O‘ZBEKISTON RESPUBLIKASI

OLIY TA’LIM, FAN VA INNOVATSIYALAR VAZIRLIGI

QARSHI DAVLAT TEXNIKA UNIVERSITETI

MAGISTRATURA BO‘LIMI

70610105 – TA‘LIMDA AXBOROT TEXNOLOGIYALARI MUTAXASISLIGI TAT 812-25 GURUH 1-KURS MAGISTRANTINING

“ALGORITMLARNI LOYIHALASHTIRISH VA TAHLIL QILISH” FANIDAN

6-AMALIY ISH

BAJARDI: **B NORMUMINOV**

QABUL QILDI: **Z UZAKOV**

Qarshi -2026

Mavzu: Saralash usullari. Pufakchali saralash. Kiritishli saralash. Tanlashli saralash. Tez saralash.

REJA

KIRISH

1. Saralash — umumiy tushunchalar

2. Pufakchali saralash (Bubble Sort)

3. Kiritishli saralash (Insertion Sort)

4. Tanlashli saralash (Selection Sort)

5. Tez saralash (Quick Sort)

6. To'rtta algoritmning umumiy qiyosi va qo'llanilishi

Xulosa

Foydalanilgan adabiyotlar

KIRISH

Mavzuning dolzarbligi. Axborot tizimlarida ma’lumotlarni tartibga solish — bu ma’lumotlar ustida bajariladigan eng ko‘p uchraydigan va fundamental amallardan biridir. Ma’lumotlar to‘g‘ri saralangan bo‘lsa, qidiruv tizimlari, ma’lumotlar bazalari va murakkab tahliliy dasturlar o‘nlab va hatto yuzlab marta tezroq ishlaydi. Bugungi kunda ma’lumotlar oqimining beqiyos darajada o‘sib borishi sharoitida, har bir konkret holat uchun eng samarali saralash usulini tanlash muhim ahamiyatga ega. Pufakchali, kiritishli va tanlashli saralash kabi klassik usullar algoritmlash asoslarini tushunishga yordam bersa, tez saralash (Quick Sort) kabi murakkab algoritmlar yuqori yuklamali tizimlarda vaqt resurslarini tejash imkonini beradi. Shu bois, saralash algoritmlarining mexanizmlari va ularning samaradorlik ko‘rsatkichlarini o‘rganish o‘ta dolzarbdir.

Muammoning qo‘yilishi. Algoritmlarni amaliyotda qo‘llashda asosiy muammo — bu ma’lumotlar hajmi ortishi bilan algoritmning ishlash vaqti va kompyuter xotirasiga bo‘lgan talabning keskin o‘zgarishidir. Masalan, sodda saralash usullari (pufakchali yoki tanlashli) kichik massivlarda yaxshi ishlasa-da, katta ma’lumotlar to‘plamida ular tizimning ishlashini sezilarli darajada sekinlashtiradi. Aksincha, tez saralash algoritmi yuqori tezlikka ega bo‘lsa-da, u ma’lum bir "yomon holatlarda" samarasiz bo‘lib qolishi mumkin. Asosiy muammo — ma’lumotlarning dastlabki holati va hajmidan kelib chiqib, eng kam solishtirish va elementlarni o‘rin almashtirish sonini talab qiluvchi optimal algoritmni aniqlashdan iborat.

Ilmiy ishning maqsadi va vazifalari. Mazkur ishning asosiy maqsadi turli xil saralash algoritmlarining ishlash prinsiplarini tadqiq qilish va ularning vaqt hamda xotira bo‘yicha samaradorligini qiyosiy tahlil qilishdan iborat. Ushbu maqsadga erishish uchun quyidagi vazifalar belgilab olindi:

- Pufakchali, kiritishli va tanlashli saralash algoritmlarining mantiqiy ketma-ketligini yoritish;

- Tez saralash (Quick Sort) algoritmining "bo‘lib tashla va hukmronlik qil" prinsipiga asoslangan mexanizmini o‘rganish;
- Har bir algoritmning eng yaxshi, o‘rtacha va eng yomon holatlardagi vaqt murakkabligini solishtirish;
- Ma’lumotlar hajmi va tartiblanganlik darajasiga qarab qaysi algoritm afzalroq ekanligi bo‘yicha tavsiyalar ishlab chiqish.

Ilmiy va amaliy ahamiyati. Tadqiqot natijalari dasturiy ta’minot ishlab chiquvchilar va ma’lumotlar muhandislari uchun tizimlarni optimallashtirishda uslubiy yordam beradi. Turli saralash usullarining kuchli va kuchsiz tomonlarini bilish resurslardan tejamli foydalanishga, dasturlarning ishlash vaqtini qisqartirishga va foydalanuvchi interfeyslarining tezkorligini oshirishga xizmat qiladi. Shuningdek, ushbu tahliliy yondashuv murakkab tizimlar arxitekturasini loyihalashda to‘g‘ri qarorlar qabul qilish uchun asos bo‘ladi.

Saralash (sorting) — massiv yoki ro'yxat elementlarini **ma'lum tartibda** (o'sish yoki kamayish bo'yicha) joylashtirish jarayoni. Saralash kompyuter fanining **eng ko'p o'rganilgan** masalalaridan biri — har yili milliardlab marta bajariladi.

Saralash algoritmlarini bilish nima uchun muhim? Birinchidan, **binar qidiruv** tartiblangan massivda ishlaydi. Ikkinchidan, **ko'plab algoritmlar** (MST, Dijkstra, konveks qobiq) avval saralashni talab qiladi. Uchinchidan, saralash **asimptotik tahlil va algoritmik paradigmalarni** tushunishning eng yaxshi platformasidir.

Ushbu amaliy ish to'rtta klassik saralash algoritmini — **Pufakchali, Kiritishli, Tanlashli va Tez saralash** — chuqur o'rganadi.

1-rasm. 7 ta saralash algoritmi — murakkablik, xotira va barqarorlik qiyosi

1. Saralash — umumiy tushunchalar

1.1 Saralash algoritmiga qo'yiladigan talablar

Saralash algoritmini baholashda **to'rtta asosiy mezon** hisobga olinadi:

- **Vaqt murakkabligi:** eng yaxshi, o'rtacha, eng yomon holatlardagi asimptotik murakkablik.
- **Xotira murakkabligi:** qo'shimcha xotira sarfi. In-place algoritmlar O(1) sarflaydi.
- **Barqarorlik (Stability):** teng elementlar o'z nisbiy tartibini saqlab qoladimi?
- **Adaptivlik:** kirish ma'lumotlarining tartiblanganligi algoritmni tezlashtiradimi?

1.2 Barqarorlik tushunchasi

Barqaror saralash (stable sort) — **kalit qiymatlari teng** bo'lgan elementlar **kirishdagi nisbiy tartibini** chiqishda ham saqlaydi. Bu ma'lumotlar bazasida **ko'p maydon bo'yicha saralashda** juda muhim.

Misol: Avval familiya bo'yicha, keyin ism bo'yicha saralashda — **barqaror algoritm** familiyalarning avvalgi tartibini buzishni oldini oladi.

Algoritm	Barqarormi?	In-place?	Eslatma
Pufakchali saralash	Ha (barqaror)	Ha	Teng elementlar almashtirılmaydi
Kiritishli saralash	Ha (barqaror)	Ha	Teng = chapga siljimaydi
Tanlashli saralash	Yo'q	Ha	Uzoqdagi element almashtiriladi
Tez saralash	Yo'q (Lomuto)	Ha	Pivot tanlash ta'sir qiladi
Qo'shib saralash	Ha (barqaror)	Yo'q (O(n))	Barqaror + O(n log n) ideal
Uyum saralash	Yo'q	Ha	Heap tuzilishi barqarorlikni buzadi

1.3 Solishtirishga asoslangan saralash pastki chegarasi

Muhim nazariy natija: **solishtirishga asoslangan** har qanday saralash algoritmi eng yomon holda **Ω(n log n) solishtirish** bajarishi kerak. Bu **qaror daraxti** (decision tree) modeli orqali isbotlanadi.

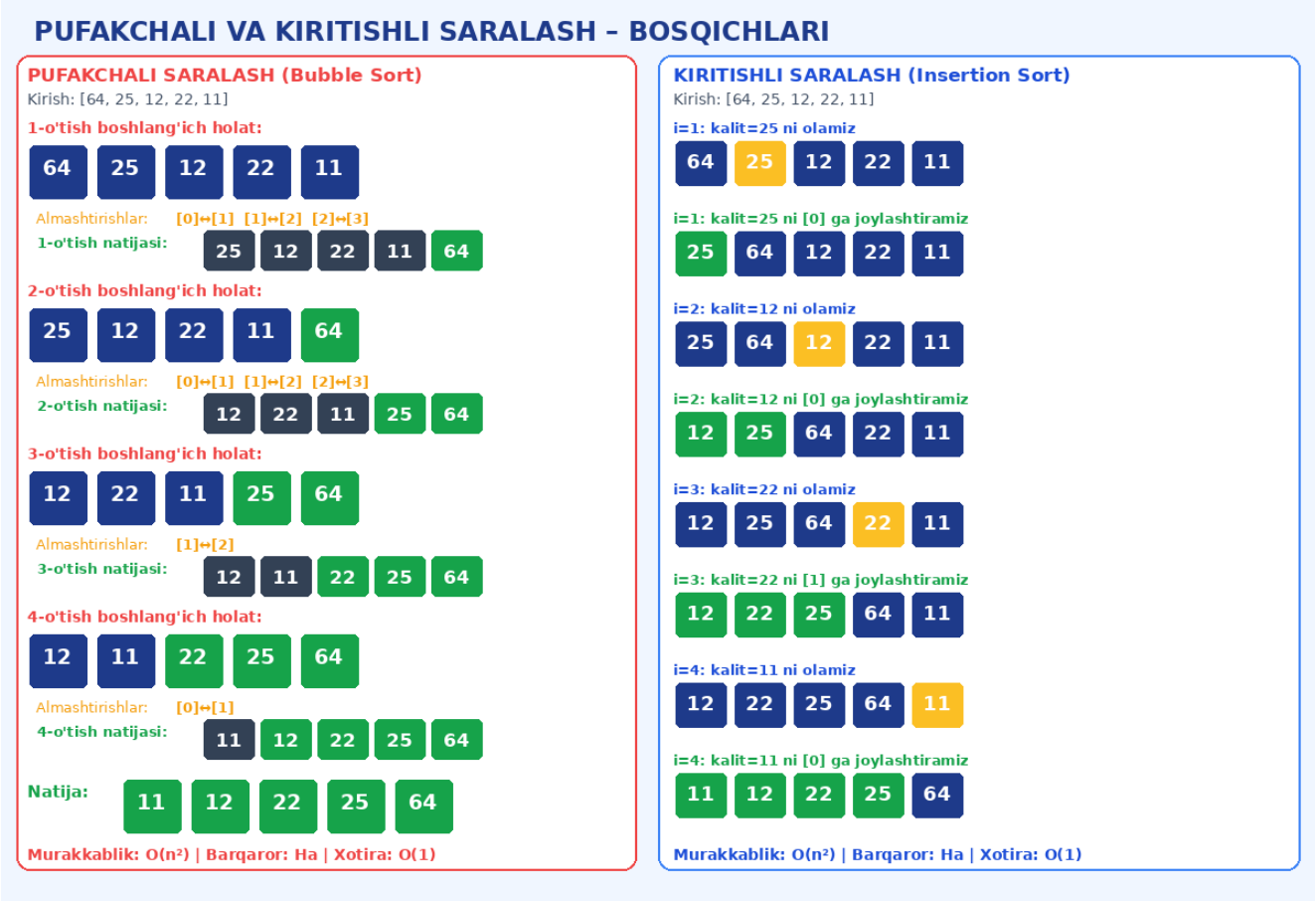
n ta element uchun n! ta mumkin joylashuv bor

Qaror daraxti chuqurligi ≥ $\log_2(n!) \approx n \cdot \log_2 n - n \cdot \log_2 e = \Omega(n \log n)$

Demak, **Merge Sort va Heap Sort** — solishtirishga asoslangan algoritmlar orasida **asimptotik jihatdan optimal**. Faqat **taqqoslamasiz algoritmlar** (Counting, Radix, Bucket Sort) bu chegarani buzishi mumkin.

2. Pufakchali saralash (Bubble Sort)

Pufakchali saralash — eng oddiy va intuitiv saralash usuli. Qo'shni elementlar taqqoslanib, **kattasi o'ngga** suriladi. Har "o'tish" da eng katta element **"pufak" kabi** massiv oxiriga suzib chiqadi — shu sababli algoritm bu nom bilan atalgan.



2-rasm. Pufakchali va kiritishli saralash — har o'tish bosqichlari (kirish: [64,25,12,22,11])

2.1 Ishlash printsipi

Algoritm qadamlari: 1) Massivning boshidan oxirigacha ketma-ket juftlarni solishtir. 2) Agar ***A[j] > A[j+1]*** bo'lsa, ularni almashtir. 3) Har ***i***-chi o'tishda

massivning oxiridagi ***i*** ta element **to'g'ri joyida**. 4) Agar bir o'tishda biror almashtirish bo'lmasa — massiv **tartiblangan, erta tugatish**.

Solishtirish soni: n(n-1)/2 (eng yomon)

Almashtirish soni: ≤ n(n-1)/2

```
// Pufakchali saralash — optimallashtirilgan
function bubbleSort(A, n):
    for i = 0 to n-2:
        swapped = false
        for j = 0 to n-2-i:
            if A[j] > A[j+1]:
                swap(A[j], A[j+1])
                swapped = true
        // Erta tugatish: O(n) eng yaxshi holat
        if not swapped: break

// Vaqt: O(n²) o'rtacha/eng yomon, O(n) eng yaxshi
// Xotira: O(1) — in-place
// Barqarorlik: Ha — A[j]>A[j+1] (≥ emas)
```

2.2 Murakkablik tahlili

Holat	Kirish	Vaqt	Izoh
Eng yaxshi	Tartiblangan massiv	**O(n)**	1 o'tish, 0 almashtirish, break
O'rtacha	Tasodifiy massiv	**O(n²)**	~n²/4 solishtirish
Eng yomon	Teskari tartiblangan	**O(n²)**	n(n-1)/2 solishtirish, almashtirish

Amaliy foydalanish: Pufakchali saralash asosan **ta'lim maqsadlarida** ishlatiladi. Amaliy dasturlashda **kiritishli saralash** kichik massivlar uchun yaxshiroq, katta massivlar uchun esa **tez saralash yoki qo'shib saralash** tanlanadi.

3. Kiritishli saralash (Insertion Sort)

Kiritishli saralash — **karta o'yinida qo'lga karta olish** usuliga o'xshaydi: har yangi karta qo'ldagi tartiblangan kartalar orasiga **to'g'ri joyiga** qo'yiladi. Massiv **chapdan o'ngga** ko'rib chiqilib, har element **chap tartiblangan qismga** joylashtiriladi.

3.1 Ishlash printsipi

Algoritm qadamlari: 1) i=1 dan boshlab, ***key = A[i]*** olinadi (kalit). 2) ***j = i-1*** dan chapga qarab, ***A[j] > key*** bo'lgan elementlar **bir o'ringa o'ngga** suriladi. 3) Kalit **bo'sh o'ringa** joylashtiriladi. 4) i=1 dan n-1 gacha takrorlanadi.

Invariant: A[0..i-1] har doim tartiblangan

```
// Kiritishli saralash
function insertionSort(A, n):
    for i = 1 to n-1:
        key = A[i]          // Joriy element — 'kalit'
        j = i - 1

        // Kattalarni o'ngga siljitish
        while j >= 0 and A[j] > key:
            A[j+1] = A[j]   // Siljitish (almashtirish emas!)
            j = j - 1

        A[j+1] = key        // Kalitni to'g'ri joyga qo'y
```

```
// Vaqt: O(n) eng yaxshi, O(n²) o'rtacha/eng yomon
// Xotira: O(1) — in-place
// Barqarorlik: Ha
// Kichik massivlar (n ≤ 32) uchun juda samarali
```

3.2 Murakkablik va afzalliklari

Xususiyat	Kiritishli saralash	Pufakchali saralash
Eng yaxshi vaqt	O(n) — tartiblangan kirish	O(n) — tartiblangan kirish
O'rtacha vaqt	$O(n^2)$ — $n^2/4$ solishtirish	$O(n^2)$ — $n^2/4$ solishtirish
Eng yomon vaqt	$O(n^2)$ — teskari kirish	$O(n^2)$ — teskari kirish
Amaliy tezlik	Pufakchali dan 2x tez	Nisbatan sekin
Siljitish vs almashtirish	Siljitish (1 yozuv)	Almashtirish (3 yozuv)
Kichik n uchun	Juda yaxshi ($n \leq 32$)	Qoniqarli
Online algoritm	Ha — elementlar qo'shilganda	Yo'q

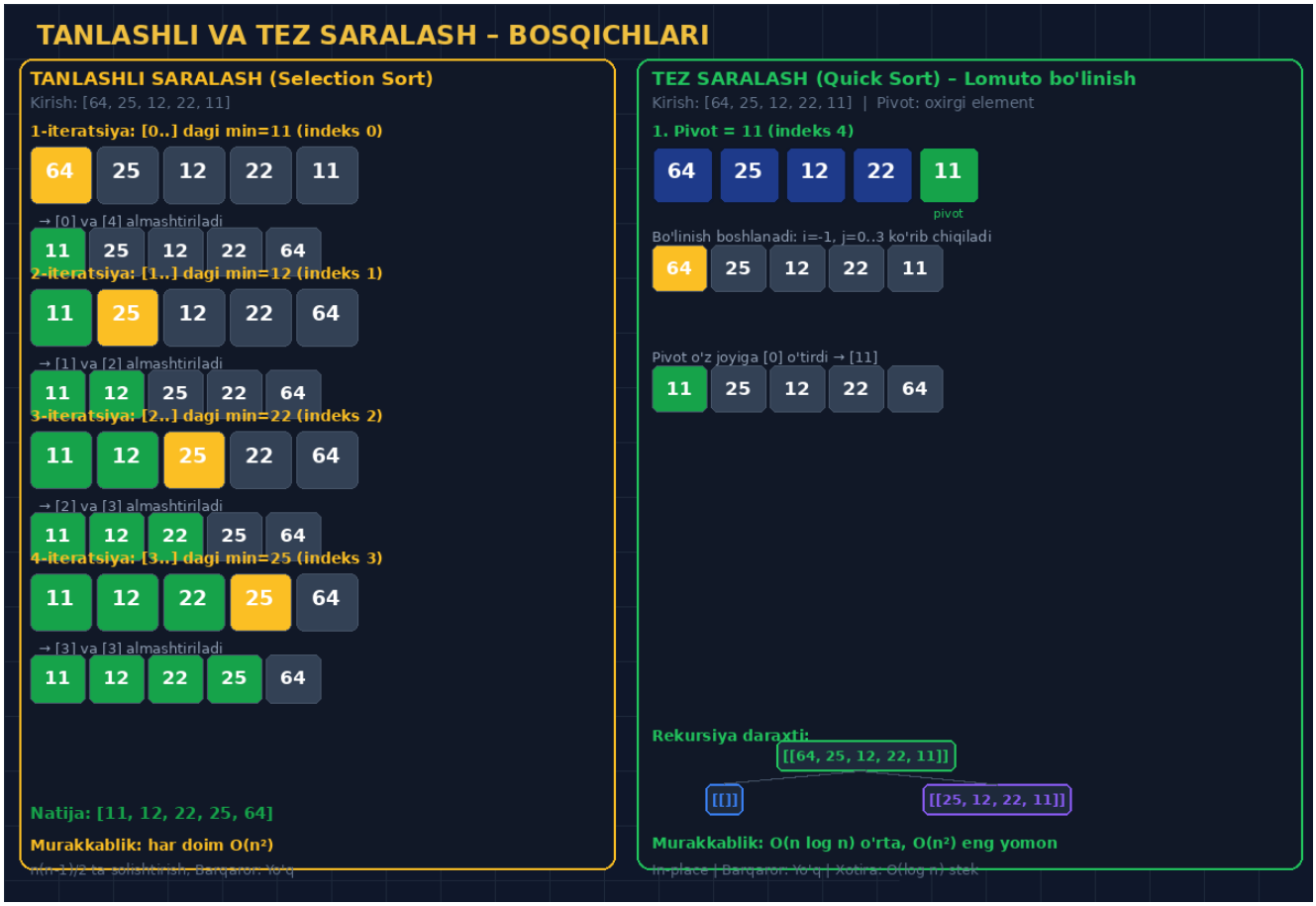
Online algoritm xususiyati: Kiritishli saralash **stream (oqim) ma'lumotlar** bilan ishlaydi — barcha elementlarni oldindan bilmasdan, **har yangi element kelganda** tartiblash davom ettiriladi. Bu **real vaqt tizimlari** uchun qimmatli xususiyat.

Tibbiy foydalanish: Amaliy dasturlarda **$n \leq 32$** bo'lganda kiritishli saralash **Tez saralashdan ham tez**. Shu sababli ko'p standart kutubxonalar (Java, C++ STL) kichik massivlar uchun kiritishli, kattalar uchun **"introsort"** (QuickSort +

HeapSort + InsertionSort aralash) ishlatadi.

4. Tanlashli saralash (Selection Sort)

Tanlashli saralash — har iteratsiyada **tartiblashlanmagan qismning eng kichik elementini** topib, uni **tartiblangan qism oxiriga** qo'yadi. Algoritm nomi "tanlash" dan kelib chiqqan — har safar **minimal element tanlanadi**.



3-rasm. Tanlashli va tez saralash — bosqichlari va rekursiya daraxti

4.1 Ishlash printsipi

Algoritm qadamlari: 1) ***i = 0*** dan boshlab, ***A[i..n-1]*** dagi minimal elementning indeksini top. 2) ***A[i]*** va ***A[min_idx]*** ni almashtir. 3) ***i*** ni 1 ga oshir, takrorla.

Har iteratsiyada: (n-1-i) ta solishtirish

Jami: (n-1) + (n-2) + ... + 1 = n(n-1)/2 solishtirish — har doim!

```
// Tanlashli saralash
function selectionSort(A, n):
    for i = 0 to n-2:
        min_idx = i

        // [i+1..n-1] da minimal topish
        for j = i+1 to n-1:
            if A[j] < A[min_idx]:
                min_idx = j

        // Minimal ni o'z joyiga almashtirish
        if min_idx != i:
            swap(A[i], A[min_idx])

// Vaqt: har doim O(n²) — adaptiv EMAS
// Xotira: O(1) — in-place
// Barqarorlik: Yo'q (uzoq almashtirish barqarorlikni buzadi)
// Almashtirish soni: ≤ n-1 (eng kam!)
```

4.2 Tanlashli saralashning o'ziga xos xususiyati

Eng kam almashtirish: Tanlashli saralash **maksimum n-1 ta almashtirish** bajaradi — bu **barcha O(n²) algoritmlar orasida eng oz**. Agar **yozish operatsiyasi qimmat** bo'lsa (masalan, EEPROM, SSD), bu afzallik muhim bo'ladi.

Barqaror emas: Tanlashli saralash **barqaror emas**, chunki minimal element **uzoqdagi element bilan almashtiriladi**, bu esa teng elementlar tartibini o'zgartirib yuborishi mumkin. *Misol: **[3a, 3b, 1]** → **[1, 3b, 3a]*** — 3a va 3b tartib o'zgardi!

Mezon	Bubble	Insertion	Selection
Solishtirish soni	n(n-1)/2	~n²/4 (avg)	n(n-1)/2 (har doim)
Almashtirish soni	~n²/4	~n²/4	≤ n-1 (minimal!)
Eng yaxshi vaqt	O(n)	O(n)	O(n²)
Adaptivlik	Ha	Ha	Yo'q
Barqarorlik	Ha	Ha	Yo'q

5. Tez saralash (Quick Sort)

Tez saralash — 1959-yilda **Tony Hoare** tomonidan ixtiro qilingan. Amalda **eng tez saralash algoritmi** hisoblanadi — kesh samaradorligi va past konstanta tufayli **Merge Sort dan ham tez** ishlaydi (garchi murakkablik bir xil).

Asosiy g'oya (Divide & Conquer): 1) **Pivot** (tayanch) element tanla. 2) **Bo'linish (Partition):** pivotdan kichiklar chapga, kattalar o'ngga. 3) **Rekursiv** ravishda ikkala qismni saralash.

5.1 Bo'linish usullari

Lomuto bo'linishi

Oxirgi element pivot. **i indeks** kichik elementlar chegarasini ko'rsatadi.

```
// Lomuto bo'linishi
function partition_lomuto(A, lo, hi):
    pivot = A[hi]     // Oxirgi element — pivot
    i = lo - 1        // Kichiklar chegarasi

    for j = lo to hi-1:
        if A[j] <= pivot:
            i = i + 1
```

```
            swap(A[i], A[j])   // Kichikni chapga

    swap(A[i+1], A[hi])  // Pivotni o'z joyiga
    return i + 1         // Pivot indeksi

// Murakkablik: O(n) har safar
```

Hoare bo'linishi (original, tezroq)

```
// Hoare bo'linishi
function partition_hoare(A, lo, hi):
    pivot = A[(lo+hi)//2]  // O'rta element
    i = lo - 1
    j = hi + 1

    while True:
        i = i + 1
        while A[i] < pivot: i = i + 1
        j = j - 1
        while A[j] > pivot: j = j - 1
        if i >= j: return j
        swap(A[i], A[j])

// Hoare: ~3x kam almashtirish Lomuto dan
```

5.2 Tez saralash to'liq kodi

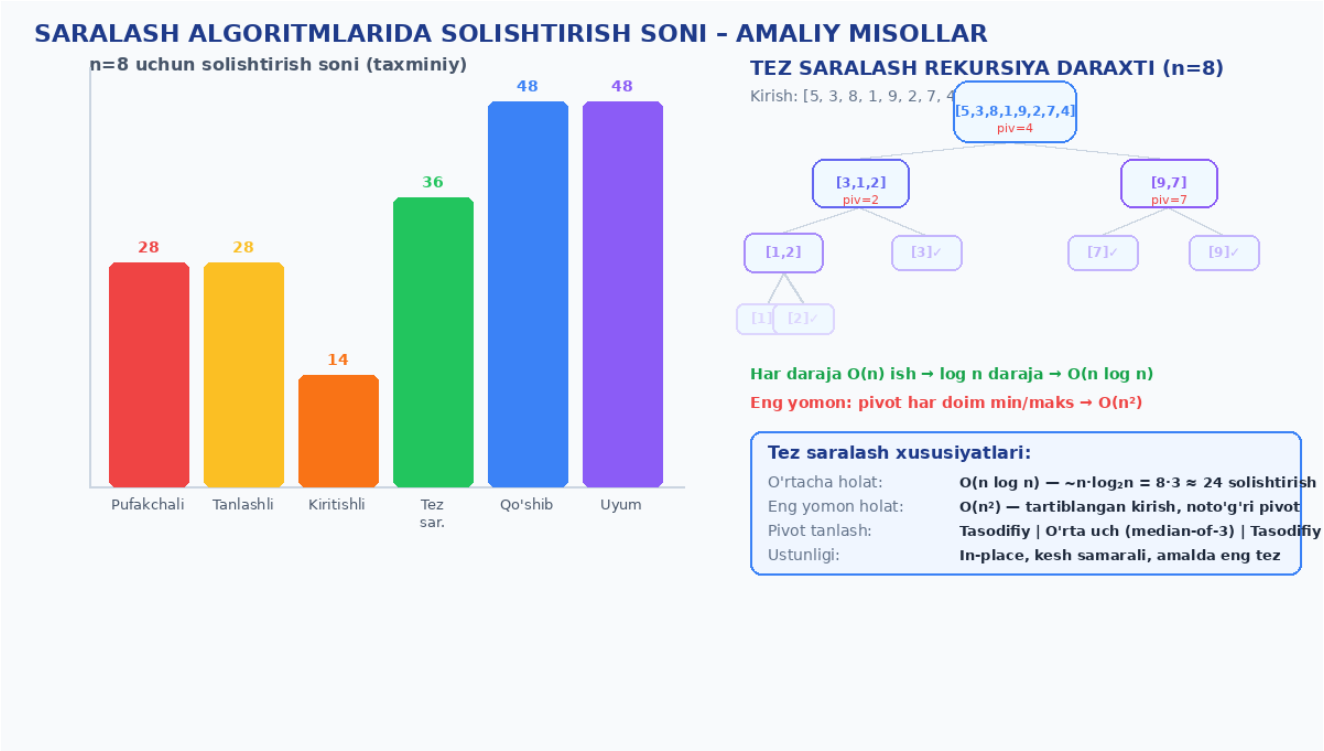
```
// Quick Sort — to'liq kod
```

```
function quickSort(A, lo, hi):
    if lo < hi:
        p = partition_lomuto(A, lo, hi)  // Bo'linish
        quickSort(A, lo, p-1)            // Chap qism
        quickSort(A, p+1, hi)            // O'ng qism

// Chaqirish: quickSort(A, 0, n-1)

// Vaqt: O(n log n) o'rtacha, O(n²) eng yomon
// Xotira: O(log n) rekursiya steki (o'rtacha)
// Barqarorlik: Yo'q
// In-place: Ha
```

5.3 Pivot tanlash strategiyalari

Tez saralashning samaradorligi **pivot tanloviga** bog'liq:

Pivot strategiya	Murakkablik	Xususiyat
Birinchi element	$O(n^2)$ tartiblangan kirish	Eng yomon strategiya
Oxirgi element (Lomuto)	$O(n^2)$ tartiblangan kirish	Keng qo'llaniladi, tushunish oson
Tasodifiy element	O(n log n) ehtimol bilan	Eng yomon holat ehtimoli minimallashadi
Median-of-3	O(n log n) ko'p hollarda	3 elementdan o'rtachasi — amalda eng yaxshi
Median-of-medians	O(n log n) kafolatli	Murakkab, katta n uchun



4-rasm. Saralash algoritmlari solishtirish soni (n=8) va tez saralash rekursiya daraxti

5.4 Tez saralash murakkablik tahlili

O'rtacha holat: O(n log n) — tasodifiy kirish uchun yoki tasodifiy pivot bilan. Rekurrens munosabat:

$$T(n) = 2T(n/2) + O(n) \rightarrow T(n) = O(n \log n) \text{ [tengga bo'linish]}$$

Eng yomon holat: O(n²) — pivot har safar minimal/maksimal bo'lsa (tartiblangan kirish + oxirgi element pivot). Rekurrens:

$$T(n) = T(n-1) + T(0) + O(n) = T(n-1) + O(n) \rightarrow T(n) = O(n^2)$$

Nima uchun amalda eng tez? Garchi murakkablik Merge Sort bilan bir xil (O(n log n)), Quick Sort **kesh samarali**: elementlarga ketma-ket murojaat qiladi. Merge Sort esa **qo'shimcha xotira** va ko'proq xotira murojaat talab qiladi. Shu sababli amalda Quick Sort **1.5–2x tez** ishlaydi.

6. To'rtta algoritmning umumiy qiyosi va qo'llanilishi

Mezon	Bubble	Insertion	Selection	Quick
Eng yaxshi	O(n)	O(n)	$O(n^2)$	O(n log n)
O'rtacha	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(n log n)
Eng yomon	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
Xotira	O(1)	O(1)	O(1)	O(log n)
Barqaror	Ha	Ha	Yo'q	Yo'q
In-place	Ha	Ha	Ha	Ha
Adaptiv	Ha	Ha	Yo'q	Yo'q
Almashtirish	Ko'p	O'rta	Eng kam	O'rta
Qachon ishlatish	Ta'lim	Kichik n	Kam yozuv	Umumiy maqsad

Amaliy tavsiyalar:

- **$n \leq 10$:** Insertion Sort — kichik massivlar uchun eng tez (cache warm, past konstanta).
- **$n \leq 1000$:** Insertion Sort yoki Selection Sort — oddiy, to'g'ri yozish oson.
- **$n > 1000$, umumiy:** Quick Sort — amalda eng tez, tizimlarda standart.
- **Barqarorlik kerak:** Merge Sort yoki Timsort (Python, Java default).
- **Xotira cheklangan:** Heap Sort — O(1) xotira + O(n log n) kafolatlangan.
- **Ma'lumot diapazon cheklangan:** Counting/Radix Sort — O(n) vaqt.

Xulosa

Ushbu amaliy ish to'rtta klassik saralash algoritmini — **Pufakchali, Kiritishli, Tanlashli va Tez saralash** — chuqur tahlil qildi. Asosiy xulosalar:

1. **Pufakchali saralash** eng oddiy lekin **amalda eng samarasiz**. Tartiblangan kirishda O(n) (adaptiv). **Faqat ta'lim maqsadlarida** qo'llaniladi.
2. **Kiritishli saralash** — **kichik massivlar (n ≤ 32) uchun ideal**. Online, barqaror, adaptiv. Standart kutubxonalar ichida **Tez saralash ichida** kichik qismlar uchun ishlatiladi.
3. **Tanlashli saralash** — har doim ***O(n²)***, **adaptiv emas**. Lekin **eng kam almashtirish** (≤ n-1) — yozish qimmat bo'lganda foydali.
4. **Tez saralash** — amalda **eng tez saralash algoritmi**. Kesh samarador, in-place. **Tasodifiy pivot yoki Median-of-3** bilan eng yomon holat ehtimoli minimallashadi.
5. Solishtirishga asoslangan saralash uchun **Ω(n log n) quyi chegara** isbotlangan — **Merge Sort va Heap Sort** bu chegaraga yetadi, Quick Sort esa amalda undan ham tez ishlaydi.

Foydalanilgan adabiyotlar

1. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. Introduction to Algorithms (CLRS). 4th Ed. MIT Press, 2022.

2. Knuth, D.E. The Art of Computer Programming, Vol. 3: Sorting and Searching. 2nd Ed. Addison-Wesley, 1998.

3. Hoare, C.A.R. Quicksort. The Computer Journal, 5(1), 10–16. 1962.

4. Sedgewick, R., Wayne, K. Algorithms. 4th Ed. Addison-Wesley, 2011.

5. Skiena, S.S. The Algorithm Design Manual. 3rd Ed. Springer, 2020.

6. Bentley, J.L., McIlroy, M.D. Engineering a Sort Function. Software: Practice and Experience, 23(11). 1993.