**8-Amaliy ish**

**Mavzu:** **Ajrat va xukmronlik qil prinsipi bo’yicha ishlaydigan algoritmlarni loyihalash. Elementlar jamlanmasini biror belgi bo’yicha tartiblashtirish algoritmi**

Birlashtirish orqali saralash(Merge Sort) algoritmi.

Bu algoritm Jon fon Neyman tamonidan 1946 yilda taklif qilingan.

Jon Fon Neyman Vengriyalik olim bo’lib matematika, kvant fizikasi, funksional analiz, to’plamlar nazariyasi, ekonomika, informatika kabi fanlarga munosib hissa qo’shgan.

**Bo’lib tashla va hukmronlik qil metodi.**

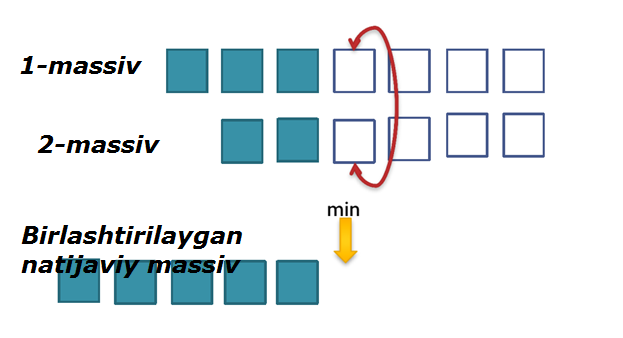
* Algoritmlarni qurishning asosiy metodlaridan biri.
* Murakkab masalani yechish uchun, uni oddiyroq bo’laklarga ajratish kerak.

**Massivni ham huddi shunday saralash mumkin:**

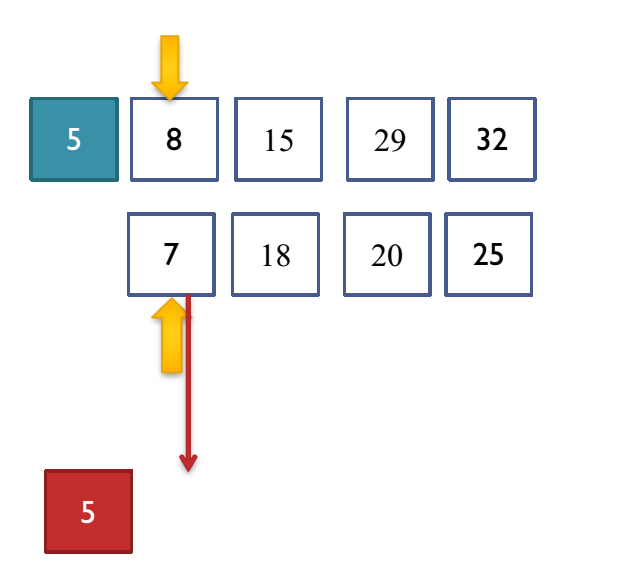
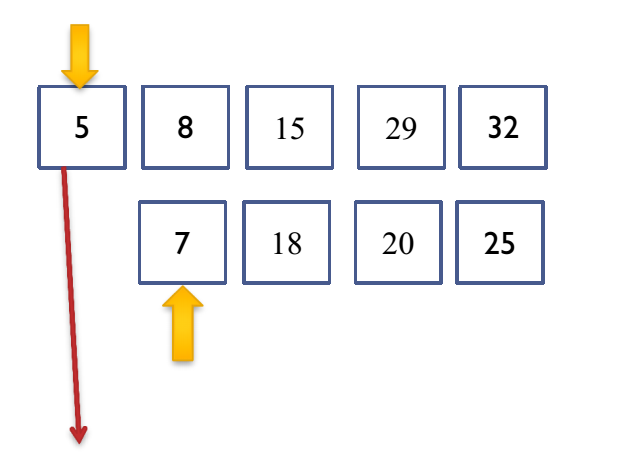
* Uni ikkita bo’lakga ajratamiz.
* Bo’laklarni alohida saralaymiz.
* Saralangan massivlarni birlashtiramiz.

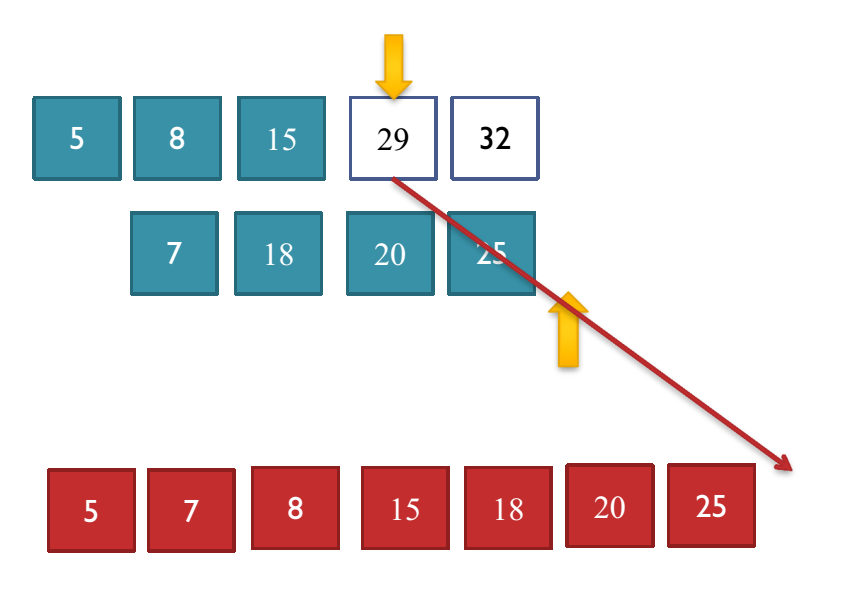
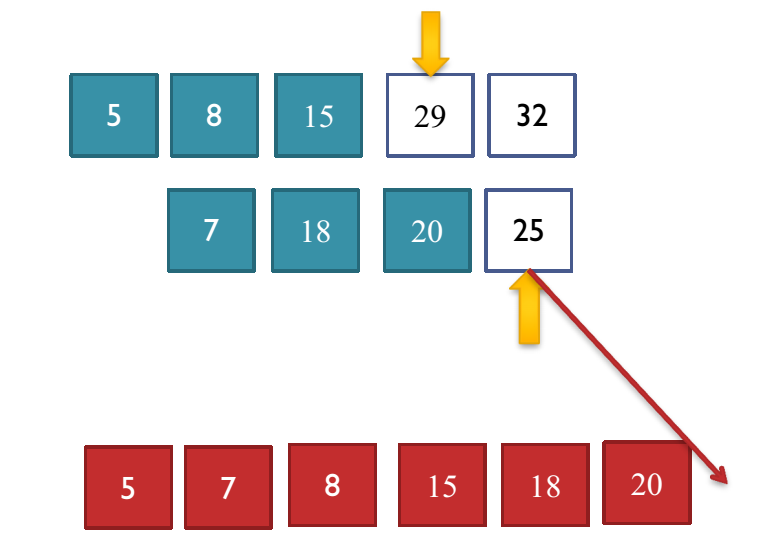
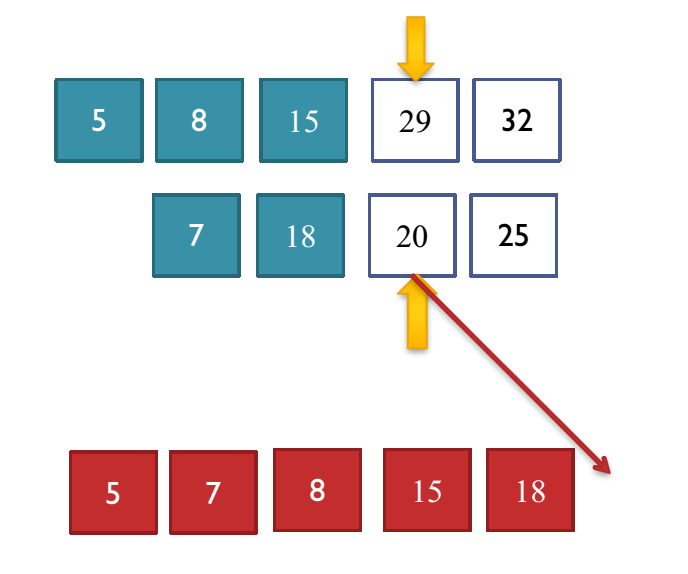
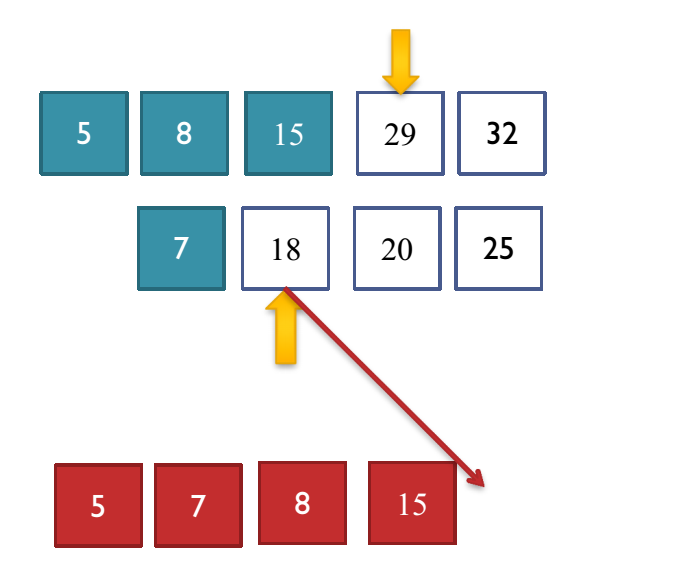
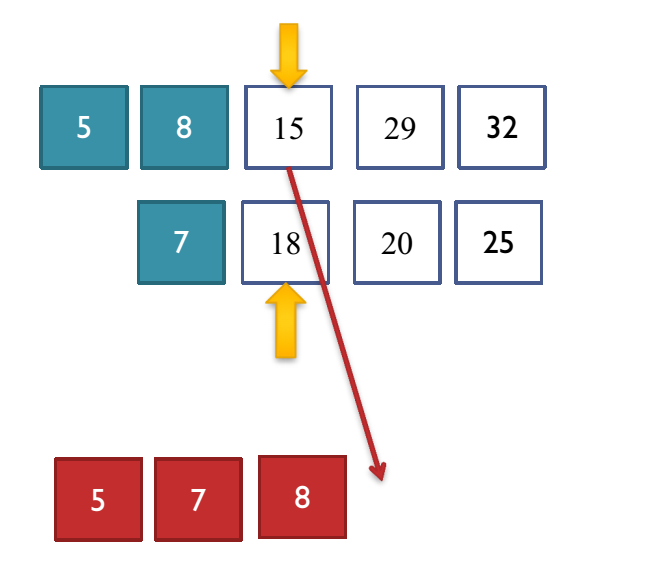
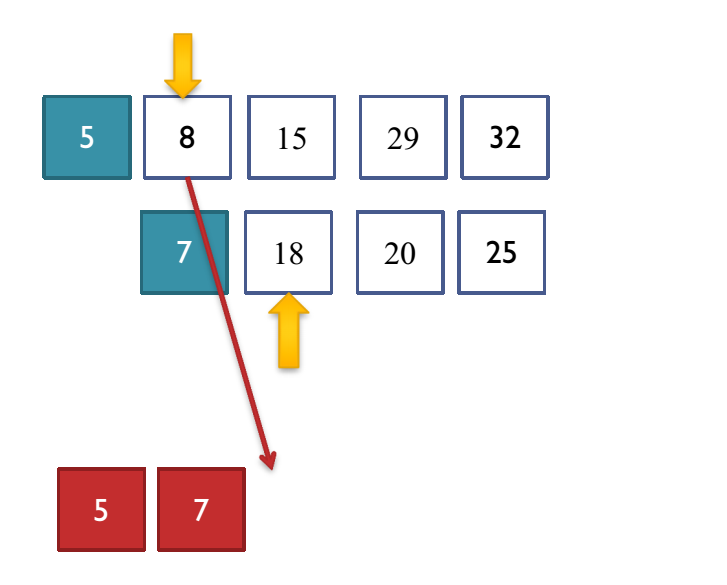
**Saralangan massivlarni birlashtirish.**

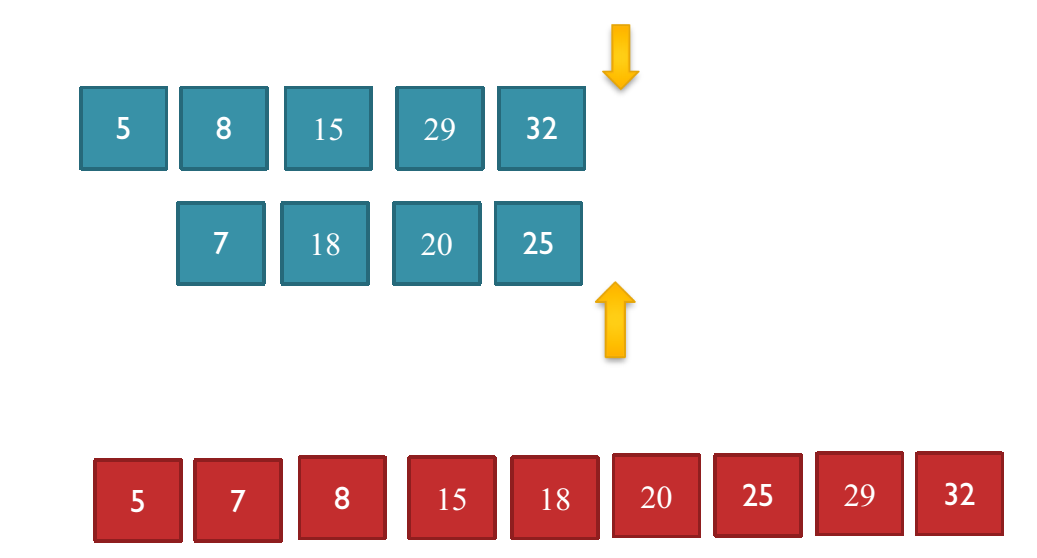
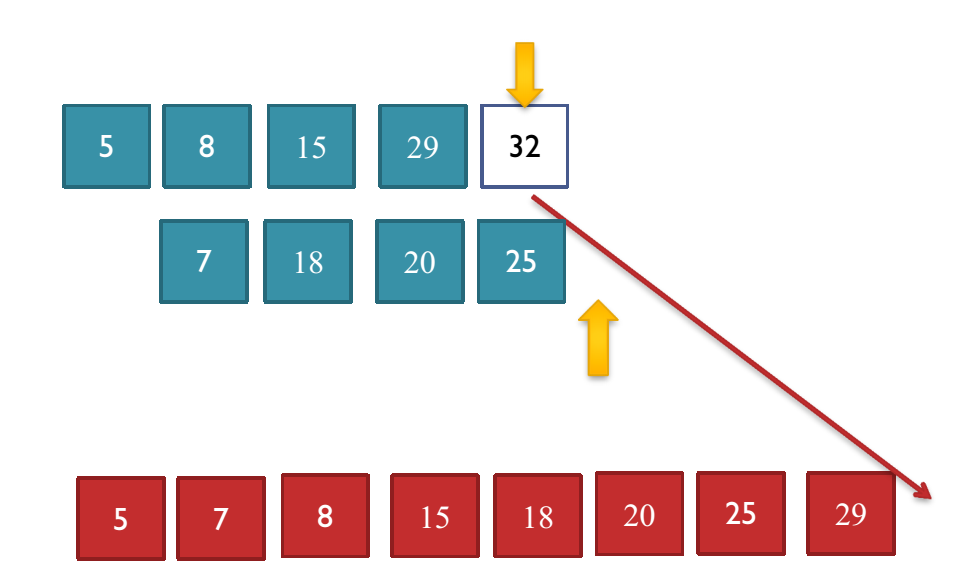
* Ikkita saralangan massiv berilgan. Ularni birlashtirib shunday massiv hosil qilish qilish kerakki, u yana saralangan bo’lsin.
* Xar safar hali ikki massivning hali ko’rilmagan qismlaridagi birinchi ikki elementni taqqoslaymiz. Ulardan kichigini olamiz. Bu jarayonni toki bitta massivning chetigacha chiqmagunga qadar davom ettiramiz. Ortib qolgan massiv elementlarini esa to’g’ridan to’g’ri natijaviy massiv iziga berilgan tartibda joylashtirib qo’yamiz.

****

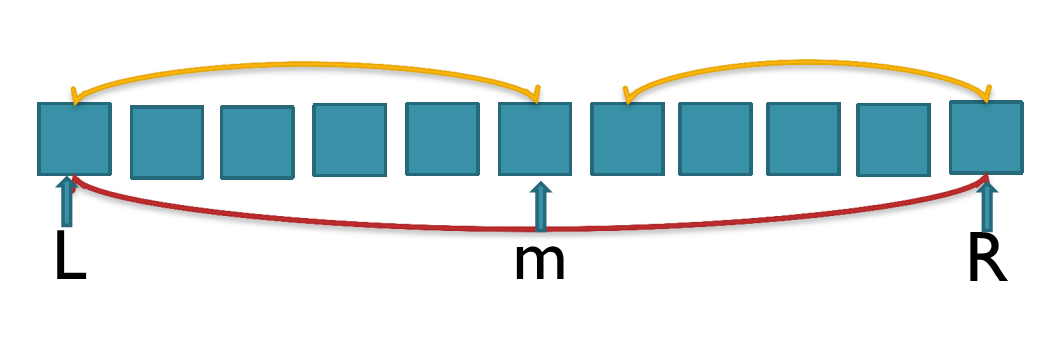
**Masalan quyida ikki saralangan massivlarni birlashtirib, saralangan bitta massiv hosil qilishni ko’rib chiqaylik:**



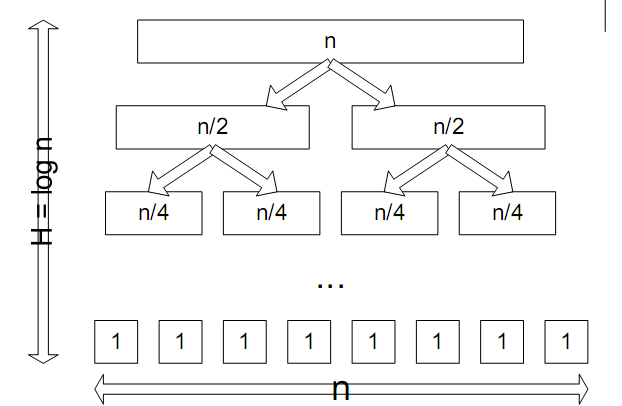




**Endi Merge sort algoritmiga qaytsak:**



[L, R] oraliq m=(L+R) / 2 o’rtasi orqali ikkita [L, m] va [m+1, R] oraliqqa ajratiladi va ular alohida saralanadi. Keyin bu saralangan oraliqlar birlashtirilib, berilgan [L, R] oraliqqa qo’yib ketiladi.

****

**Birlashtirish uchun bajariladigan amallar :**

int p1 = L, p2 = m+1;

int p = L;

while (p1 <= m && p2 <= R) {

if (a[p1] <= a[p2]) {

b[p] = a[p1];

p++;

p1++;

}

else {

b[p] = a[p2];

p++;

p2++;

}

}

while (p1 <= m) {

b[p] = a[p1];

p++;

p1++;

}

while (p2 <= R) {

b[p] = a[p2];

p++;

p2++;

}

for (int i = L; i <= R; i++)

a[i] = b[i];

**Merge sort ning to’liq kodi:**

#include <iostream>

#include <math.h>

#include <stdio.h>

using namespace std;

int a[100000], b[100000];

void mergesort(int L, int R) {

if (L >= R)

return;

else {

int m = (L+R) / 2;

mergesort(L, m);

mergesort(m+1, R);

//Birlashtirish yoziladi

}

}

int main() {

int n;

cin>>n;

for (int i = 0; i < n; i++)

cin>>a[i];

mergesort(0, n-1);

for (int i = 0; i < n; i++)

cout<<a[i]<<" ";

return 0;

}

**Merge sort algoritmining qo’llanilishi**

Merge sort(birlashtirish orqali saralash) algorit-mining qo’llanilishining eng yaxshi misoli bu inversiyalar sonini topishdir. Inversiya deb 1≤i<j≤n va a[i]>a[j] bo’lgan (i, j) juftliklarga aytildi, ya’ni katta son kichik sondan oldin joylashgan bo’lsa inversiya xisoblanadi. Oddiy usulda barcha juftliklarni ko’rib chiqish O(n2) amal talab qiladi. Merge sort algoritmida massivning ikki saralangan qismini saralangan-ligini saqlab qolgan holda birlashtirishda inversiyalar sonini hisoblab boramiz. Masalan:

* **5 8 15 29 32 va 7 18 20 25**

Yangi massivga dastlab 5 ni olamiz. 5 ikkinchi qismning birinchi sonidan katta bo’lmagani uchun qolganlaridan ham katta emas demak u bilan bog’liq inversiya yo’q. Ikkinchida 5 ketgach qolgan qismlar boshidagi sonlardan minomali 7 va u ikkinchi qismda. Birinchi qismda undan katta bo’lgan 4 ta son bor(8, 15, 29, 32). Demak inversiyalar sonini 4 ga ortiramiz. Shuq tariqadavom etasi.

* **int p1 = L, p2 = m+1;**
* **int p = L;**
* **while (p1 <= m && p2 <= R) {**
* **if (a[p1] <= a[p2]) {**
* **b[p] = a[p1];**
* **p++;**
* **p1++;**
* **}**
* **else {**
* **inv += m-p1+1;**
* **b[p] = a[p2];**
* **p++;**
* **p2++;**
* **}**
* **}**

**5-Topshiriq**

     Sizga satr berilgan. Undagi so’zlar bir-biridan bitta probel bilan ajratib berilgan.Sizning vazifangiz undagi so’zlarni uning ichidagi ‘a’ harfining soni kamayish bo’yichasaralashdan iborat. Agar birnechta so’zda ‘a’ harfi soni bir xil bo’lib qolsa ularning bir-biriga nisbatan tartibi o’zgarmay qoldirilsin.

**Kiruvchi ma’lumotlar**

Birinchi qatorda sart berilgan. U kichik lotin alfaviti harflaridan va probellardaniborat bo’lishi mumkin. So’zlar bir-biridan bitta probel bilan ajratilgan. Satr bo’sh emasva uzunligi 1000 belgidan oshmaydi.

**Chiquvchi ma’lumotlar**

Saralangan so’zlarni birinchi qatorda bitta probel bilan ajrtib chiqaring.

**Misollar**

|  |  |  |
| --- | --- | --- |
| **№** | **Kiruvchi ma’lumotlar** | **Chiquvchi ma’lumotlar** |
| 1 | umida anora marhabo | anora marhabo umida |

**Python kodi:**

s = input()

words = s.split()

# So'zlarni 'a' harfi soniga asoslangan tartibda saralash

sorted\_words = sorted(words, key=lambda word: (word.count('a'), word), reverse=True)

# Saralangan so'zlarni birinchi qatorda bitta probel bilan ajratib chiqarish

result = ' '.join(sorted\_words)

print(result)

**Test:**

**umida anora marhabo amvarali marhabo guli azamat**

**azamat amvarali marhabo marhabo anora umida guli**