PYTHON DASTURLASH TILIDA MASSIVLARNI TANLASH USULI BILAN SARALASHNI O‘QITISH METODIKASI

Fathiddinov Saidma'ruf Lazizxo‘ja o‘g‘li
Osiyo xalqaro universiteti "Aniq fanlar va raqamli texnologiyalar" kafedrasi o‘qituvchisi

Annotatsiya. Maqolada "Python dasturlash tili asoslari" kursida massivlarni (ro‘yxatlarni) tanlash usuli bilan saralash algoritmini o‘qitish metodikasi taklif etiladi. Metodika PRIMM (bashorat qil – ishga tushir – tekshir – o‘zgartir – yarat) yondashuviga tayanadi va besh bosqichdan iborat: kartochkalar yordamida algoritmni qo‘lda bajarish, kuzatuv jadvalini to‘ldirish va natijani bashorat qilish, kodni bosqichma-bosqich yozish, dastur bajarilishini vizualizatsiya qilish va avtomatik testlash, algoritm murakkabligini tahlil qilib boshqa usullar bilan taqqoslash. Maqolada algoritmning Python tilidagi namunaviy dasturi, bajarilish jadvali, talabalarda ko‘p uchraydigan xatolar va ularni bartaraf etish yo‘llari, saralash usullarining qiyosiy tavsifi hamda pedagogik tajriba rejasi bayon etilgan.

Kalit so‘zlar: Python, massiv, ro‘yxat, saralash, tanlash usuli bilan saralash, algoritm murakkabligi, PRIMM, kuzatuv jadvali, dasturlashni o‘qitish metodikasi.

KIRISH

Saralash algoritmlari dasturlashni o‘rganishning markaziy mavzularidan biridir: ular orqali talaba massiv bilan ishlash, ichma-ich sikllar, indekslar, shart operatorlari va algoritm samaradorligi tushunchalarini bir vaqtda o‘zlashtiradi hamda hisoblash tafakkurini rivojlantiradi [2–4]. Tanlash usuli bilan saralash (*selection sort*) g‘oyasining soddaligi – har bir qadamda saralanmagan qismdagi eng kichik elementni topib, uni o‘z o‘rniga qo‘yish – tufayli u odatda birinchi o‘rganiladigan saralash algoritmi hisoblanadi. "Raqamli O‘zbekiston – 2030" strategiyasi [1] yoshlarning dasturlash va raqamli ko‘nikmalarini rivojlantirishni ustuvor vazifa sifatida belgilaganini hisobga olsak, bunday asosiy algoritmlarni samarali o‘qitish metodikasini ishlab chiqish dolzarb masaladir.

Amaliyotda esa bir qator muammolar kuzatiladi. Birinchidan, Python tilida tayyor sorted() funksiyasi va list.sort() metodi mavjudligi sababli talabalar saralash algoritmini o‘rganishni ortiqcha deb hisoblaydi. Ikkinchidan, ko‘pchilik talabalar algoritmni darslikdagi namunaviy koddan yodlab oladi, ammo ichma-ich sikl har bir qadamda massivni qanday o‘zgartirishini kuzata olmaydi. Lister va hammualliflarining ko‘p mamlakatli tadqiqoti boshlovchi dasturchilarning sezilarli qismi kod bajarilishini kuzatish va uning natijasini oldindan aytishda qiynalishini ko‘rsatgan [7]. Sorva bu holatni talabada dastur bajarilishining to‘g‘ri mental modeli – "tushunchaviy mashina" (*notional machine*) shakllanmaganligi bilan izohlaydi [11].

Bu muammolarni hal qilishda bir necha yondashuvlar ma'lum: kompyutersiz (*unplugged*) mashg‘ulotlar algoritm g‘oyasini koddan oldin tushunishga yordam beradi [6]; Parsons masalalari sintaksis yukini kamaytiradi [8]; algoritm vizualizatsiyasi talaba uni faol boshqarganda samarali bo‘ladi [9, 10]; PRIMM yondashuvi esa kodni yozishdan oldin uni o‘qish va natijasini bashorat qilishni taklif etadi [5]. Biroq bu yondashuvlar odatda alohida-alohida qo‘llaniladi. *Tadqiqot maqsadi* – ularni tanlash usuli bilan saralash mavzusi misolida yaxlit metodikaga birlashtirish va uning samaradorligini baholash tizimini asoslash. *Ishchi gipoteza:* algoritmni qo‘lda bajarish, kuzatuv jadvali, bosqichma-bosqich kodlash va murakkablik tahlili izchil bog‘lansa, talabalarning algoritmni tushunish va kodni kuzatish ko‘nikmalari an'anaviy "tushuntirish – namunaviy kod – mashq" tartibiga nisbatan yuqori bo‘ladi.

© Ushbu material Creative Commons Attribution 4.0 International (CC BY 4.0) litsenziyasi asosida tarqatiladi.

ASOSIY QISM

Algoritm mohiyati. n ta elementdan iborat massiv ikki qismga – chap tomondagi saralangan va o‘ng tomondagi saralanmagan qismlarga ajratiladi. i-o‘tishda (i = 0, 1, …, n − 2) saralanmagan qism a[i..n−1] dagi eng kichik elementning indeksi topiladi va bu element a[i] bilan o‘rin almashtiriladi; shundan so‘ng saralangan qism bitta elementga kengayadi [2, 3]. Algoritmning Python tilidagi namunaviy dasturi 1-listingda keltirilgan.

1-listing. Tanlash usuli bilan saralash funksiyasi

```
def tanlash_saralash(a):
    """Massivni tanlash usuli bilan o'sish tartibida saralaydi."""
    n = len(a)
    for i in range(n - 1):            # i - saralanmagan qism boshi
        min_i = i                     # joriy minimum indeksi
        for j in range(i + 1, n):
            if a[j] < a[min_i]:
                min_i = j
        if min_i != i:                # almashtirish kerakmi?
            a[i], a[min_i] = a[min_i], a[i]
    return a

sonlar = [64, 25, 12, 22, 11]
print(tanlash_saralash(sonlar))       # [11, 12, 22, 25, 64]
```

Funksiyaning [64, 25, 12, 22, 11] massivi uchun bajarilishi 1-jadvalda ko‘rsatilgan (saralangan qism qalin harflarda berilgan). Aynan shu jadval darsda talaba bilan birgalikda to‘ldiriladi va metodikaning asosiy didaktik vositasi hisoblanadi.

1-jadval. [64, 25, 12, 22, 11] massivini saralashning kuzatuv jadvali

O‘tish (i)	Minimum (indeksi)	Almashtirish	O‘tishdan keyingi massiv	Taqqoslashlar
0	11 (4)	a[0] ↔ a[4]	[**11**, 25, 12, 22, 64]	4
1	12 (2)	a[1] ↔ a[2]	[**11, 12**, 25, 22, 64]	3
2	22 (3)	a[2] ↔ a[3]	[**11, 12, 22**, 25, 64]	2
3	25 (3)	bajarilmaydi	[**11, 12, 22, 25, 64**]	1
Jami		**3 ta**		**10**

Metodika bosqichlari. Metodika PRIMM yondashuviga [5] tayanadi, ikki amaliy mashg‘ulot va uyga vazifaga mo‘ljallangan hamda besh bosqichdan iborat: (1) *qo‘lda bajarish* – juftliklarga raqamlar yozilgan 6–8 ta kartochka beriladi; talaba “bir vaqtda faqat ikkita kartochkani solishtirish yoki ularning o‘rnini almashtirish mumkin” degan qoida asosida ularni tartiblaydi va o‘z harakatlarini so‘z bilan ifodalaydi, o‘qituvchi esa bu harakatlarni algoritm qadamlariga umumlashtiradi [6]; (2) *kuzatuv jadvali va bashorat* – talaba 1-listingdagi kodni ishga tushirmasdan, yangi massiv uchun 1-jadvalga o‘xshash jadvalni to‘ldiradi va har bir o‘tishdan keyingi massiv holatini, taqqoslashlar va almashtirishlar sonini oldindan aytadi; (3) *bosqichma-bosqich kodlash* – avval massivning berilgan qismidagi minimum indeksini qaytaruvchi yordamchi funksiya yoziladi, so‘ng u tashqi siklga joylashtiriladi; dasturlash

© Ushbu material Creative Commons Attribution 4.0 International (CC BY 4.0) litsenziyasi asosida tarqatiladi.

tajribasi kam talabalarga aralashtirilgan kod qatorlarini to‘g‘ri tartibga keltirish (Parsons masalasi) taklif etiladi [8]; (4) *vizualizatsiya va testlash* – kod Python Tutor muhitida [9] qadamma-qadam bajarilib, (2)-bosqichdagi bashorat bilan solishtiriladi, so‘ngra funksiya tasodifiy massivlarda sorted() natijasi bilan avtomatik tekshiriladi (2-listing); (5) *tahlil, o‘zgartirish va yaratish* – taqqoslashlar soni uchun umumiy formula keltirib chiqariladi, talaba funksiyani kamayish tartibida, satrlar ro‘yxati uchun yoki kalit (key) parametri bilan saralaydigan qilib o‘zgartiradi va natijani boshqa usullar bilan taqqoslaydi (2-jadval).

2-listing. Funksiyani avtomatik testlash

```
import random

for _ in range(1000):
    n = random.randint(0, 30)
    m = [random.randint(-100, 100) for _ in range(n)]
    assert tanlash_saralash(m.copy()) == sorted(m), m
print("Barcha testlar muvaffaqiyatli o'tdi")
```

Murakkablik tahlili. 1-jadvaldan ko‘rinadiki, i-o‘tishda $n - 1 - i$ ta taqqoslash bajariladi, demak taqqoslashlarning umumiy soni $(n - 1) + (n - 2) + \ldots + 1 = n(n - 1)/2$ ga teng; $n = 5$ uchun bu 10 ga mos keladi. Bu son massivning dastlabki holatiga bog‘liq emas: hatto saralangan massiv uchun ham algoritm barcha taqqoslashlarni bajaradi, shuning uchun uning vaqt murakkabligi barcha hollarda $O(n^2)$ ga teng. Almashtirishlar soni esa $n - 1$ dan oshmaydi – bu xususiyat yozish amali qimmat bo‘lgan hollarda tanlash usulining afzalligi hisoblanadi [2]. Talabalar bu xulosalarni funksiyaga taqqoslash va almashtirishlarni sanovchi hisoblagichlar qo‘shib, turli uzunlikdagi massivlarda tajribada tekshiradilar va massiv uzunligi ikki marta oshganda bajarilish vaqti taxminan to‘rt marta oshishini kuzatadilar.

Barqarorlik. Tanlash usuli barqaror (*stable*) emas, ya’ni teng kalitli elementlarning dastlabki tartibi saqlanmasligi mumkin. Masalan, [(2, "a"), (2, "b"), (1, "c")] ro‘yxati faqat birinchi element bo‘yicha saralanganda birinchi o‘tishdayoq (2, "a") va (1, "c") o‘rin almashadi va natija [(1, "c"), (2, "b"), (2, "a")] bo‘ladi. Bu misol Python dagi sorted() funksiyasining (Timsort algoritmi) barqarorligini va uning amaliy ahamiyatini tushuntirish uchun qulay asos bo‘ladi. Saralash usullarining asosiy xususiyatlari 2-jadvalda jamlangan.

2-jadval. Saralash usullarining qiyosiy tavsifi

Usul	Eng yaxshi holat	O‘rtacha va eng yomon holat	Almashtirishlar (eng ko‘p)	Barqarorlik
Tanlash usuli	$O(n^2)$	$O(n^2)$	$n - 1$	barqaror emas
Pufakcha usuli (to‘xtatish bayrog‘i bilan)	$O(n)$	$O(n^2)$	$n(n - 1)/2$	barqaror
Qo‘yish usuli	$O(n)$	$O(n^2)$	$n(n - 1)/2$ (siljitish)	barqaror
sorted() / list.sort() (Timsort)	$O(n)$	$O(n \log n)$	–	barqaror

© Ushbu material Creative Commons Attribution 4.0 International (CC BY 4.0) litsenziyasi asosida tarqatiladi.

Ko'p uchraydigan xatolar. Dasturlashni o'qitish amaliyotida ushbu mavzu bo'yicha quyidagi xatolar tez-tez uchraydi: (a) minimumning indeksi emas, qiymati saqlanadi va natijada almashtirish uchun element o'rni noma'lum bo'lib qoladi; (b) ichki sikl range(i + 1, n) o'rniga range(n) ko'rinishida yoziladi – bunda saralangan qism elementlari ham qayta ko'rib chiqiladi va natija noto'g'ri chiqadi; (c) almashtirish ichki sikl ichida bajariladi – dastur to'g'ri natija bersa-da, almashtirishlar soni keskin oshadi va algoritm aslida boshqa usulga aylanadi; (d) a[i], a[min_i] = a[min_i], a[i] o'rniga ketma-ket ikkita o'zlashtirish yoziladi va qiymatlardan biri yo'qoladi. Har bir xato uchun "xatoni top" topshirig'i tayyorlanadi: talabaga xatoli kod beriladi va u kuzatuv jadvali yordamida qaysi o'tishda massiv holati noto'g'ri bo'lishini aniqlaydi. Shuningdek, min_i = a.index(min(a[i:]), i) kabi qisqa "pythoncha" yozuv muhokama qilinadi: u ichki siklni yashiradi, ammo murakkablikni kamaytirmaydi va har bir o'tishda kesim nusxasini yaratadi. Bu misol "qisqa kod har doim tez kod emas" degan muhim xulosaga olib keladi.

Baholash va pedagogik tajriba rejasi. Tajriba "Python dasturlash tili asoslari" fani doirasida tajriba va nazorat guruhlarida o'tkazilishi rejalashtirilgan. Nazorat guruhida mavzu an'anaviy tartibda, tajriba guruhida esa taklif etilgan besh bosqichli metodika asosida o'tiladi. Har ikkala guruhda mashg'ulotdan oldin va keyin bir xil test o'tkaziladi; u kodni kuzatish, natijani bashorat qilish, xatoni topish va taqqoslashlar sonini hisoblash topshiriqlaridan iborat. Natijalar normallashtirilgan o'sish ko'rsatkichi g = (post − pre)/(100 − pre) [12] va Styudent t-mezoni yordamida taqqoslanadi. Amaliy topshiriqlar to'g'rilik (avtomatik testlar), kod sifati va himoya vaqtida tushuntirish mezonlari bo'yicha baholanadi.

Kutilayotgan natijalar. Metodikadan quyidagilar kutiladi: (1) kodni kuzatish va natijani bashorat qilish ko'nikmalarining yaxshilanishi, chunki talaba har bir o'tishni avval qo'lda bajaradi va jadvalga yozadi [5, 7]; (2) algoritm murakkabligini tayyor formula orqali emas, o'z hisob-kitobi va o'lchovi orqali tushunish; (3) tayyor sorted() funksiyasining afzalliklarini ongli baholash. Ushbu natijalar gipoteza bo'lib, ularni empirik ma'lumotlar bilan tasdiqlash pedagogik tajribaning vazifasidir.

Cheklovlar va ularni kamaytirish. Birinchidan, besh bosqich an'anaviy darsga nisbatan ko'proq vaqt talab qiladi, shuning uchun (1)–(2) bosqichlar birinchi mashg'ulotga, (3)–(5) bosqichlar ikkinchi mashg'ulot va uyga vazifaga ajratiladi. Ikkinchidan, sun'iy intellekt vositalari tayyor kodni oson berishi sababli baholashda asosiy urg'u kodning o'ziga emas, kuzatuv jadvali, "xatoni top" topshiriqlari va og'zaki himoyaga beriladi. Uchinchidan, talabalarning dasturlash darajasi turlicha bo'lgani uchun Parsons masalalari va juftlikda ishlash orqali topshiriqlar murakkabligi tabaqalashtiriladi.

XULOSA

Taklif etilgan metodika tanlash usuli bilan saralashni yodlanadigan kod sifatida emas, balki talaba qo'lda bajaradigan, kuzatadigan, dasturlaydigan, tekshiradigan va tahlil qiladigan algoritm sifatida o'rgatadi. Kuzatuv jadvali metodikaning markaziy vositasi bo'lib, u kartochkalar bilan bajarilgan harakatni Python kodi va n(n − 1)/2 murakkablik formulasi bilan bog'laydi. Metodikaning samaradorligi tajriba va nazorat guruhlarini taqqoslaydigan pedagogik tajriba, test va baholash mezonlari asosida aniqlanadi. Keyingi bosqichda qo'yish, birlashtirish va tezkor saralash usullari uchun ham shunday topshiriqlar to'plami va o'quv-uslubiy qo'llanma ishlab chiqish rejalashtirilgan.

FOYDALANILGAN ADABIYOTLAR

1. O'zbekiston Respublikasi Prezidentining 2020-yil 5-oktabrdagi PF-6079-son Farmoni. "Raqamli O'zbekiston – 2030" strategiyasini tasdiqlash va uni samarali amalga oshirish chora-tadbirlari to'g'risida.
2. Knuth D.E. The Art of Computer Programming. Vol. 3: Sorting and Searching. 2nd ed. – Reading, MA: Addison-Wesley, 1998. – 780 p.

© Ushbu material Creative Commons Attribution 4.0 International (CC BY 4.0) litsenziyasi asosida tarqatiladi.

3. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms. 4th ed. – Cambridge, MA: MIT Press, 2022. – 1312 p.
4. Wing J.M. Computational thinking // Communications of the ACM. – 2006. – Vol. 49, No. 3. – P. 33–35.
5. Sentance S., Waite J., Kallia M. Teaching computer programming with PRIMM: a sociocultural perspective // Computer Science Education. – 2019. – Vol. 29, No. 2–3. – P. 136–176.
6. Bell T., Witten I.H., Fellows M. CS Unplugged: An enrichment and extension programme for primary-aged students. – Christchurch: University of Canterbury, 2015.
7. Lister R., Adams E.S., Fitzgerald S., et al. A multi-national study of reading and tracing skills in novice programmers // ACM SIGCSE Bulletin. – 2004. – Vol. 36, No. 4. – P. 119–150.
8. Parsons D., Haden P. Parson's programming puzzles: a fun and effective learning tool for first programming courses // Proceedings of the 8th Australasian Conference on Computing Education. – 2006. – Vol. 52. – P. 157–163.
9. 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.
10. 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, No. 3. – P. 259–290.
11. Sorva J. Notional machines and introductory programming education // ACM Transactions on Computing Education. – 2013. – Vol. 13, No. 2. – Article 8.
12. Hake R.R. Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses // American Journal of Physics. – 1998. – Vol. 66, No. 1. – P. 64–74.

© Ushbu material Creative Commons Attribution 4.0 International (CC BY 4.0) litsenziyasi asosida tarqatiladi.