# ALGORITMLARNI TAHLIL QILISHNING NAZARIY VA AMALIY JIHATLARI

Toshkent Davlat Iqtisodiyot Universiteti
**Serobiddinova Moxlaroyim Sirojiddin qizi**
serobiddinovamoxlaroyim@gmail.com
Toshkent Davlat Iqtisodiyot Universiteti
**Ismailov Alisher Shakirovich**
alisherismailov1991@gmail.com

**Anontatsiya**:
Algoritm – berilgan natijaga erishish uchun qilinishi kerak bo‘lgan aniq ko‘rsatmalar ketma-ketligi hisoblanadi. Algoritm keng ma’noda faqat kompyuterga oid atama bo‘lmay, balki unda berilgan ko‘rsatmalarni bajara oluvchi har qanday narsaga oid deb ta’riflasa ham bo‘ladi. Algoritm - ma’lum bir turga oid masalalarni yechishda ishlatiladigan amallarning muayyan tartibda bajarilishi haqidagi aniq qoida (dastur) hisoblanadi. Kibernetika va matematikaning asosiy tushunchalaridan biri ham hisoblanadi.

Har qanday algoritm quyidagi asosiy xususiyatlarga ega bo‘lishi kerak:

1. Diskretlilik (Cheklilik).
2. Tushunarlilik
3. Aniqlik
4. Ommaviylik
5. Natijaviylik

Algoritmlarning ishlab chiqishning amaliyotda ikkita asosiy usullari mavjud:

1. matnli algoritmlar
2. sxematik(grafik) algoritmlar

## I. Matnli algoritm

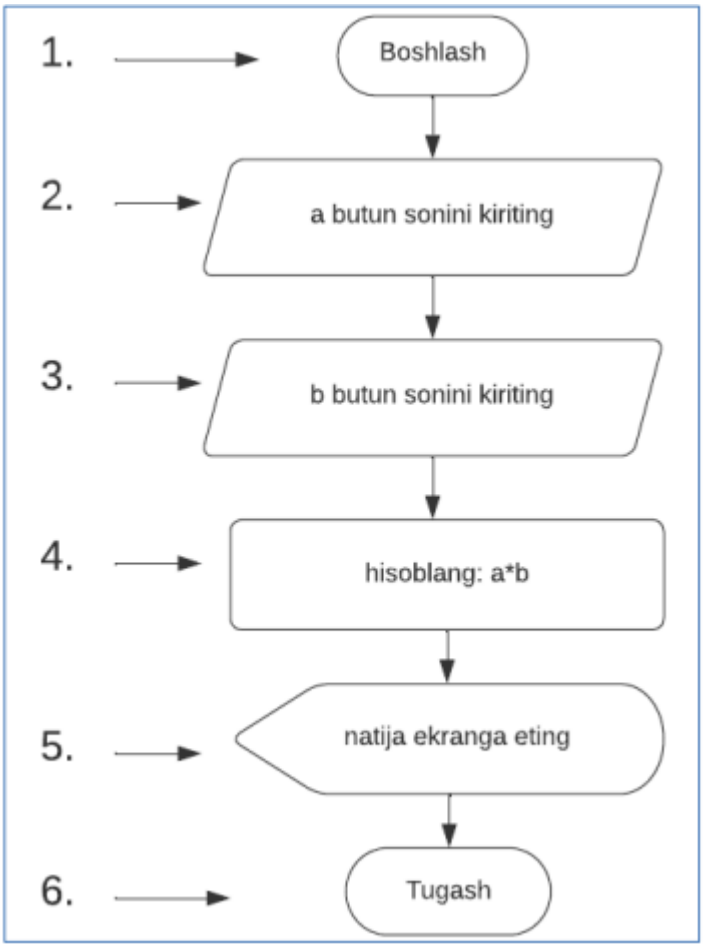
Matnli algoritm bu muammoni qadam va qadam yechimi va har qadam oddiy inson tushunadigan matn bilan ifodalanadi. Matnli algoritmning asosiy yutug‘i bu uning soddaligi hisoblanadi. Ya’ni dasrtuchi yoki informatik bo‘lmagan inson ham matnli algoritmni o‘qib masala yechimini tushunishi mumkin. Matnli algortimlar odatda sodda masalalar uchun qo‘llaniladi. Matnli algoritmning har bir qadami raqamlab boriladi. Esda saqlash kerak bo‘lgan qoida bu algoritmning birinchi qadami doim ‘boshlash’ va oxirgi qadami ‘tugatish, yaakunlash’ bo‘lishi kerak. Agar algoritm

qadamlarini oxiri ‘tugash’ bilan yakunlamasa bunday algoritm, algoritm deb tan olinmaydi. Mantli algoritmni dasturchi kodga o‘zgartishi juda oson hisoblanadi. Matnli algoritmga misol qilib kundalik hayotimizdagi vazifalarni keltirish ham mumkin. Masalan piyodalar o‘tish joyidan yo‘lni narigi tomoniga o‘tish algoritmi mana bunday ifodalanishi mumkin: Matnli algoritm piyodalar o‘tish joyidan yo‘lini kesib o‘tish:

1. Boshlash
2. chapga qarang
3. agar mashina yo‘q bo‘lsa, yo‘lni o‘rtasigacha yuring va to‘xtang
4. o‘ngga qarang
5. agar mashina yo‘q bo‘lsa, yuring
6. Tugatish.

**II. Sxematik(grafik) algoritmlar**

Aytib o‘tilganimizdek matnli algoritmlar murakkab masalalarni yechishda tushunmovchiliklar keltirib chiqaradi. Murakkab masalalarni yechish uchun grafik algoritmlar qo‘llaniladi. Bunday algoritmlar blok-sxemali algoritmlar deb ham ataladi. Blok-sxemali algoritmlar geometrik figuralar orqali ifodalanadi (2-mavzuda keltirilgan). Blok-sxemali algoritmlarda har bir geometrik figuraning o‘z vazifasi bor. Bu vazifalarni dasturchi bilishi shart hisoblanadi aks holda katta hajmdagi dastur tuzilishida uzulishlar va bu dastur muddatidan kechga qolib tayyorlanishiga olib keladi. Blok-sxemali algoritmlar masalani aniq yechimini taqdim etadi va bu dasturchiga ancha vaqt tejalishini ta’limlaydi. Matnli algoritmda aytib o‘tkanimizdek har bir algoritmning boshlash va tugash qadamlari shart bo‘lgan qadamlar hisoblanadi. Blok-sxemali algoritmda boshlash va tugatish buyruqlarini qovunsimon dumolaq figura ifodalaydi. Oddiy misol sifatida grafik algoritmga 2 ta butun sonni ko‘paytmasini topishni keltirishimiz mumkin.




1-qadam. Boshlash. Algoritmning boshlanishini ifodalaydi.
2-qadam. Kiritish ma’lumoti, bu yerda son kiritilyapti.
3-qadam. Kiritish ma’lumoti, bu yerda son kiritilyapti.
4-qadam.Hisoblash. a va b sonlarini ko‘paytmasini hisoblayapti.
5-qadam. Chop etish. Natijani ya’ni a va b sonlarni ko‘paytmasi natijasini ekranga chop etadi.
6-qadam. Tugash. Algoritmni yakuniga yetganini ifodalaydi.

## III. Algoritmni tahlil qilish

Algoritm tahlili aniq hisoblash muammosini hal qilish uchun algoritmning zarur resurslarini nazariy baholashni ta’minlaydi. Ko‘pgina algoritmlar ixtiyoriy uzunlikdagi kirishlar bilan ishlash uchun mo‘ljallangan. Algoritmlarni tahlil qilish - uni bajarish uchun zarur bo‘lgan vaqt va xotiradigi joy resurslari miqdorini aniqlash jarayoni hisoblanadi. Odatda, algoritmning samaradorligi ishlash vaqti, kirish ma’lumotlari ko‘pligi bilan baholanadi.

Algoritmlar ko‘pincha bir-biridan juda farq qiladi, ammo bir nechta algoritm bir xil natija taqdim etish uchun yaratilgan bo‘lishi ham mumkin. Masalan, raqamlar to‘plamini turli xil algoritmlar yordamida tartiblash mumkin. 2 ta algoritm bir xil

natija berganda, dasturchi sifatida bulardan qaysi biri samaraliroq ekanligini aniqlab o‘sha algoritmdan foydalanish maqsadga muvofiq hisoblanadi. Algortimning samaradorligi uning qancha vaqt sarflashi va qancha xotiradan joy talab etishi bilan baholanadi. Agar algoritm tez va xotirada oz joy talab etsa shu algortim samarador deb hisoblanadi. Algoritm tahlil turlari quyidagilardan iborat:

• Eng yomon holat - har qanday misolda bajarilgan qadamlarning soni maksimal darajada bo‘lishi

• Eng yaxshi holat - har qanday misolda bajarilgan qadamlarning soni minimal darajada bo‘lishi

• O‘rtacha holat - Har qanday misolda bajarilgan qadamlarning soni o‘rtacha darajada bo‘lish.

Algoritmlar dasturchiga berilgan muammoni hal qilish mumkinmi yoki yo‘qligini aniqlashga yordam beradi. Agar muammoni hal qilish mumkin bo‘lsa, qanday qilib, qanchalik tez va qanchalik aniq hal qilish mumkinligi haqida to‘liq ma’lumot beradi. Agar muammoni hal qilish mumkin bo‘lmasa, algoritm muammoni bir qismini hal qila oladimi yoki yo‘q degan savolga javob topishga yordam beradi.

## IV. Algoritmni dasturlashga bog‘lash

Har bir yaratilgan algoritm dasturlashda ifodalanishi mumkin. Eng muhimi algoritm dasturlash tili tanlamaydi. Ya’ni algoritm yaratilganidan keyin uni xoxlasangiz C++ dasturlash tilida, xoxlasangiz Piton (Python) dasturlash tilida dastur sifatida ifodalashingiz mumkin bo‘ladi. Algortim qadamlarini ifodalash uchun har bir dasturlash tilida maxsus jarayonlar mavjud. Masalan yuqoridagi misolda a va b sonlarini kiritish degan qadam bor. Ma’lumot kiritishni C++ dasturlash tilida cin>> orqali amalga oshirish mumkin. Yoki 5-qadamda natijani ekranga chop etish jarayoni mavjud, buni C++dasturlash tilida cout<< funksiyasi orqali amalga oshirish mumkin. Demak, yaratilayotgan algoritmlarni dasturiy ifodasini sizga ma’qul bo‘lgan dasturlash tilida amalga oshirish mumkin. Ushbu kitobda biz dasturlash tili sifatida C++ tilini o‘rganamiz. Dasturlashda nima uchun algoritmlardan foydalanishimiz kerakligi haqida gapirganda, kompyuter dasturlari protsessor va xotiraga ega kompyuter uskunasida ishlaydigan turli xil algoritmlarni qabul qiladi va bu komponentlar cheklovlarga ega hisoblanadi. Protsessor cheklangan resurslar iborat. Ulardan oqilona foydalanish kerak va vaqt nuqtai nazaridan samarali bo‘lgan yaxshi algoritm buni amalga oshirishga yordam beradi. Dasturlashda masalani yechishning turli usullari mavjud. Biroq, mavjud usullar samaradorligi bilan farq qiladi. Ba’zi usullar boshqalarga qaraganda aniqroq javob berish uchun juda mos keladi. Algoritmlar muammoni hal qilishning eng yaxshi usulini topish uchun ishlatiladi.

Algortimlar dastur samaradorligini ham oshiradi. Dastur samaradorligi turlicha talqin qilinishi mumkin. Ulardan biri dasturiy ta'minotning aniqligi hisoblanadi. Eng yaxshi algoritm bilan kompyuter dasturi juda aniq natijalarni berishi mumkin bo'ladi. Dasturiy ta'minot samaradorligini ko'rishning yana bir usuli - bu dasturning tezligi hisoblanadi. Dasturning masalani bajarish tezligini oshirish uchun algoritmdan foydalanish mumkin. Yaxshi algoritm muammoni hal qilish uchun dastur vaqtini qisqartirish imkoniyatiga ega hisoblanadi.

## V. Algoritmning afzalliklari va dasturlashdagi ahamiyati

Algoritmlar dasturchiga berilgan muammoni hal qilish mumkinmi yoki yo'qligini aniqlashga yordam beradi. Agar muammoni hal qilish mumkin bo'lsa, qanday qilib, qanchalik tez va qanchalik aniq hal qilish mumkinligi haqida to'liq ma'lumot beradi. Agar muammoni hal qilish mumkin bo'lmasa, algoritm muammoni bir qismini hal qila oladimi yoki yo'q degan savolga javob topishga yordam beradi

## VI. Foydalanilgan adabiyotlar:

[1] Ismailov, A., Jalil, M. A., Abdullah, Z., & Abd Rahim, N. H. (2016, August). A comparative study of stemming algorithms for use with the Uzbek language. In 2016 3rd
International conference on computer and information sciences (ICCOINS) (pp. 7-12).
IEEE.

[2] Jalil, M. M., Ismailov, A., Abd Rahim, N. H., & Abdullah, Z. (2017). The Development
of the Uzbek Stemming Algorithm. Advanced Science Letters, 23(5), 4171-4174.

[3] Allworth, E. A. (2013). The modern Uzbeks: from the fourteenth century to the present: a
cultural history. Hoover Press.

[4]Ismailov A Algoritmlash va dasturlash asoslari (2024)

[5] Y. AI-Nashashibi, D. D. Neagu and Y. Ali, "Stemming Techniques for Arabic Words: A Comparative Study," 2nd International Conference on Computer Technology and Development (ICCTD), 2010, pp. 270-276.

[6] H. Mohammad, B. Zuhair, C. Keely and M. David, "An Arabic Stemming Approach Using Machine Learning with Arabic Dialogue System," ICGST AIML-11
Conference, Dubai, April 2011, pp. 9-16.

[7] L. S. Leah, B. Lisa and C. E. Margaret, "Improving Stem- ming for Arabic Information Retrieval: Light Stemming and Co-occurance Analysis," SIGIR, ACM,

11-15 August 2002.
[8] L. S. Leah, B. Lisa and C. E. Margaret, "Conservatice Stemming for Search and Indexing," ACM, August 2005, pp. 15-19.
[9] S. Jikitsha and P. C. Bankim, "Stemming Techniques and Naïve Approach for Gujarati Stemmer," International Conference in Recent Trends in Information Technology and Computer Science, IJCA, 2012, pp. 9-11.
[10] A. F. Alajmi, E. M. Saad and M. H. Awadalla, "Hidden Makov Model Based Arabic Morphological Analyzer," International Journal of Computer Engineering Research, IJCER, Vol. 2, No. 2, 2011, pp. 28-33.
[11] M. Upendra and P. Chandra, "MAULIK: An Effective Stem- mer for Hindi Lanuage," International Journal of Computer Science and Engineering, IJCSE, Vol. 4, No. 5, 2012, pp. 711-717.
[12] R. Ananthakrishnana and R. D. Durgesh, "A Light Stem- mer for Hindi."
[13] K. Dinesh and R. Kumar, "Design and Development of Stemmer for Pujabi," International Journal of Computer Applications, IJCA, Vol. 11, No. 12, 2010, pp. 18-23
[14] F. B. William and F. J. Christopher, "Strength and Simi- larity of Affix Removal Stemming Algorithms," James Maison University and Virginia Tech.
[15] O. H. M. Ali and L. Ma Shi, "Stemming Algorithm to Clas- sify Arabaic Documents," Symposium on Progress in Information & Communication Technology, 2009, pp. 111- 115.
[16] A. James and K. Giridhar, "Stemming in the Language Modeling Framework," SIGIR, ACM, Toronto, 28 July-1 August 2003.
[17] A. Farag and N. Andreas, "N-Gram Conflation Approach for Arabic Text," SIGIR, ACM, Amsterdam, 7 July 2007