINING QARASHLARI ASOSIDA TASHKIL QILISH. Евразийский журнал технологий и инноваций, 1(5), 172-174.

8. O‘G‘Li, S. J. D. (2022). TA’LIM OLUVCHILARNING MA’LUMOTLAR BAZASI FANIGA BO ‘LGAN QIZIQISHLARINI KOMPETENSIYALIY YONDASHUVLAR ASOSIDA OSHIRISH MUAMMOLARI. Science and innovation, 1(B3), 89-93.

9. Irsaliyeva, S., Irsaliyev, F., & Mavlonov, S. (2024). FIZIKADAN NOSTANDART NAMOYISH TAJRIBALARINI BAJARISHDA O ‘QUVCHI KREATIV FAOLIYATINI RIVOJLANTIRISHNING PSIXOLOGIK XUSUSIYATLARI. Центральноазиатский журнал междисциплинарных исследований и исследований в области управления, 1(2), 84-89.

10. Toshtemirov, D. (2023). TECHNOLOGIES FOR CREATING E-LEARNING RESOURCES. Science and innovation, 2(B1), 396-401.

11. Toshtemirov, D. E., & Djumoboyeva, Y. E. (2021). METHODOLOGY OF PROGRAMMING OF PROBLEMS CONCERNING PYTHON DATABASE. Bulletin of Gulistan State University, 2021(2), 9-17.

12. Абдуқодиров, А. А., & Тоштемиров, Д. Э. (2019). Таълим муассасаларида ахбороткоммуникация технологияларидан фойдаланиш методикаси. Монография. Гулистон:"Университет, 232.

13. Toshtemirov, D. E. (2019). USING CLOUD TECHNOLOGY TO PROVIDE INFORMATION RESOURCES IN THE PROCESS OF DISTANCE LEARNING. Bulletin of Gulistan State University, 2020(1), 21-26.

14. Jonibekov, D. B. O. G. L., & Toshtemirov, D. (2021). AQLIY BILISH DARAJASINI ANIQLASHDA DIDAKTIK O 'YIN METODLARIDAN FOYDALANISH USULLARI. Scientific progress, 2(2), 1052-1062.

15. Toshtemirov, D., Muminov, B., & Saidov, J. (2020). Fundamentals of compilation of electronic tasks for students to test and strengthen their knowledge of database. International Journal of Scientific and Technology Research, 9(4), 3226-3228.