# MASSIVLARNI AJRATISH USULI BILAN SARALASHNI PYTHON DASTURLASH TILIDA O‘QITISH METODIKASI

**S.L.Fathiddinov**[1]

*1. Osiyo Xalqaro Universiteti “Aniq fanlar va raqamli texnologiyalar” kafedrasi o‘qituvchisi*

**Mas'ul muallif:** Fathiddinov Saidma’ruf Lazizxo‘ja o‘g‘li, Osiyo Xalqaro Universiteti “Aniq fanlar va raqamli texnologiyalar” kafedrasi o‘qituvchisi.

**e-mail:** fathiddinovsaidmaaruf@gmail.com

Received: 12 Noyabr 2026
Accepted: 17 Noyabr 2026
Published: 22 Noyabr 2026

**Copyright:** © 2026 by the author(s). Licensee Buxoro xalqaro universiteti ilmiy axborotnomasi.

**Iqtibos uchun:** Fathiddinov, S. L. MASSIVLARNI AJRATISH USULI BILAN SARALASHNI PYTHON DASTURLASH TILIDA O‘QITISH METODIKASI / S. L. Fathiddinov // Buxoro xalqaro universiteti ilmiy axborotnomasi. — 2026. — T. 11, № 2. — B. 183–187.

**Annotatsiya.** Maqolada massivlarni ajratish (Merge Sort) usuli bilan saralashni Python dasturlash tilida o‘qitish metodikasi yoritilgan. Ajratish usuli “bo‘lib tashla va hukmronlik qil” tamoyiliga asoslanib, massivni bir elementli qismlarga qadar ikkiga ajratadi va saralangan qismlarni tartib bilan birlashtiradi. Algoritmni o‘rgatishning bosqichma-bosqich modeli taklif etilgan: kompyutersiz faoliyat orqali g‘oyani anglash, birlashtirish amalini alohida o‘rganish, rekursiya daraxtini tasvirlash, Python’da dasturiy yechimni qurish va samaradorlikni tahlil qilish. Talabalar ko‘p yo‘l qo‘yadigan xatolar va ularni bartaraf etish yo‘llari ham ko‘rib chiqilgan.

**Kalit so'zlar:** massiv, saralash, ajratish usuli, Merge Sort, rekursiya, Python, o‘qitish metodikasi, algoritmik fikrlash.

## МЕТОДИКА ОБУЧЕНИЯ СОРТИРОВКЕ МЕТОДОМ РАЗДЕЛЕНИЯ МАССИВОВ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON

**Аннотация.** В статье рассматривается методика обучения сортировке массивов методом слияния (Merge Sort) на языке Python. Метод основан на принципе «разделяй и властвуй»: массив делится пополам до одноэлементных частей, после чего отсортированные части сливаются. Предложена поэтапная модель обучения: внекомпьютерная деятельность, отдельное освоение операции слияния, дерево рекурсии, пошаговое построение программы и анализ эффективности. Рассмотрены типичные ошибки студентов и способы их устранения.

**Ключевые слова:** массив, сортировка слиянием, Merge Sort, рекурсия, Python, методика обучения, алгоритмическое мышление.

## METHODOLOGY FOR TEACHING SORTING USING THE ARRAY PARTITIONING METHOD IN THE PYTHON PROGRAMMING LANGUAGE

**Abstract.** The article presents a methodology for teaching merge sort in Python. Based on the divide-and-conquer principle, the array is split in half down to single elements, and the sorted parts are then merged. A stage-by-stage teaching model is proposed: unplugged activities, learning the merge operation separately, recursion trees, incremental program construction and efficiency analysis. Typical student errors and ways to eliminate them are discussed.

**Keywords:** array, merge sort, recursion, Python, teaching methodology, algorithmic thinking.

Kirish. Dasturlashni o‘qitishda algoritmlar markaziy o‘rin egallaydi, ma’lumotlarni saralash esa izlash, tahlil va ma’lumotlar bazalari bilan ishlashning tayanch amali hisoblanadi. Ta’limda odatda pufakcha, tanlash va qo‘yish usullari o‘rganiladi, biroq ularning vaqt murakkabligi $O(n^2)$ bo‘lib, katta massivlarda samarasiz. 1945-yilda Jon fon Neyman taklif etgan ajratish usuli (Merge Sort) esa har qanday holatda $O(n \log n)$ murakkablikda ishlaydi va barqaror (stable) algoritm hisoblanadi.

Usulning mohiyati: massiv ikkiga ajratiladi, har bir qism xuddi shu usulda saralanadi va ikki saralangan qism bitta saralangan massivga birlashtiriladi. Biroq uni o‘qitishda qiyinchiliklar mavjud: rekursiya tushunchasi abstrakt, talabalar chaqiruvlar ketma-ketligini kuzatishda qiynaladi, birlashtirishda ikki indeks bilan bir vaqtda ishlash e’tibor talab qiladi. Python’ning ro‘yxat kesmalari, sodda rekursiv funksiyalari va Python Tutor kabi vizualizatsiya vositalari bu qiyinchiliklarni yengillashtiradi. Maqolaning maqsadi ajratish usulini Python’da o‘qitishning bosqichma-bosqich metodikasini ishlab chiqish va amaliy misollar bilan asoslashdan iborat.

Mavzuga oid adabiyotlar sharhi. D. Knut asarining uchinchi jildida birlashtirish orqali saralashning variantlari va tahlili batafsil yoritilgan. T. Kormen va hammualliflari ajratish usulini “bo‘lib tashla va hukmronlik qil” paradigmasining namunaviy misoli sifatida ko‘rib, ishlash vaqtini $T(n) = 2T(n/2) + \Theta(n)$ munosabati va rekursiya daraxti orqali asoslaydi. M. Gudrich va hammualliflari algoritmning Python ro‘yxatlari asosidagi talqinini beradi.

Metodik tadqiqotlardan K. Hundxauzen va hammualliflari vizualizatsiya samaradorligi talabaning faol ishtirokiga bog‘liqligini, R. Lister va hammualliflari esa boshlovchilarda kodni trassirovka qilish ko‘nikmasi zaif ekanligini ko‘rsatgan. F. Guoning Python Tutor vositasi funksiya chaqiruvlarini vizual kuzatish imkonini beradi, “CS Unplugged” loyihasida esa saralashni kompyutersiz o‘rgatish metodlari ishlab chiqilgan. O‘zbek tilidagi adabiyotlarda rekursiv saralashni Python’da o‘qitish metodikasi nisbatan kam yoritilgan.

Tadqiqot metodologiyasi. Tadqiqotda nazariy va algoritmik tahlil, modellashtirish, pedagogik kuzatish va dasturiy tajriba metodlaridan foydalanildi; maqoladagi barcha Python dasturlari bajarilib, natijalari tekshirildi. Metodik model “soddadan murakkabga”, “o‘qishdan yozishga” va “aniqdan mavhumga” tamoyillariga tayanadi. Uning muhim jihati — birlashtirish amali rekursiyadan oldin mustaqil ko‘nikma sifatida o‘rgatiladi. O‘qitish bosqichlari 1-jadvalda keltirilgan.

1-jadval

Ajratish usulini o‘qitish bosqichlari

| № | Bosqich | Mazmuni va vositalar | Kutilayotgan natija |
|---|---|---|---|
| 1 | G‘oyani anglash | Kartochkalarni guruhlarga bo‘lib saralash va juftlab birlashtirish (unplugged) | “Bo‘lib tashla va hukmronlik qil” g‘oyasini tushunadi |
| 2 | Birlashtirishni o‘rganish | Ikki saralangan ro‘yxatni birlashtirish, qo‘lda trassirovka | birlashtir() funksiyasini yozadi |
| 3 | Rekursiv ajratish | Rekursiya daraxti, Python Tutor’da bashorat qilib kuzatish | Chaqiruvlar ketma-ketligini tushuntiradi |
| 4 | Dasturiy amalga oshirish | ajratib_saralash() funksiyasini qurish va sinash | Ishlaydigan dastur yaratadi |
| 5 | Tahlil va mustahkamlash | Murakkablikni baholash, “xatoni toping” topshiriqlari | O(n log n) bahosini asoslaydi |

Tahlil va natijalar. Birinchi bosqichda talabalar raqamlangan kartochkalarni bittadan qolguncha ikkiga bo‘ladi, so‘ngra juft-juft bo‘lib saralangan holda birlashtiradi. Ikkinchi bosqichda quyidagi massiv misolida ishlanadi:

A = [38, 27, 43, 3, 9, 82, 10]

Saralash oxirida chap = [27, 38, 43] va o‘ng = [3, 9, 10, 82] qismlar birlashtiriladi: har ikki ro‘yxat boshidagi elementlar taqqoslanadi, kichigi natijaga qo‘shiladi va ko‘rsatkich suriladi (3, 9, 10, 27, 38, 43); bir ro‘yxat tugagach, ikkinchisining qolgan elementi (82) taqqoslashsiz qo‘shiladi. Talabalar buni avval qog‘ozda trassirovka qilib, so‘ngra Python’da yozadi:

```
def birlashtir(chap, ong):
natija = []
i = j = 0
while i < len(chap) and j < len(ong):
if chap[i] <= ong[j]:
natija.append(chap[i])
i += 1
else:
natija.append(ong[j])
j += 1
natija.extend(chap[i:])
natija.extend(ong[j:])
return natija
```

Uchinchi bosqichda ajratish daraxti chiziladi (o‘rta indeks len(a) // 2):

[38, 27, 43, 3, 9, 82, 10] → [38, 27, 43] | [3, 9, 82, 10]

[38] | [27, 43] [3, 9] | [82, 10] [27] | [43] [3] | [9] [82] | [10]

Qismlar pastdan yuqoriga birlashtiriladi: [27, 43] → [27, 38, 43]; [3, 9] va [10, 82] → [3, 9, 10, 82]; yakunda [3, 9, 10, 27, 38, 43, 82]. To‘rtinchi bosqichda rekursiv funksiyaning uch qismi — bazaviy holat, ajratish va birlashtirish — ajratib ko‘rsatiladi:

```
def ajratib_saralash(a):
if len(a) <= 1: # bazaviy holat
return a
orta = len(a) // 2 # ajratish
chap = ajratib_saralash(a[:orta])
ong = ajratib_saralash(a[orta:])
return birlashtir(chap, ong) # birlashtirish
print("Saralangan massiv:", ajratib_saralash([38, 27, 43, 3, 9, 82, 10]))
```

Natija: Saralangan massiv: [3, 9, 10, 27, 38, 43, 82]

Funksiyaga chuqurlik parametri qo‘shilib, har bir “Ajratish” va “Birlashtirish” natijasi chekinish bilan chiqarilsa, chekinishlar rekursiya daraxti darajalariga aynan mos keladi. Kamayish tartibida saralash uchun <= belgisini >= ga

almashtirish kifoya. Beshinchi bosqichda ataylab xato kiritilgan kodlar “xatoni toping” topshirig‘i sifatida beriladi (2-jadval).

2-jadval

Ajratish usulini dasturlashda ko‘p uchraydigan xatolar

| Xato | Oqibati | Bartaraf etish |
|---|---|---|
| Bazaviy holat (if len(a) <= 1) yo‘q | Cheksiz rekursiya, RecursionError | Rekursiv funksiyaning uch qismini aniq yozish |
| Qolgan elementlar natijaga qo‘shilmagan | Elementlar yo‘qoladi: A uchun natija [3] | natija.extend(chap[i:]) va natija.extend(ong[j:]) |
| Rekursiv chaqiruv natijasi olinmagan | Noto‘g‘ri natija: [3, 9, 38, 27, 43, 82, 10] | chap = ajratib_saralash(a[:orta]) |
| O‘rta indeks len(a) / 2 bilan hisoblangan | TypeError: kesma indeksi butun bo‘lishi kerak | Butun bo‘lish amali // dan foydalanish |

Samaradorlik rekursiya daraxti orqali tahlil qilinadi: daraxt balandligi taxminan $\log_2 n$, har bir darajada n tartibidagi amal bajariladi, demak murakkablik O(n log n), qo‘shimcha xotira esa O(n). Talabalar taqqoslashlar sonini hisoblaydi: pufakcha va tanlash usullarida bu n(n−1)/2 ga teng, ajratish usulida eng yomon holatda ham n·⌈log₂n⌉ − 2^⌈log₂n⌉ + 1 dan oshmaydi. Masalan, n = 1 000 da 499 500 ga qarshi 8 977, n = 100 000 da esa 4 999 950 000 ga qarshi 1 568 929 taqqoslash, ya’ni taxminan 3 000 marta kam. Pufakcha usuli barqaror va O(1) xotira talab qilsa, tanlash usuli barqaror emas; ajratish usuli esa barqaror, ammo qo‘shimcha xotira talab qiladi. Bu talabalarga algoritm tanlovi ma’lumotlar hajmi va xotira cheklovlariga bog‘liqligini anglatadi. Python’ning sorted() funksiyasidagi Timsort algoritmi ham birlashtirish g‘oyasiga asoslanganligi mavzuning amaliy ahamiyatini oshiradi.

Birlashtirish amalini puxta o‘zlashtirishda trassirovka jadvali muhim didaktik vosita bo‘lib xizmat qiladi. Talaba dasturni kompyuterda ishga tushirishdan avval har bir qadamda i va j ko‘rsatkichlarining qiymatini, taqqoslanayotgan juftlikni hamda natija ro‘yxatining joriy holatini qog‘ozda qayd etib boradi. Bunday mashq ikki indeks bilan bir vaqtda ishlash mexanizmini ko‘z oldiga keltirishga yordam beradi va siklning qaysi shartda to‘xtashini aniq tushunishga imkon yaratadi. Yuqorida ko‘rilgan chap = [27, 38, 43] va ong = [3, 9, 10, 82] ro‘yxatlarini birlashtirish jarayonining trassirovkasi 3-jadvalda aks ettirilgan.

3-jadval

Birlashtir() funksiyasining trassirovka jadvali

| Qadam | i | j | Taqqoslash | Qo‘shilgan element | Natija holati |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 27 va 3 | 3 | [3] |
| 2 | 0 | 1 | 27 va 9 | 9 | [3, 9] |
| 3 | 0 | 2 | 27 va 10 | 10 | [3, 9, 10] |
| 4 | 0 | 3 | 27 va 82 | 27 | [3, 9, 10, 27] |
| 5 | 1 | 3 | 38 va 82 | 38 | [3, 9, 10, 27, 38] |
| 6 | 2 | 3 | 43 va 82 | 43 | [3, 9, 10, 27, 38, 43] |
| 7 | 3 | 3 | — (chap tugadi) | 82 | [3, 9, 10, 27, 38, 43, 82] |

Jadvaldan ko‘rinadiki, uzunliklari m = 3 va k = 4 bo‘lgan ikki ro‘yxatni birlashtirish uchun oltita taqqoslash yetarli bo‘ldi. Muhokama davomida talabalar bilan umumiy qonuniyat keltirib chiqariladi: birlashtirishda taqqoslashlar soni kamida min(m, k) ga, ko‘pi bilan esa m + k − 1 ga teng. Eng kam holat bir ro‘yxatning barcha elementlari ikkinchisining eng kichik elementidan kichik bo‘lganda kuzatiladi, eng ko‘p holat esa umumiy eng katta ikki element turli ro‘yxatlarda joylashganda (bizning misolda 43 va 82) yuzaga keladi. Talabalarga ana shu ikki chegaraviy holatga mustaqil misol tuzish topshirig‘i beriladi, bu esa ularni algoritm xatti-harakatini kiruvchi ma’lumotlar bilan bog‘lab tahlil qilishga o‘rgatadi. 7-qadam alohida ta’kidlanadi: sikl tugagach, qolgan element taqqoslashsiz qo‘shiladi va aynan shu qatorni unutish 2-jadvalda qayd etilgan xatoga sabab bo‘ladi.

Rekursiv yechim o‘zlashtirilgach, talabalarga ajratish usulining iterativ, ya’ni “pastdan yuqoriga” (bottom-up) varianti taqdim etiladi. Bu yondashuvda massiv ataylab ikkiga bo‘linmaydi: dastlab har bir element uzunligi 1 ga teng saralangan qism sifatida qaraladi, so‘ngra qo‘shni qismlar juft-juft birlashtirilib, qism uzunligi har bir o‘tishda ikki barobar oshiriladi. Ushbu variantni o‘rganish ikki maqsadga xizmat qiladi. Birinchidan, talaba bitta masalani rekursiya va sikl yordamida turlicha hal etish mumkinligiga amin bo‘ladi. Ikkinchidan, avval yozilgan birlashtir() funksiyasidan qayta foydalanish modulli dasturlashning amaliy afzalligini yaqqol namoyon etadi:

```
def iterativ_saralash(a):
natija = list(a)
qadam = 1
while qadam < len(natija):
yangi = []
for k in range(0, len(natija), 2 * qadam):
chap = natija[k:k + qadam]
ong = natija[k + qadam:k + 2 * qadam]
```

```
yangi.extend(birlashtir(chap, ong))
natija = yangi
qadam *= 2
return natija
```

A massiv uchun dastur uch o‘tishda natijaga erishadi: qadam = 1 bo‘lganda [27, 38, 3, 43, 9, 82, 10], qadam = 2 da [3, 27, 38, 43, 9, 10, 82], qadam = 4 da esa yakuniy [3, 9, 10, 27, 38, 43, 82] ro‘yxati hosil bo‘ladi. O‘tishlar soni $\lceil \log_2 7 \rceil = 3$ ekanligi rekursiya daraxtining balandligi bilan solishtiriladi va talabalar ikkala variantning murakkabligi bir xil degan xulosaga o‘zlari keladi. Shuningdek, massiv uzunligi ikkining darajasi bo‘lmagan holatda oxirgi qism to‘liq bo‘lmasligi (masalan, birinchi o‘tishda [10] elementi juftsiz qolishi) va Python kesmalari bu vaziyatni hech qanday qo‘shimcha tekshiruvsiz to‘g‘ri qayta ishlashi alohida ko‘rsatib o‘tiladi.

Mustahkamlash bosqichida ajratish usulining g‘oyasi boshqa masalalarga ko‘chiriladi. Shunday masalalardan biri massivdagi inversiyalar sonini aniqlashdir. Agar $p < q$ bo‘lib, $a[p] > a[q]$ shart bajarilsa, (p, q) juftligi inversiya deb ataladi. Inversiyalar soni massiv tartiblangan holatdan qanchalik uzoqligini bildiradi va, masalan, ikki foydalanuvchining reytinglarini o‘zaro qiyoslashda qo‘llaniladi. Bu sonni ikki ichma-ich sikl orqali $O(n^2)$ vaqtda topish mumkin, biroq birlashtirish jarayoniga bitta amal qo‘shish bilan uni O(n log n) vaqtda hisoblash ham mumkin. Gap shundaki, o‘ng qismdagi element natijaga o‘tayotgan paytda chap qismda qolgan barcha elementlar undan katta bo‘ladi, demak, inversiyalar soniga len(chap) − i qo‘shiladi:

```
def inversiyalar(a):
if len(a) <= 1:
return a, 0
orta = len(a) // 2
chap, x = inversiyalar(a[:orta])
ong, y = inversiyalar(a[orta:])
natija, i, j, z = [], 0, 0, 0
while i < len(chap) and j < len(ong):
if chap[i] <= ong[j]:
natija.append(chap[i]); i += 1
else:
natija.append(ong[j]); j += 1
z += len(chap) - i
natija += chap[i:] + ong[j:]
return natija, x + y + z
```

A = [38, 27, 43, 3, 9, 82, 10] massivi uchun funksiya ([3, 9, 10, 27, 38, 43, 82], 11) qiymatini qaytaradi, ya’ni massivda 11 ta inversiya bor. Talabalar bu natijani ikki ichma-ich siklga asoslangan sodda dastur yordamida tekshirib ko‘radi. Mazkur topshiriq algoritmni yodlab olishdan ko‘ra uning ichki mantiqini anglashni talab etadi, chunki talaba birlashtirish jarayonida qaysi ma’lumot qo‘shimcha hisob-kitobsiz, “tekinga” olinayotganini tushunib yetishi kerak.

Guruhdagi talabalarning tayyorgarlik darajasi bir xil bo‘lmagani sababli amaliy mashg‘ulotlarda darajalashtirilgan topshiriqlar tizimini qo‘llash maqsadga muvofiq. Topshiriqlar reproduktiv, produktiv va ijodiy darajalarga ajratiladi (4-jadval). Har bir talaba dastlab birinchi daraja topshiriqlarini bajaradi, keyin esa o‘z imkoniyatiga qarab yuqoriroq darajaga o‘tadi. Bunday tashkil etish kuchli talabalarning zerikib qolishining oldini oladi, qiynalayotganlarga esa asosiy ko‘nikmani shoshilmasdan shakllantirish imkonini beradi.

4-jadval

Ajratish usuli bo‘yicha darajalashtirilgan topshiriqlar

| Daraja | Topshiriq namunasi | Shakllanadigan ko‘nikma |
|---|---|---|
| Reproduktiv | Berilgan massiv uchun rekursiya daraxtini chizish; birlashtir() funksiyasini trassirovka jadvali asosida qo‘lda bajarish | Algoritm qadamlarini aniq takrorlash |
| Produktiv | Ro‘yxatni kamayish tartibida, so‘zlarni uzunligi bo‘yicha yoki talabalarni o‘rtacha bali bo‘yicha saralash; iterativ variantni yozish | Algoritmni yangi shartlarga moslashtirish |
| Ijodiy | Inversiyalarni sanash; massivni uch qismga ajratuvchi variantni tuzish; ajratish usuli, pufakcha usuli va sorted() ish vaqtini time moduli yordamida qiyoslash | Algoritmni tahlil qilish va yangi masalaga ko‘chirish |

Talabalar bilimini baholashda yakuniy natija bilan birga jarayonga oid ko‘rsatkichlar ham inobatga olinadi. Xususan, trassirovka jadvalining to‘g‘ri to‘ldirilgani, Python Tutor’da kuzatishdan oldin aytilgan bashoratning aniqligi, dastur kodining tushunarli va izohlangan holda yozilgani hamda murakkablik bahosining asoslangani alohida mezonlar sifatida baholanadi. Bashorat topshiriqlari ayniqsa samara beradi: talaba keyingi chaqiruv qaysi kesma bilan bajarilishini yoki natija ro‘yxati qanday ko‘rinish olishini oldindan aytadi, so‘ngra vizualizatsiya orqali o‘z taxminini tekshiradi. Natijada vizualizatsiya passiv kuzatishdan faol o‘rganish vositasiga aylanadi.

Mashg‘ulot yakunida ajratish usulining amaliy qo‘llanilish sohalari muhokama qilinadi. Tezkor xotiraga

sig‘maydigan katta hajmli fayllarni saralashda ma’lumotlar bo‘laklarga ajratiladi, har bir bo‘lak alohida saralanadi va keyin diskda birlashtiriladi; tashqi saralash deb ataluvchi bu usul aynan birlashtirish g‘oyasiga asoslanadi. Usul bog‘langan ro‘yxatlarni saralashda ham qulay, chunki unda elementlarga indeks orqali ixtiyoriy murojaat qilish zarur emas. Mustaqil qismlarni bir vaqtning o‘zida saralash mumkinligi esa usulni parallel hisoblashlarda qo‘llashga yaroqli qiladi. Bunday misollar talabalarga o‘rganilayotgan algoritm shunchaki o‘quv mashqi emas, balki real dasturiy tizimlarning tarkibiy qismi ekanini ko‘rsatadi va mavzuga qiziqishni kuchaytiradi.

Xulosa va takliflar. Massivlarni ajratish usuli bilan saralashni Python’da o‘qitishning besh bosqichli metodikasi ishlab chiqildi va amaliy misollar bilan asoslandi. Quyidagilar taklif etiladi: birlashtirish amalini rekursiyadan oldin alohida o‘rgatish; rekursiv chaqiruvlarni daraxt ko‘rinishida chizish va Python Tutor’da bashorat topshiriqlari bilan kuzatish; dastur yozishdan oldin trassirovka jadvallarini to‘ldirish; taqqoslashlar sonini hisoblash orqali algoritmlarni qiyoslash; “xatoni toping” topshiriqlaridan foydalanish. Ajratish usulini o‘rganish rekursiv fikrlash va algoritm samaradorligini baholash ko‘nikmalarini shakllantiradi hamda tezkor saralash kabi algoritmlarni o‘rganish uchun asos yaratadi.

**Foydalanilgan adabiyotlar ro'yxati**

[1] Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. – 3rd ed. – MIT Press, 2009.

[2] Knuth D. E. The Art of Computer Programming, Volume 3: Sorting and Searching. – 2nd ed. – Addison-Wesley, 1998.

[3] Sedgewick R., Wayne K. Algorithms. – 4th ed. – Addison-Wesley, 2011.

[4] Goodrich M. T., Tamassia R., Goldwasser M. H. Data Structures and Algorithms in Python. – Wiley, 2013.

[5] Hundhausen C. D., Douglas S. A., Stasko J. T. A meta-study of algorithm visualization effectiveness // Journal of Visual Languages and Computing. – 2002. – Vol. 13, № 3. – P. 259–290.

[6] Lister R. et al. A multi-national study of reading and tracing skills in novice programmers // ACM SIGCSE Bulletin. – 2004. – Vol. 36, № 4. – P. 119–150.

[7] Guo P. J. Online Python Tutor: embeddable web-based program visualization for CS education // Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE ’13). – 2013. – P. 579–584.

[8] Левитин А. В. Алгоритмы: введение в разработку и анализ. – Москва: Вильямс, 2006.

[9] Abduqodirov A. A. Informatika va axborot texnologiyalari. – Toshkent: O‘zbekiston, 2011.

[10] Xayrullayev A. Dasturlash asoslari. – Toshkent: Fan va texnologiya, 2018.