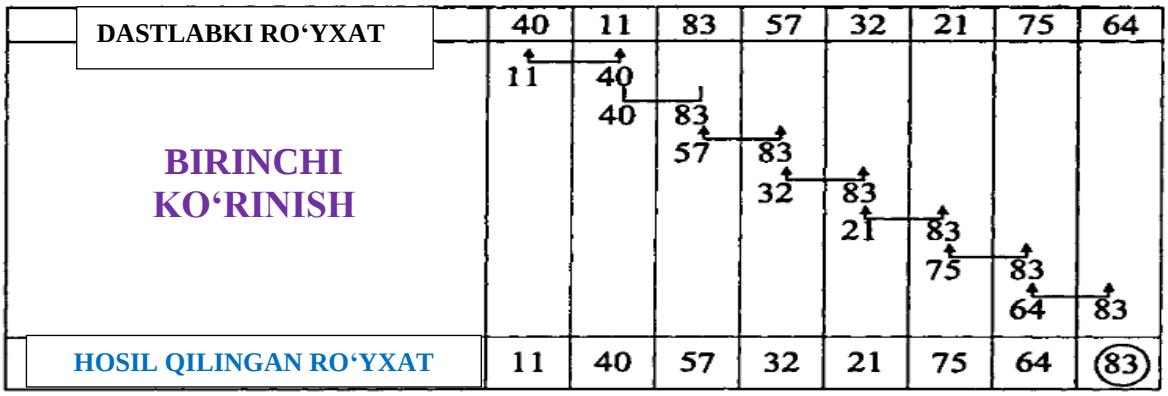
## 7-MA’RUZA. ALMASHISH USULIDA SARALASH, SARALASHNING SHEYKER USULI

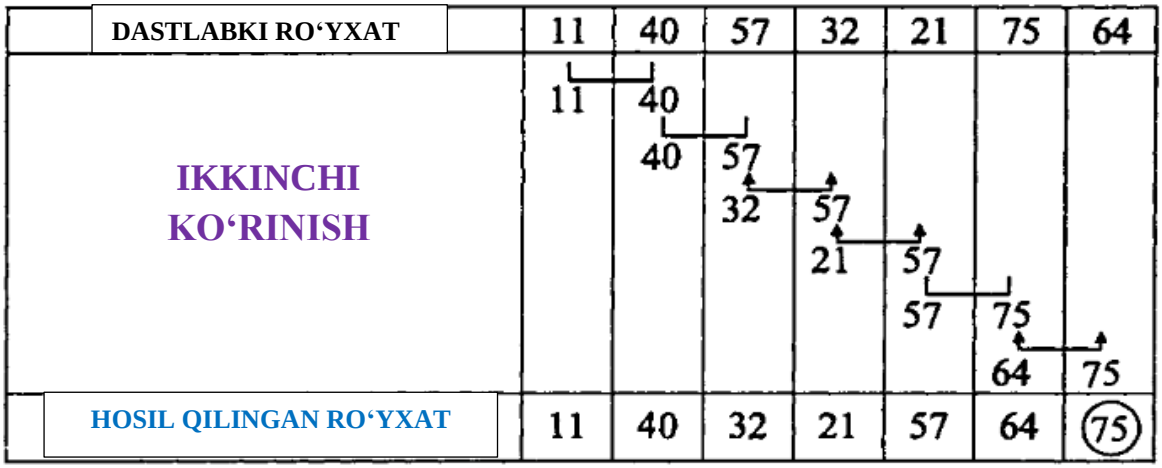
**Almashish usulida saralash.** Almashish usulida saralash – bu ro‘yhat elementlari ketma-ket o‘zaro taqqoslanadi va avvalgi element keyingi elementdan katta bo‘lgan holda joyini almashtiradi.

**7.1-misol.** Ro‘yxatni standart almashish usulida saralash talab qilinsin: **{40, 11, 83, 57, 32, 21, 75, 64}.**




**7.1-rasm. Almashish usulida saralash (birinchi ko‘rinish).**

Almashtiriladigan elementlarni ↑ ↑ strelkali kvadrat qavslar orqali, taqqoslanayotgan elementlarni esa ⌊ ⌋ kvadrat qavslar orqali belgilaymiz. Saralashning birinchi bosqichi 7.1-rasmda, ikkinchi bosqichi esa 7.2-rasmda ko‘rsatilgan. Har bir ro‘yhatni ko‘rishdan so‘ng, oxiridan boshlab barcha elementlar o‘zlarining oxirgi pozisiyalarini egallashini ko‘rish qiyin emas.




**7.2-rasm. Almashish usulida saralash (birinchi ko‘rinish).**

Har bir keyingi ko‘rib chiqish eng katta element topgan pozitsiyani yo‘qotib borib, ro‘yhatni qisqartirib boradi. Birinchi ko‘rib chiqishdan so‘ng oxirgi pozitsiyada 83 ga teng eng katta element qoldi.

Ikkinchi ko‘rib chiqishdan eng katta element 75 ga teng ekenligini kelib chiqadi (7.2-rasmga qarang).

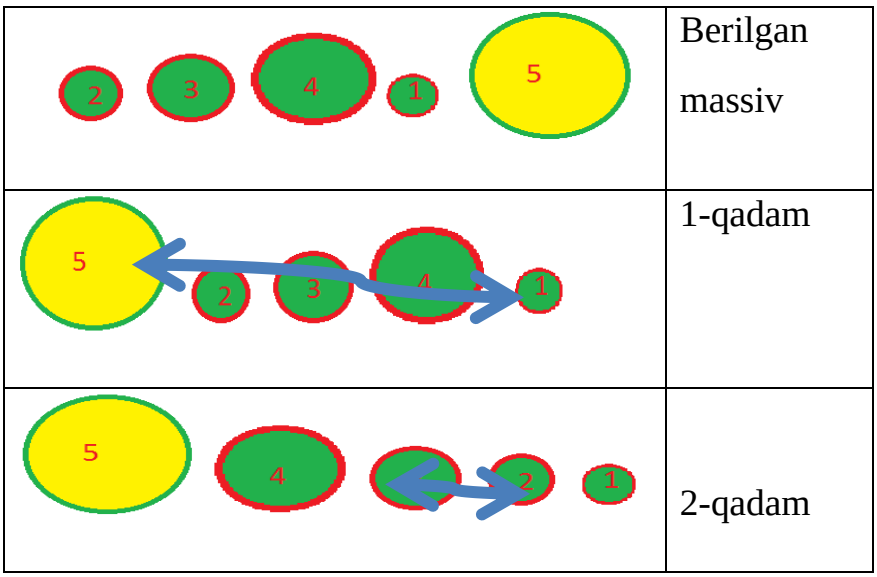
Saralash jarayoni oxirgi ro‘yhatning barcha elementlari shakllangunicha davom etadi, aks holda Ayverson sharti bajarilamydi.

**Ayverson sharti**: agar saralash jarayonida elementlarni taqqoslashda bir marotaba bo‘lsa ham o‘zgartirish bo‘lmasa, u holda to‘plam tartiblangan hisoblanadi (Ayverson sharti qadam *d=1* bo‘lganda bajariladi).

**Sheker usulida saralash**

Standart almashtirish saralash usulidan biri sheker yoki chelnochnaya saralash hisoblanadi. Bu yerda elementlar o‘zaro saralanib boriladi. Bu bilan birinchi o‘tish chapdan o‘ngga, ikkinchisi esa o‘ngdan chapga bo‘ladi va hokazo. Bir so‘z bilan aytganda, ro‘yhat elementlarning yo‘nalishi o‘zgaradi.

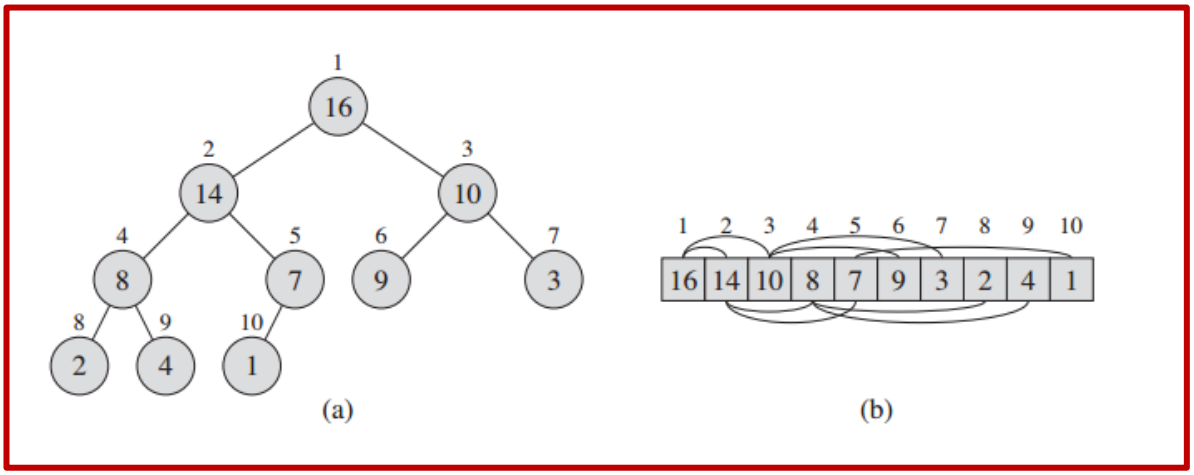
Standart almashish usulining murakkabligi- $O(n^2)$




**Pufakcha (qalqib chiqish) usuli.** A[0], A[1],.., A[N] massivning elementlari berilgan bo‘lsin. Ketma-ket ravishda A[0], va A[1], A[1], va A[2] elementlar o‘zaro taqqoslanib agar A[i]>a[i+1] bo‘lsa, ular o‘zaro o‘rin almashadilar. Ikkinchi qadamda shu holat A[N-1] gacha davom ettiriladi va hokazo. Bu usul

pufakcha(qalqib chiqish) usuli deyilishiga sabab har safar hajmi katta “sharcha” element qolganlarini ortda qoldirib yuzaga “qalqib” chiqadi.

**Piramida usulida saralash.** (Binar) piramidaning tuzilishi butun binar daraxt sifatida qaralishi mumkin bo‘lgan obyekt-massivni ifodalaydi. Ushbu daraxtning har




bir tuguni massiv elementlariga mos keladi. Daraxt chapdan o‘ngga qarab to‘lib boradigan mavjud bo‘lishi mumkin bo‘lgan eng pastkidan tashqari barcha sathlar bo‘yicha to‘ldirilgan. piramidani tasvirlovchi **A** massiv ikkita atributlarga ega obyekt hisoblanadi: odatda massiv elementlar sonini beradigan **A.length** va piramidaning nechta elementlari **A** massivda joylashganini ko‘rsatuvchi **A.heap-size.** Shuningdek, $\mathbf{A}[\mathbf{1}..\mathbf{A}.\mathbf{length}]$ ega faqat $\mathbf{0} \leq \mathbf{A}.\mathbf{heap} - \mathbf{size} \leq \mathbf{A}.\mathbf{length}$ oraliqagi $\mathbf{A}[\mathbf{1}..\mathbf{A}.\mathbf{heap} - \mathbf{size}]$ qism massiv elementlari piramidaning to‘g‘ri elementlari hisoblanadigan ba’zi bir sonlarga ega bo‘lishi mumkin. $\mathbf{A}[\mathbf{1}]$ daraxtning ildizi bo‘ladi, berilgan **i** indeks tuguni uchun uning ota-ona indekslarini chap va o‘ng tugunlar orqali oson hisoblash mumkin.

Binar piramidalarni ikki turga ajratishadi: kamaymaydigan va o‘smaydigan. Piramidalarning asosiy xususiyatlaridan biri hisoblanadigan piramidalar xossalari (heap property) qoniqtiradigan ikkala ko‘rinishdagi qiymatlar tugunlarda joylashadi. **O‘smovchi piramidalar xossalari (max-heap property)** dan biri har bir ***i*** indeksli ildiz tugun uchun quyidagi tengsizlik bajariladi:

$$\boldsymbol{A}[\boldsymbol{PARENT}(\boldsymbol{i})] \geq \boldsymbol{A}[\boldsymbol{i}].$$

Shu tarzda, O‘smovchi piramidaning eng katta elementi daraxt ildizida qiymatlari esa qism daraxt tugunlarida joylashgan bo‘ladi. Kamayvovchi piramida

prinsipi esa mutlaqo teskaridir. **Kamayvovchi piramida xossasi (min-heap property)** da har bir ***i*** indeksli ildiz tugun uchun quyidagi tengsizlik bajariladi:

$$A[PARENT(i)] \leq A[i].$$

Shuning uchun ham piramidaning eng kichik elementi ildizda joylashgan bo‘ladi.

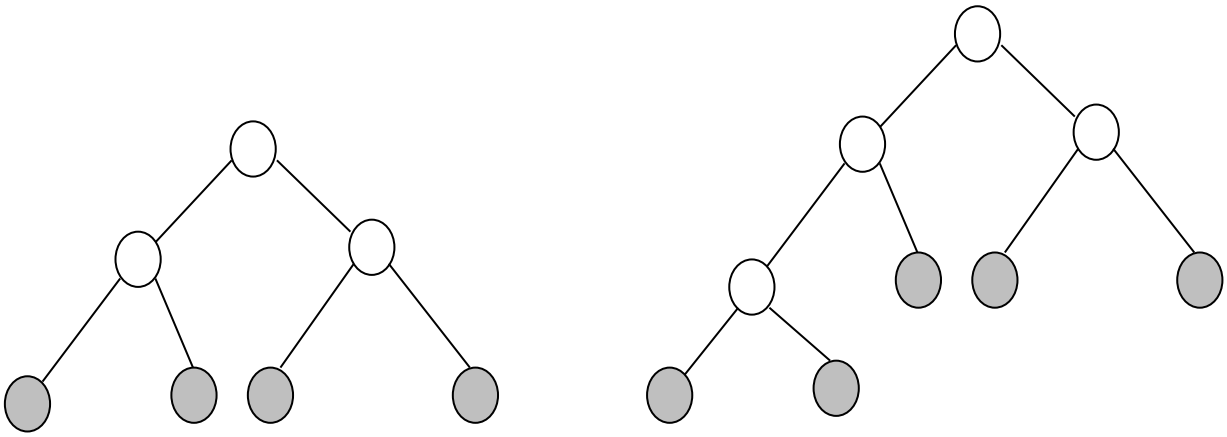
**Piramida xossasini saqlanishi.**

O‘smovchi piramida xossalarini saqlab turish uchun **MAX-HEAPIFY** prosedurasini chaqiramiz. Uning kiritilganlari **A** massiv va shu massivdagi **i** indeks hisoblanadi. **MAX-HEAPIFY** prosedurasini chaqirishda **LEFT(i) va RIGHT(i)** ildizlardan iborat binar daraxt o‘smovchi piramidani ifodalashi taxmin qilinadi, ammo **A**[**i**] farzand tugunlardan kichik bo‘lishi ham mumkin.

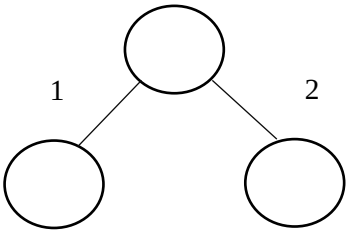
**MAX-HEAPIFY** prosedurasi ko‘rsatilgan. har bir qadamda **A**[**i**], **A**[**LEFT**(**i**)] **va A**[**RIGHT**(**i**)] elementlaridan eng katta element aniqlanadi va uning indeksi **largest** o‘zgaruvchida saqlanadi. Agar **A**[**i**] eng katta bo‘lsa, u holda **i** ildizdan iborat qismdaraxt o‘smovchi piramidani tasvirlaydi va prosedura to‘xtatiladi. Agar **A**[**LEFT**(**i**)] , **A**[**RIGHT**(**i**)] lardan biri eng katta bo‘lsa, u holda prosedura **A**[**i**] **ni A**[**largest**] bilan almashtirilib, **i** tugun va uning farzand tugunlari uchun o‘smovchi piramida xossasi bajariladi. Biroq **A**[**i**] ning dastlabki qiymati tugunning **largest** indeksida ekanligi kelib chiqdi, bu esa **largest** ildizdan iborat qismdaraxt o‘zining o‘smovchi piramida xossalarini buzishiga olib keladi. Shuning uchun ham ushbu daraxt uchun **MAX-HEAPIFY** prosedurasini rekursiv chaqirish kerak bo‘ladi.

Piramida saralash usuli piramidali daraxtni qurish bilan ifodalanadi. Piramidali daraxt – bu binar daraxt bo‘lib, uchta xossaga egadir:

1. Har bir triad uchida eng katta element joylashadi.
2. Binar daraxt barglari bir sathda yoki ikkilasi qo‘shni joylashadi (7.3-rasm).
3. Pastki sath barglari baland barglar sathidan chaproqda joylashadi.



**7.3-rasm. Binar daraxt:a-bir sathdagi barglar; b-qo‘shni sathlardagi burglar**




**7.4-rasm. Triad elementlarini taqqoslash: 1- birinchi taqqoslash; 2- ikkinchi taqqoslash.**

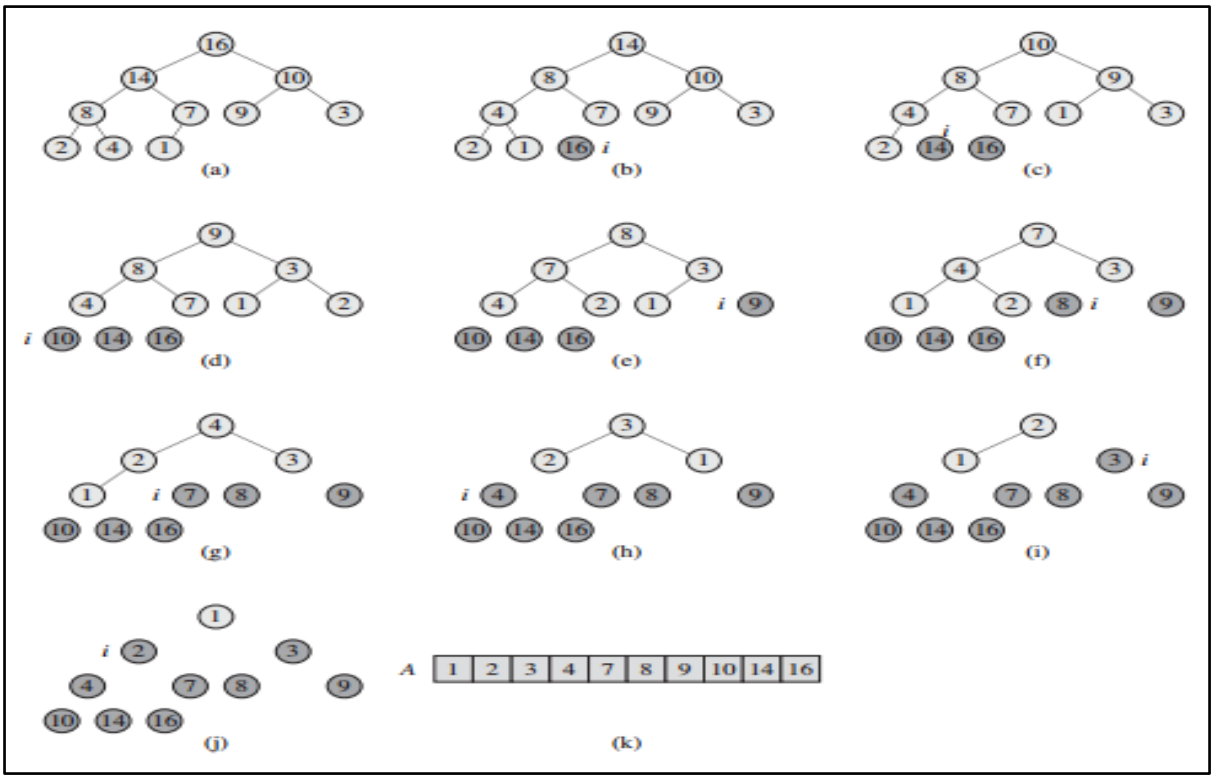
Aylantirish jarayonida triad elementlari ikki marotaba taqqoslanadi (7.4-rasm), shu bilan birga eng katta element yuqoriga, eng kichik element esa pastga o‘tadi.

$\boldsymbol{n = A.length}$ bo‘lgan $\boldsymbol{A[1..n]}$ kirish massivida o‘smovchi piramidani qurish uchun piramida usulida saralash algoritmi **BUILD-MAX-HEAP** prosedurasini chaqirishdan boshlanadi. Massivning eng katta elementi $\boldsymbol{A[1]}$ bo‘lganligi sababli, uni $\boldsymbol{A[n]}$ elementi o‘rni bilan almashtirib, saralangan massivning aniq va oxirgi pozitsiyaga qo‘yish mumkin.

1-qatorda boshlang‘ich o‘smovchi piramida qurilgandan so‘ng **HEAPSORT** prosedurasini ishlashi keltirilgan. Rasmda 2-5 qatorlarda birinchi va keyingi har bir **for** silk iteratsiyasidan o‘smovchi piramida ko‘rsatilgan.

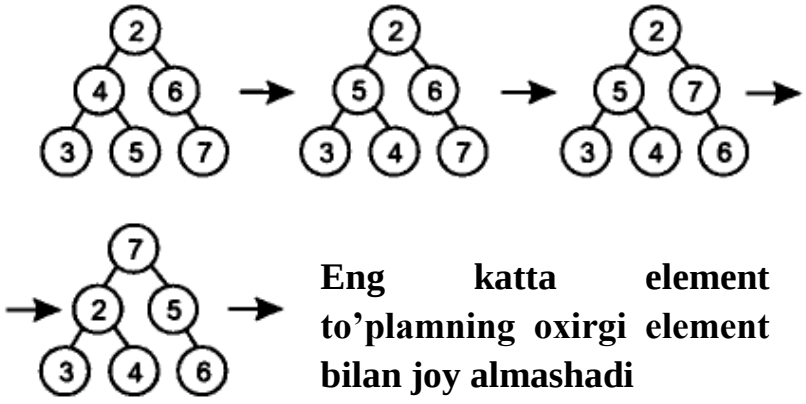
**BUILD-MAX-HEAP** prosedurasini chaqirish $\boldsymbol{O(n)}$ vaqt talab etishi hamda **MAX-HEAPIFY** prosedurasining har bir ***n-1*** chaqiruv vaqti $\boldsymbol{O(lgn)}$ ga teng

ekanligidan, **HEAPSORT** proseduralarining ish vaqti $\boldsymbol{O(nlgn)}$ ga teng bo‘ladi.

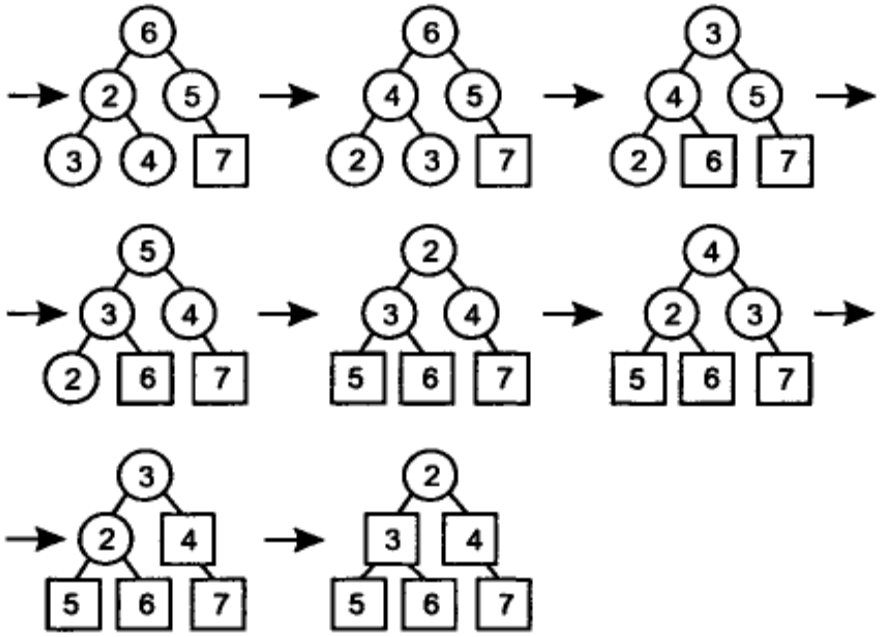



**Piramida usulida saralash.**

**7.2-misol.** Dastlabki $\{\mathbf{2},\mathbf{4},\mathbf{6},\mathbf{3},\mathbf{5},\mathbf{7}\}$ to‘plam berilgan. Piramida usulida saralash 7.5-rasmda ko‘rsatilgan.

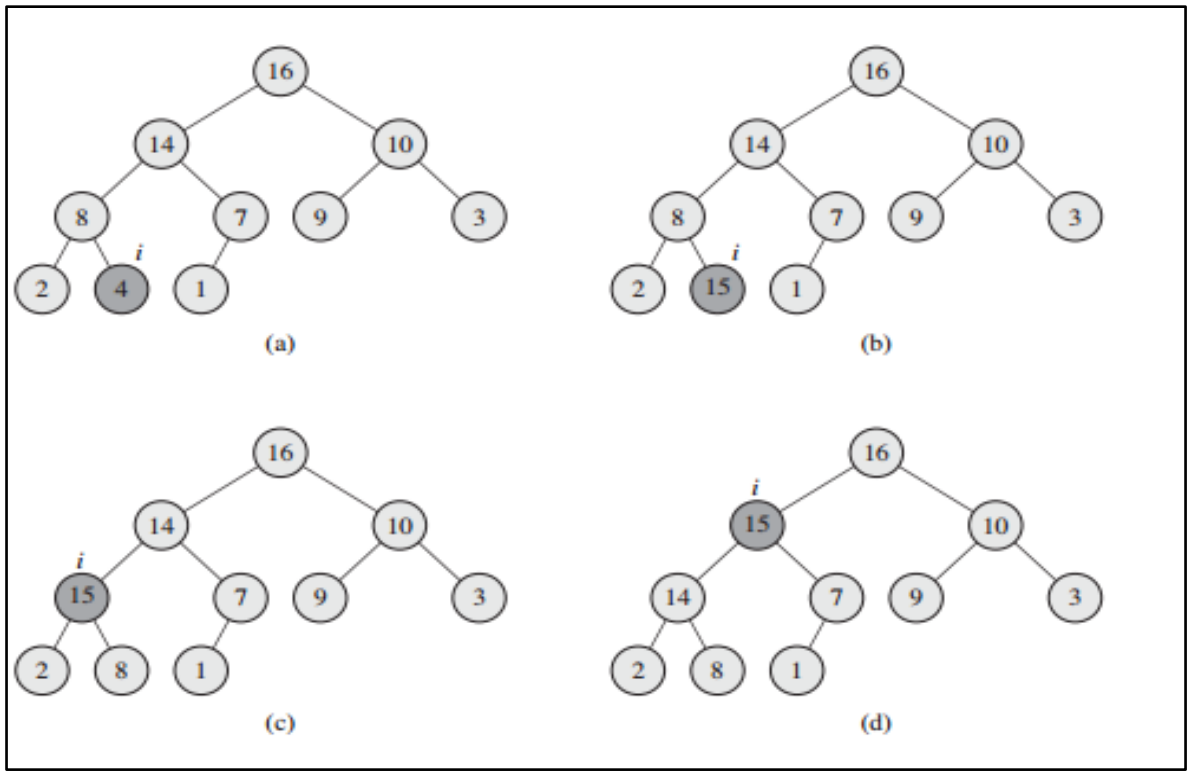


**Eng katta element to’plamning oxirgi element bilan joy almashadi**




**7.5-rasm. Piramida usulida saralash.**

Natijada tartiblangan $\{\mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, \mathbf{7}\}$ to‘plam hosil bo‘ladi.




**7.3-misol**. Rasmdan namuna sifatida foydalanib, **MAX-HEAPIFY*(A,3)*** prosedurasini $\boldsymbol{A} = \{\mathbf{27}, \mathbf{17}, \mathbf{3}, \mathbf{16}, \mathbf{13}, \mathbf{10}, \mathbf{1}, \mathbf{5}, \mathbf{7}, \mathbf{12}, \mathbf{4}, \mathbf{8}, \mathbf{9}, \mathbf{0}\}$ massiv uchun ishlashini ko‘rsating.

**Qo‘shimchalar bilan saralash.** Qo‘shimchalar bilan saralashning asosiy g‘oyasi shundaki, yangi elementni saralangan ro‘yxatga qo‘shishda uni ixtiyoriy

joyga emas, balki kerakli joyga joylashtirish lozim, so‘ng butun ro‘yxatni boshqatdan saralash kerak. Qo‘shimchalar bilan saralashda ixtiyoriy ro‘yxatning birinchi elementi saralangan ro‘yxatning 1 uzunligi deb hisoblanadi. 2 elementli saralangan ro‘yxat boshlang‘ich ro‘yxatdagi 2-elementni 1-element joylashgan ro‘yxatning kerakli o‘rniga joylashtirish orqali yaratiladi. Endi boshlang‘ich ro‘yxatning 3-elementini saralangan 2 elementli ro‘yxatga qo‘yish mumkin. Bu jarayon boshlang‘ich ro‘yxatning barcha elementlari kengayuvchi saralangan ro‘yxatga joylashgunga qadar davom etadi.

Bu jarayonni ifodalovchi algoritm quyida keltirilgan:

```
InsertionSort(list,N)
list
N
For i=2 to N do
     newElement=list[i]
     location=i-1
     while(location>=1) and (list[location]>newElement)
do
          list[location=1]=list[location]
          location=location-1
     end while
     list[location+1]=newElement
end for
```

Bu algoritm yangi qo‘yiladigan elementni newElement o‘zgaruvchisiga yuklaydi. So‘ng barcha katta elementlarni 1 pozitsiyaga surib, bu elementga joy ajratadi. siklning oxirgi iteratsiyasi elementni location+1 nomer bilan location+2 pozitsiya o‘tkazadi.

**Eng yomon holat tahlili.** Agar ichki while siklini ko‘rib chiqadigan bo‘lsak, yana boshqatdan qo‘shilayotgan element barcha elementlarda kichik va ro‘yxatning saralangan qismida joylashgan bo‘lsa, jarayonning katta qismi bajariladi. Bu holatda location o‘zgaruvchisining qiymati 0 ga teng bo‘lganda sikl to‘xtaydi. SHuning uchun algoritm bajarilishining asosiy qismi har bir yangi element ro‘yxat boshiga

qo‘shilganda amalga oshiriladi. Bu faqatgina boshlang‘ich ro‘yxat elementlari kamayish tartibda bo‘lganda bajariladi. Bu eng yomon hodisalardan biri, lekin boshqalari ham bor.

Bunday ro‘yxatni qayta ishlashni qanday amalga oshirish kerakligini ko‘rib chiqamiz. Dastlab ro‘yxatning 2-elementi qo‘yiladi. U faqatgina bitta element bilan taqqoslanadi. Ikkinchi qo‘yilgan element oldingi ikkita element bilan, uchinchi qo‘yiladigan element uchta element bilan va hokazo, i-qo‘yiladigan element i ta element bilan taqqoslanadi hamda bu jarayon N-1 marta takrorlanadi. Shunday qilib, eng yomon holatdagi saralash murakkabligi quyidagiga teng:

$$W(N) = \sum_{i=1}^{N-1} i = \frac{(N-1)N}{2} = \frac{N^2 - N}{2}$$

$$W(N) = O(N^2)$$

**O‘rta holat tahlili.** O‘rta hol tahlilini 2 bosqichga ajratamiz. Dastlab navbatdagi element holatini aniqlash uchun kerak bo‘ladigan taqqoslashlarning o‘rta qiymatini hisoblaymiz. Keyin, 1-qadam natijasidan foydalanib, barcha kerakli amallarning o‘rta qiymatini hisoblash mumkin. Dastlab, i-elementning joylashish o‘rnini aniqlash uchun taqqoslashlarning o‘rta qiymatini hisoblaymiz. YUqorida aytib o‘tganimizdek, ro‘yxatga i-elementni qo‘shganda u kerakli joyda joylashgan bo‘lsa ham, hech bo‘lmaganda bitta taqqoslash bajariladi.

i-element uchun nechta mumkin bo‘lgan holat mavjud? Qisqa ro‘yxatlarni ko‘rib chiqamiz va ixtiyoriy holatda natijalarni umumlashtirishga harakat qilamiz. 1-qo‘shiladigan element uchun 2 ta imkoniyat bor: ikki elementdan birinchisi yoki ikkinchisi bo‘lishi mumkin. 2-qo‘shiladigan elementning 3 holatdan birida bo‘lishi mumkin: 1,2,3 raqamlari bilan. Demak, i-qo‘shiladigan element i+1 ta holatdan birini egallashi mumkin. Barcha imkoniyatlar ehtimollik darajasini teng deb olamiz.

Har bir i+1 pozitsiyaga yetib olish uchun nechta taqqoslash talab etiladi? i ning kichik qiymatlarini ko‘rib chiqamiz va natijani birlashtirishga harakat qilamiz. Agar 4-qo‘shilayotgan element 5-pozitsiyaga tushsa, u holda taqqoslash yolg‘on bo‘ladi. Agar uning 4-pozitsiyasi to‘g‘ri bo‘lsa, u holda 1-taqqoslash rost hisoblanadi, ikkinchisi esa – yolg‘on. 3-pozitssiyaga tushsa, 1- va 2-taqqoslash rost,

3-si esa yolg‘on bo‘ladi. 2-pozitsiyada 1-,2- va 3-taqqoslash rost va 4-si yolg‘on bo‘ladi. 1-pozitsiyada barcha taqqoslashlar rost bo‘ladi, yolg‘on taqqoslashlar bo‘lmaydi. Bundan esa i+1, i, i-1,…, 2 pozitsiyalarga tushuvchi i-element uchun taqqoslashlar mos holda 1,2,3 …i ekanligi kelib chiqadi. 1-pozitsiyaga tushganda esa taqqoslashlar soni i ga teng bo‘ladi. i- qo‘yiladigan element uchun taqqoslashlarning o‘rta qiymati quyidagi tenglik bilan beriladi:

$$A_i = \frac{1}{i+1}\left(\sum_{p=1}^{i} p + i\right) = \frac{1}{i+1}\left(\frac{i(i+1)}{2} + i\right) = \frac{i}{2} + \frac{i}{i+1} = \frac{i}{2} + 1 - \frac{1}{i+1}$$

Biz i-elementni qo‘yishdagi taqqoslashlarning o‘rta qiymatini hisobladik. Endi bu natijani ro‘yxatdagi har bir N-1 element uchun yig‘ish kerak:

$$A(N) = \sum_{i=1}^{N-1} A_i = \sum_{i=1}^{N-1}\left(\frac{i}{2} + 1 - \frac{1}{i+1}\right)$$

$$A(N) = \sum_{i=1}^{N-1} \frac{i}{2} + \sum_{i=1}^{N-1} 1 - \sum_{i=1}^{N-1} \frac{1}{i+1}$$

$$\sum_{i=1}^{N-1} \frac{i}{2} = \sum_{i=1}^{N-1} 1 = \sum_{i=1}^{N-1} \frac{1}{i+1}$$

$$A(N) \approx \frac{1}{2}\frac{(N-1)N}{2} + (N-1) - (\ln N - 1) = \frac{N^2 - N}{4} + (N-1) - (\ln N - 1)$$
$$= \frac{N^2 + 3N - 4}{4} - (\ln N - 1) \approx \frac{N^2}{4};$$

$$A(N) = O(N^2).$$

**Pufaksimon saralash.** Bu usulning asosiy prinsipi kichik qiymatlarni surish natijasida ro‘yxatning yuqori qismiga o‘tkazish va shu vaqtning o‘zida katta elementlarni pastga tushurishdan iborat. Pufaksimon saralashning turli variantlari bor. Bu paragrafda variantlardan birini ko‘rib chiqamiz, qolganlarini topshiriqlarda uchratishingiz mumkin.

Pufaksimon saralash algoritmi ro‘yxat bo‘yicha bir nechta o‘tish yo‘llarini amalga oshiradi. Har bir o‘tishda qo‘shni elementlar taqqoslanadi. Agar qo‘shni elementlarning tartibi noto‘g‘ri bo‘lsa, u holda ularning joyi almashadi. Har bir

o‘tish ro‘yxat boshidan boshlanadi. Dastlab birinchi va ikkinchi element taqqoslanadi, so‘ng 2- va 3-element, keyin 3- va 4-element taqqoslanadi va hokazo; noto‘g‘ri tartibda turgan elementlar joy almashadi. Birinchi o‘tishda ro‘yxatning eng katta elementi topilsa, u barcha elementlar bilan ro‘yxat oxirigacha o‘rin almashadi. 2-o‘tishda ro‘yxatning kattaligi bo‘yicha 2-elementi oxiridan ikkinchi o‘ringa o‘tadi. Bu jarayonning davom etishi natijasida har bir element o‘z o‘rnini egallaydi. Bunda kichik qiymatlar ham yuqoridan o‘z o‘rniga joylashadi. Agar biror o‘tishda elementlarning o‘rin almashishi bajarilmasa, u holda barcha elementlar kerakli tartibda joylashgan bo‘ladi va algoritmning bajarilishi to‘xtatiladi. SHuni ta’kidlash kerakki,har bir o‘tishda bir vaqtning o‘zida bir nechta element o‘z o‘rni tomon siljib boradi. Pufaksimon saralash algoritmi quyida keltirilgan:

```
BubbleSort(list,N)
list
N
number0fPairs=N
swappedElements=true
while swappedElements do
     number0fPairs=number0fPairs-1
     swappedElements=false
     for i=1 to number0fPairs do
          if list[i]>list[i+1]then
               Swap(list[i],list[i+1])
               swappedElements=true
          end if
     end for
end while
```

**Yaxshi holat tahlili.** SwappedElements bayroq izohi noto‘g‘ri ekanligini ta’kidlash uchun yaxshi holat tahlilini qisqacha ko‘rib chiqamiz. Bajarilayotgan ish

hajmi qaysi holatda minimal bo‘lishini ko‘rib chiqamiz. For sikli 1-o‘tishda to‘liq bajarilishi kerak va shuning uchun algoritmga kamida N-1 ta taqqoslash talab etiladi. 2 ta imkoniyatni ko‘rib chiqish kerak: 1-o‘tishda hech bo‘lmaganda bitta o‘rin almashtirish yoki o‘rin almashtirish bajarilmaydi. 1-holatda o‘rin almashtirish SwappedElements bayrog‘ining qiymati true ga o‘zgarishiga olib keladi, demak while sikli yana takrorlanadi, bunda N-2 ta taqqoslash talab etiladi. 2-holatda SwappedElements element bayrog‘i false qiymatini saqlab qoladi va algoritm bajarilishi to‘xtatiladi.

Shuning uchun, yaxshi holda N-1 ta taqqoslash bajariladi, bunda 1-o‘tishda o‘rin almashtirishlar bo‘lmaydi. Bundan esa ma’lumotlarning yaxshi to‘plami talab qilingan tartibda elementlar ro‘yxatini tashkil etishi kelib chiqadi.

**Yomon holat tahlili.** Yaxshi holda ro‘yxatdagi elementlar talab qilingan taribda joylashar ekan, elementlarning teskari tartibda joylashishi yomon holni bermasmikan, degan savol tug‘iladi. Agar eng katta element 1-o‘rinda tursa, u holda u ro‘yxat oxirigacha qolgan barcha elementlar bilan yonma-yon qo‘yib boriladi.1-o‘tishdan oldin katta qiymatdagi 2-element 2-holatni egallaydi, biroq 1-taqqoslash va 1-o‘rin almashtirish natijasida u 1-o‘ringa o‘tkaziladi. 2-o‘tishning boshida 1-holatda katta qiymatdagi 2-element joylashgan bo‘ladi va u ro‘yxatning oxiridan 2-elementgacha qolgan elementlar bilan yonma-yon o‘rin almashib boradi. Bu jarayon barcha qolgan elementlar uchun bajariladi, shuning uchun for sikli N-1 marta takrorlanadi. SHuning uchun berilganlarning teskari tartibi yomon holga olib keladi.

Yomon holda nechta taqqoslash amalga oshiriladi? 1-o‘tishda qo‘shni elementlarning N-1 ta taqqoslashi, 2-o‘tishda esa N-2 ta taqqoslash bajariladi. Tadqiqotlar shuni ko‘rsatadiki, har bir navbatdagi o‘tishda taqqoslashlar soni bittaga kamayadi. SHuning uchun yomon hol murakkabligi quyidagi formula bilan beriladi:

$$W(N) = \sum_{i=N-1}^{1} i = \sum_{i=1}^{N-1} i = \frac{(N-1)N}{2} = \frac{N^2 - N}{2} \approx \frac{1}{2} N^2 = O(N^2).$$

**O‘rta holat tahlili.** Yuqorida ta’kidlaganimizdek, yomon holatda for sikli N-1 marta takrorlanadi. O‘rta holda elementlarning o‘rnini almashtirmasdan hosil bo‘lgan o‘tishlarning ixtiyoriysida ehtimolligi teng deb faraz qilamiz. Har bir

holatda nechta taqqoslash bajarilishini ko‘rib chiqamiz. 1-o‘tishdan keyin taqqoslash soni N-1 ga teng bo‘ladi. 2 ta o‘tishdan keyin N-1+N-2 ta bo‘ladi. birinchi i ta o‘tishda taqqoslashlar sonini S(i) deb belgilaymiz. Agar birorta ham o‘rin almashtirish bajarilmasa, algoritm o‘z ishini to‘xtatadi. O‘rta hol tahlilida bunday imkoniyatlarni ko‘rib chiqish lozim. Natijada quyidagi tenglikka ega bo‘lamiz:

$$A(N) = \frac{1}{N-1}\sum_{i=1}^{N-1} C(i)$$

bu yerda S(i) – for siklining dastlabki i ta o‘tish oralig‘idagi taqqoslashlar soni. Bunga nechta taqqoslash talab etiladi? S(i) ning qiymati quyidagi tenglikda ko‘rsatilgan:

$$C(i) = \sum_{j=N-1}^{i} j = \sum_{j=i}^{N-1} j = \sum_{i=i}^{N-1} j - \sum_{j=i}^{i-1} j$$

yoki

$$C(i) = \frac{(N-1)N}{2} - \frac{(i-1)i}{2} = \frac{N^2 - N - i^2 + i}{2}$$

Oxirgi tenglikni A(N) ifodaga qo‘yib quyidagini hosil qilamiz:

$$A(N) = \frac{1}{N-1}\sum_{i=1}^{N-1} \frac{N^2 - N - i^2 + i}{2}$$

N i ga bog‘liq bo‘lmaganligi uchun:

$$A(N) = \frac{1}{N-1}((N-1)\frac{N^2 - N}{2} + \sum_{i=1}^{N-1} \frac{-i^2 + i}{2})$$

$$A(N) = \frac{N^2 - N}{2} + \frac{1}{2(N-1)}(\sum_{i=1}^{N-1}(-i^2) + \sum_{i=1}^{N-1} i)$$

Natijada:

$$A(N) = \frac{N^2 - N}{2} + \frac{1}{2(N-1)}(-\frac{(N-1)N(2N-1)}{6} + \frac{(N-1)N}{2})$$

$$= \frac{N^2 - N}{2} - \frac{(2N-1)N}{12} + \frac{N}{4} = \frac{6N^2 - 6N - 2N^2 + N + 3N}{12}$$

$$= \frac{4N^2 - 2N}{12} \approx \frac{1}{3}N^2 = O(N^2)$$

## Savol va topshiriqlar

1. Bir o‘lchovli massiv elementlarini “Pufakcha” usulida saralang.

2. Ikkita bir o‘lchovli (K va N) massivlarni tartiblangan K+N o‘lchamda shakllantiring.

3. Tezkor saralash (Quick sort) algoritmini yozing.

4. M={13,2,6,66,51,19,3,11,34,20} massivni o‘sish tartibida Sheker saralash usulida saralang.

5. M={5, 4, 8, 2, 9} massivni o‘sish tartibida almashtirish saralash usulida saralang.

6. M={12, 4, 7, 11, 9, 13, 5, 2} massivni o‘sish tartibida pufakcha saralash usulida saralang.

7. M={12,6,65,7,0,4,9,16,36,7,64,13,25,1,44,5} massivni kamaymaslik tartibida almashtirish saralash usulida saralang.

8. M={11,12,17,3,9,10,34,20} massivni o‘sish tartibida sheker saralash usulida saralang.