SARALASH ALGORITMLARI

[1]Xursandov Hamidullo
[2]Keldiyorova Zarinabonu
[1]SamDU Urgut filiali Tabiiy fanlarni o‘qitish metodikasi kafedrasi asissenti
[2]Samarqand davlat universiteti Urgut filiali talabasi
E-mail: bonukeldiyorova35@gmail.com

ANNOTATSIYA

Ushbu maqola dasturlashda eng muhim mavzulardan biri bo‘lgan saralashga bag‘ishlanadi. Ma'lumotlarni saralash yoki tartibga solish juda muhim, chunki tartibsiz ma'lumotlar bilan ishlash qiyinchilik tug‘diradi va bunday tizimlar sekin va xatolarga moyil bo‘ladi.

Kalit so‘zlar: *saralash, algoritm, massivda saralash, faylda saralash, elementlarni saralash, dasturchilar.*

АННОТАЦИЯ

Эта статья посвящена сортировке, одной из наиболее важных тем в программировании. Сортировка или организация данных важна, поскольку с неорганизованными данными трудно работать, а такие системы работают медленно и подвержены ошибкам.

Ключевые слова: *сортировка, алгоритм, сортировка массива, сортировка файлов, сортировка элементов, программисты.*

ABSTRACT

This article focuses on sorting, one of the most important topics in programming. Sorting or organizing data is important because disorganized data is difficult to work with and such systems are slow and prone to errors.

Key words: *sorting, algorithm, array sorting, file sorting, element sorting, programmers.*

KIRISH

Saralash - bu berilgan ma’lumot elementlarini ba'zi bir xususiyatlariga ko‘ra tartibga solish jarayonidir. Odatda, saralash mezoni sifatida aniq bir raqamli maydon (kalit) ishlatiladi. Elementlarni kalit bo‘yicha tartibga solish jarayonida har bir keyingi elementning kaliti oldingisidan kichik bo‘lsa kamayish tartibida, aksincha bo‘lsa o‘sish tartibida saralash amalga oshiriladi.

MUHOKAMA VA NATIJALAR

Saralashning asosiy maqsadi - saralangan ma’lumotlarni qayta ishlash jarayonida zarur bo‘ladigan elementlarni tez va oson topishni soddalashtirishdan iborat.

Saralash algoritmlari ikki asosiy guruhga bo‘linadi:

- Ichki saralash algoritmlari (massivda saralash)
- Tashqi saralash algoritmlari (faylda saralash)

Ichki saralash algoritmlari ma'lumotlarni tezkor xotirada saralaydi. Bu usullarda elementlarni qayta joylashtirish jarayoni tezkor xotiraning o‘zida amalga oshiriladi. Ichki saralash algoritmlari uch asosiy sinfga bo‘linadi:

- ✓ Qo‘yish orqali saralash
- ✓ Tanlash asosida saralash
- ✓ Almashtirish orqali saralash[1]

Misol uchun, telefon raqamlari yozilgan daftarchangiz bor (qo‘l telefonlari ommalashmagan davrni eslang) va nomlar tartibsiz yozilgan. Nodir ismli do‘stingizning raqamini topish uchun butun daftarchani tekshirib chiqishingiz kerak bo‘ladi, bu esa ko‘p vaqt talab etadi. Ammo, agar ular alifbo tartibida saralangani bo‘lsa, N harfiga o‘tib qidirishni boshlaysiz va vaqtni sezilarli darajada tejaysiz.

Saralash - bu berilgan ma’lumot elementlarini ba'zi bir xususiyatlariga ko‘ra tartibga solish jarayonidir. Odatda, saralash mezoni sifatida aniq bir raqamli maydon (kalit) ishlatiladi. Elementlarni kalit bo‘yicha tartibga solish jarayonida har bir keyingi elementning kaliti oldingisidan kichik bo‘lsa kamayish tartibida, aksincha bo‘lsa o‘sish tartibida saralash amalga oshiriladi.

Saralashning asosiy maqsadi - saralangan ma’lumotlarni qayta ishlash jarayonida zarur bo‘ladigan elementlarni tez va oson topishni soddalashtirishdan iborat.

Bu bo‘limda biz arraydagi ma'lumotlarni saralashni ko‘rib chiqamiz va saralash algoritmlarining olti turini o‘rganamiz:

Selection sort (Tanlab saralash)

Bubble sort (Pufakchali saralash)

Insertion sort (Joylashtirib saralash)

Quick sort (Tezkor saralash)

Merge sort (Qo‘shib saralash)

Radix sort

Elementlarni saralash – bu dasturchilar o‘rganishi kerak bo‘lgan asosiy algoritmlardan biridir. Ilgari informatika faniga kam e'tibor berilgan bo‘lsa-da, hozirda maktablarda saralash algoritmlarini tushunish va amalga oshirish muhimdir. Asosiy algoritmlar, odatda, for tsikli yordamida yoziladi. Massiv kabi elementlar

[1] Алгоритмы в современной мутематике и ее приложениях. Материалы международного симпозиума. Ургенч. 1979 г. Под редакцией А.П. Ершова и Д. Кнута. 8-с.

to‘plamini saralash uchun ushbu to‘plamni qandaydir tarzda kesib o‘tish kerak. [2]Masalan:

```
int[] array = {10, 2, 10, 3, 1, 2, 5};
for (int i = 0; i < array.length; i++) {
    System.out.println(array[i]);
}
```

```
sonlar = list(range(1, 11))
toqlar = []
juftlar = []
for son in sonlar:
    if son % 2 == 0:
        juftlar.append(son)
    else:
        toqlar.append(son)
print("Toqlar:", toqlar)
print("Juftlar:", juftlar)
```

[3]Ushbu kod qismini ko‘rib chiqaylik: int i qiymatini 0 dan massivning oxirgi elementiga qadar o‘zgartiradigan tsikl mavjud. Asosan, biz oddiygina massivdagi har bir elementni olib, uning mazmunini chop etamiz. Massivdagi elementlar qancha ko‘p bo‘lsa, kodning bajarilishi shunchalik uzoq davom etadi. Ya'ni, agar n elementlar soni bo‘lsa, n=10 bo‘lganda dastur n=5 bo‘lgandan ikki marta ko‘proq vaqtni oladi. Dasturimizda bitta tsikl bo‘lgani uchun bajarilish vaqti chiziqli ravishda oshadi: qancha ko‘p element bo‘lsa, bajarilish shunchalik uzoq davom etadi. Bu degani, yuqoridagi kod algoritmi chiziqli vaqtda (O(n)) ishlaydi. Bunday hollarda "algoritm murakkabligi" O(n) deb ataladi. Bu belgi "katta O" yoki "asimptotik xatti-harakatlar" deb ham ataladi. Siz shunchaki "algoritmning murakkabligini" eslab qolishingiz mumkin.

Shunday qilib, bizda massiv bor va uni takrorlashni bilamiz. Keling, uni o‘sish tartibida saralashga harakat qilaylik. Bu shuni anglatadiki, ikkita element (masalan, a=6, b=5) berilgan bo‘lsa, agar a b dan katta bo‘lsa (a > b bo‘lsa), a va b ni almashtirishimiz kerak. Massivda bu shunday amalga oshiriladi: agar array[i] > array[i - 1] bo‘lsa, elementlarni almashtirish kerak. Almashtirishni quyidagi kod yordamida amalga oshirish mumkin:

```
private void swap(int[] array, int ind1, int ind2) {
    int tmp = array[ind1];
```

[2] Амиров ва б. Ахборот-коммуникация технологиялари изоҳли луғати . БМТТДнинг Ўзбекистондаги ваколатхонаси, 2010. 12-б.

[3] Белов М.П. Основы алгоритмизации в информационных системах.Учебное пособие. СПб.:СЗТУ. 2003г.,26-с.

~~array[ind1] = array[ind2];~~
~~array[ind2] = tmp;~~
~~}~~
~~Endi quyidagicha yozish mumkin:~~
~~Int [ ] array = {10, 2, 10, 3, 1, 2, 5};~~
~~System.out.println(Arrays.toString(array));~~
~~for (int i = 1; i < array.length; i++) {~~
~~if (array[i] < array[i - 1]) {~~
~~swap(array, i, i-1);~~
~~}~~
~~}~~
~~System.out.println(Arrays.toString(array));~~

```
array = [10, 2, 10, 3, 1, 2, 5]
print(array)

for j in range(len(array)-1):
    for i in range(len(array)-1):
        if array[i] > array[i+1]:
            tmp = array[i]
            array[i] = array[i+1]
            array[i+1] = tmp

print(array)
```

Saralash algoritmlarining samaradorligini bir necha parametrlari bo‘yicha farqlash mumkin. Bu parametrlarning asosiylari quyidagilar hisoblanadi:

- saralash uchun sarflanadigan vaqt;
- saralash uchun talab qilinadigan tezkor xotira hajmi;

Saralash algoritmlarini baholashda faqat «joyida» saralash usullarini qarab chiqamiz, ya’ni saralash jarayoni uchun qo‘shimcha xotira zahirasi talab qilinmaydi. Saralash uchun sarf qilinadigan vaqtni esa, saralash bajarilishi jarayonida amalga oshiriladigan taqqoslashlar va o‘rin almashtirishlar soni orqali hisoblash mumkin. Ixtiyoriy saralash usulida taqqoslashlar soni O(nlog2n) dan O(n2) gacha bo‘lgan oraliqda yotadi.[4]

Ma’lumotlarni saralashning qat’iy (to‘g‘ri) va yaxshilangan usullari mavjud bo‘lib, qat’iy usullariga quyidagilarni misol qilib olish mumkin:

1) to‘g‘ridan-to‘g‘ri qo‘yish orqali saralash usuli;
2) to‘g‘ridan-to‘g‘ri tanlash orqali saralash usuli;
3) to‘g‘ridan-to‘g‘ri almashtirish orqali saralash usuli;

Bu uchala saralash usullarining samaradorligi deyarli bir xil.

[4] Успенский В.А., Семенов А.Л. Теория алгоритмов: основные открытия и приложения. М., Наука, 1987.15-с.

Tanlash Saralash

Tanlash saralash algoritmi - bu elementlarni tartibga solish uchun foydalaniladigan yana bir usul bo‘lib, u kvadratik murakkablikka ega. Tanlash saralash algoritmining asosiy g‘oyasi shundan iboratki, har bir o‘tishda eng kichik element topiladi va u boshiga o‘tkaziladi. Har bir yangi o‘tish keyingi elementdan boshlanadi. Quyidagi kod tanlash saralash algoritmini ifodalaydi:

```
int[] array = {10, 2, 10, 3, 1, 2, 5};
System.out.println(Arrays.toString(array));
for (int left = 0; left < array.length; left++) {
  int minInd = left;
  for (int i = left; i < array.length; i++) {
    if (array[i] < array[minInd]) {
      minInd = i;
    }
  }
  swap(array, left, minInd);
}
System.out.println(Arrays.toString(array));
Bu yerda swap funksiyasi elementlarni joylarini almashtiradi:
private void swap(int[] array, int ind1, int ind2) {
  int tmp = array[ind1];
  array[ind1] = array[ind2];
  array[ind2] = tmp;
}
```

```python
array = [10, 2, 10, 3, 1, 2, 5]
print(array)

for left in range(len(array)):
    min_ind = left
    for i in range(left, len(array)):
        if array[i] < array[min_ind]:
            min_ind = i
    array[left], array[min_ind] = array[min_ind], array[left]

print(array)
```

Tanlash saralash algoritmi beqaror, chunki bir xil elementlar o‘z o‘rnini o‘zgartirishi mumkin. Masalan, ikki bir xil qiymatli elementning tartibi o‘zgarishi mumkin. Bu algoritm $O(n^2)$ murakkablikka ega bo‘lib, tsikl ichida yana bir tsikl mavjud bo‘lganligi sababli, n element uchun n * n operatsiya bajariladi.

Saralash algoritmlarining samaradorligini bir necha parametrlari bo‘yicha farqlash mumkin. Bu parametrlarning asosiylari quyidagilar hisoblanadi:

- saralash uchun sarflanadigan vaqt;
- saralash uchun talab qilinadigan tezkor xotira hajmi;

Saralash algoritmlarini baholashda faqat «joyida» saralash usullarini qarab chiqamiz, ya’ni saralash jarayoni uchun qo‘shimcha xotira zahirasi talab qilinmaydi. Saralash uchun sarf qilinadigan vaqtni esa, saralash bajarilishi jarayonida amalga oshiriladigan taqqoslashlar va o‘rin almashtirishlar soni orqali hisoblash mumkin. Ixtiyoriy saralash usulida taqqoslashlar soni O(nlog2n) dan O(n2) gacha bo‘lgan oraliqda yotadi.

Ma’lumotlarni saralashning qat’iy (to‘g‘ri) va yaxshilangan usullari mavjud bo‘lib, qat’iy usullariga quyidagilarni misol qilib olish mumkin:

1) to‘g‘ridan-to‘g‘ri qo‘yish orqali saralash usuli;
2) to‘g‘ridan-to‘g‘ri tanlash orqali saralash usuli;
3) to‘g‘ridan-to‘g‘ri almashtirish orqali saralash usuli;

Bu uchala saralash usullarining samaradorligi deyarli bir xil.[5]

XULOSA

Tanlash va qo‘shish saralash algoritmlari har ikkalasi ham o‘ziga xos afzalliklarga ega. Tanlash saralash oson tushunarli va amalga oshiriladi, lekin beqaror. Qo‘shish saralash barqaror va elementlar tartiblangan yoki deyarli tartiblangan bo‘lsa yaxshi ishlaydi. Har ikki algoritm kvadratik murakkablikka ega bo‘lib, katta hajmdagi ma'lumotlar uchun samarali emas. Shu bilan birga, ular oddiy va o‘rganish uchun juda mos keladi.

FOYDALANILGAN ADABIYOTLAR (REFERENCES):

1. Алгоритмы в современной мутематике и ее приложениях. Материалы международного симпозиума. Ургенч. 1979 г. Под редакцией А.П. Ершова и Д. Кнута. 8-с.
2. Амиров ва б. Ахборот-коммуникация технологиялари изоҳли луғати. БМТТДнинг Ўзбекистондаги ваколатхонаси, 2010. 12-б.
3. Белов М.П. Основы алгоритмизации в информационных системах.Учебное пособие. СПб.:СЗТУ. 2003г.,26-с.
4. Успенский В.А., Семенов А.Л. Теория алгоритмов: основные открытия и приложения. М., Наука, 1987.15-с.
5. В.И.Игошин. Математическая логика и теория алгоритмов. Издательство Саратовского Университета, 1991. С.243.

[5] В.И.Игошин. Математическая логика и теория алгоритмов. Издательство Саратовского Университета, 1991. С.243.