4 - AMALIY MASHG‘ULOT. *QO’SHIMCHA SARALASH ALGORITMLARI.*

Ishdan maqsad: Talabalarni qo‘shimcha saralash algoritmlar bilan tanishtirish. C++ dasturlash tilida saralash algoritmlarga oid misollar yaratish.

Nazariy qism. Saralash algoritmlari informatika sohasidagi fundamental operatsiyalardan biridir. Bu ma’lumotlar to’plamining elementlarini ma’lum bir tartibda joylashtirish jarayonidir. Ko’plab saralash usullari mavjud, ularning har biri o’ziga xos xususiyatlar va afzalliklarga ega. Biroq, ayrim hollarda, faqat saralash usulidan foydalanish ma’lumotlarning tabiati yoki ishlash talablari tufayli samarali bo’lmasligi mumkin. Bunday vaziyatlarda birlashtirilgan saralash algoritmlari yordamga keladi, ular yaxshi ishlashga erishish uchun bir nechta saralash usullarining afzalliklarini birlashtiradi.

Birlashtirilgan saralash algoritmlari - ularni ikkita asosiy turga bo’lish mumkin: turli bosqichlarda turli xil saralash texnikasini birlashtirgan algoritmlar va saralash texnikasini bitta usulda birlashtirgan algoritmlar.

1.Tandem saralash (**Tandem Sort** ***):*** Bu usul tanlash va kiritish tartiblarini birlashtiradi. Birinchidan, biz tanlash tartibidan foydalanamiz, so’ngra ro’yxat hajmi kichiklashganda, biz kiritish tartibiga o’tamiz. Bu ro’yxatning allaqachon tartiblangan qismlarini saralashda ortiqcha operatsiyalardan qochadi.

2.Gibrid saralash (**Hybrid Sort** ***):*** Bu yondashuv ma’lumotlar hajmi va tuzilishiga qarab turli xil saralash usullarini birlashtiradi. Masalan, katta ro’yxatlar uchun siz Quick dan foydalanishingiz mumkin Saralash , va kichiklar uchun - Qo’shish Saralash , chunki ikkinchisi kichik hajmdagi ma’lumotlar uchun samaraliroq.

3.Saralashni qo‘shimchalar bilan birlashtirish (**Merge Insertion Sort** ***):*** Bu usul birlashtirib tartiblash va kiritish tartiblarini birlashtiradi. Bunday holda, ro’yxat quyi ro’yxatlarga bo’linadi, ular qo’shish tartibi yordamida tartiblanadi. Keyinchalik bu quyi ro’yxatlar birlashma tartiblash yordamida birlashtiriladi. Ushbu yondashuv kichik pastki ro’yxatlar uchun birlashma tartiblashning murakkabligini kamaytiradi.

Amaliy qism

1-topshiriq: Kombinatsiyalangan algoritm yordamida butun sonlar qatorini kamaymaydigan tartibda tartiblash.

Masalaning qo‘yilishi: Kombinatsiyalangan algoritm yordamida butun sonlar massivni kamaymaydigan tartibda tartiblang.

Masalaning ishlash algoritmi:

1. Massiv hajmini tekshiring.
2. Agar o’lcham 10 dan kichik bo’lsa, tez tartiblashdan foydalaning.

3. Aks holda biz barqaror birlashma tartiblashdan foydalanamiz.

Dastur ishlash jarayonida:

- butun sonlar vektoriga havolani kirituvchi ‘ hybridSort‘ funksiyasini yarating.

- funktsiya ichida biz vektorning o’lchamini tekshiramiz.

- agar o‘lcham 10 dan kichik bo‘lsa, tez saralash uchun “ std :: sort” dan foydalaning.

- aks holda, birlashtirish tartiblash uchun ‘ std :: stable_sort ‘ dan foydalaning.

```
#include <iostream>
#include <vektor>
#include <algoritm>
void hybridSort (std::vector<int>& arr ) {
   if ( arr.size () < 10) {
      std :: sort ( arr.begin (), arr.end ()); // Kichik massivlar uchun tez
tartiblashdan foydalaning
} else {
      std :: stable_sort ( arr.begin (), arr.end ()); // Katta massivlar uchun
birlashtirish tartibidan foydalaning
   }
}
int main() {
std::vector<int> arr = {5, 2, 9, 1, 7, 3, 8, 4, 6};
   hybridSort ( arr );
       (int raqami: arr ) {
      std :: cout << num << " ";
}
   return 0;
}
```

2-topshiriq: Kombinatsiyalangan algoritm yordamida satrlar massivini satr uzunligi bo‘yicha saralash.

Masalaning qo‘yilishi: Kombinatsiyalangan algoritm yordamida satrlar massivini satr uzunligi bo‘yicha tartiblang.

Masalaning ishlash algoritmi:

1. Satr massivining o’lchamini tekshiring.

2. Agar o’lcham 10 dan kichik bo’lsa, biz satr uzunligiga asoslangan maxsus taqqoslash funktsiyasi bilan tez tartiblashdan foydalanamiz.

3. Aks holda, biz bir xil taqqoslash funktsiyasi bilan barqaror birlashma tartiblashdan foydalanamiz.

Dastur ishlash jarayonida:

- satrlar vektoriga havolani kiritish sifatida qabul qiluvchi ‘ hybridSort‘ funksiyasini yarating.

- funktsiya ichida biz vektorning o’lchamini tekshiramiz.

- agar o’lcham 10 dan kichik bo’lsa, moslashtirilgan taqqoslash funksiyasi bilan ‘ std::sort‘ dan foydalaning.

- aks holda, bir xil taqqoslash funktsiyasi bilan ‘ std::stable_sort ‘ dan foydalaning.

```
#include <iostream>
#include <vektor>
#include <algoritm>
bool compareLength (const std::string & str1, const std::string & str2) {
return str1.length() < str2.length();
}
void hybridSort (std::vector<std::string>& arr ) {
if ( arr.size () < 10) {
std::sort( arr.begin (), arr.end (), compareLength ); // Foydalanishda tez
tartiblash uchun kichik massivlar
} else {
std:: stable_sort ( arr.begin (), arr.end (), compareLength ); // Foydalanishda
tartiblash birlashish uchun katta massivlar
}
}
int main() {
std::vector<std::string> arr = {"olma", "banan", "kivi", "apelsin", "uzum"};
   hybridSort ( arr );
(const std::string& str : arr ) {
      std :: cout << str << " ";
}
   return 0;
}
```

3-topshiriq: Birlashtirilgan algoritmdan foydalanib, maydonlar bo’yicha tuzilmalar qatorini saralash.

Masalaning qo‘yilishi: Birlashtirilgan algoritm yordamida tuzilmalar massivni maydonlar bo‘yicha saralang.

Masalaning ishlash algoritmi:

1. Struktura massivining o'lchamini tekshiring.

2. Agar o'lcham 10 dan kichik bo'lsa, biz struktura maydoni bo'yicha moslashtirilgan taqqoslash funktsiyasi bilan tez saralashdan foydalanamiz.

3. Aks holda, biz bir xil taqqoslash funktsiyasi bilan barqaror birlashma tartiblashdan foydalanamiz.

Dastur ishlash jarayonida:

- inshootlar vektoriga havolani kirituvchi ‘ hybridSort‘ funksiyasini yarating.

- funktsiya ichida biz vektorning o'lchamini tekshiramiz.

- agar o'lcham 10 dan kichik bo'lsa, moslashtirilgan taqqoslash funksiyasi bilan ‘ std::sort‘ dan foydalaning.

- aks holda, bir xil taqqoslash funktsiyasi bilan ‘ std :: stable_sort ‘ dan foydalaning.

```
#include <iostream>
#o'z ichiga <vektor>
#include <algoritm>
struct Person {
std::string name;
int age;
};
bool compareAge (const Person & p1, const Person & p2) {
return p1.age < p2.age;
}
void hybridSort (std::vector<Person>& arr ) {
if ( arr.size () < 10) {
std::sort( arr.begin (), arr.end (), compareAge ); // Foydalanish tez tartiblash
uchun kichik massivlar
} else {
std:: stable_sort ( arr.begin (), arr.end (), compareAge ); // Foydalanish
tartiblash birlashish Uchun katta massivlar
}
}
int main() {
std::vector<Person> people = {{"Elis", 25}, {"Bob", 30}, {"Charli", 20},
{"David", 35}, {"Havo", 28}};
   hybridSort (people);
(const Person & p: people) {
std::cout << p.name << " " << p.age << std:: endl;
   }
```

```
    return 0;
}
```

4-topshiriq: Ikki o‘lchovli massivni kombinatsiyalangan algoritm yordamida qatorlar bo‘yicha saralash.

Masalaning qo‘yilishi: Birlashtirilgan algoritm yordamida ikki o‘lchovli massivni qatorlar bo‘yicha saralash.

Masalaning ishlash algoritmi:

1. Ikki o‘lchovli massivning har bir qatori uchun:

- chiziq o’lchamini tekshirish.

- agar o’lcham 10 dan kichik bo’lsa, biz tez tartiblashdan foydalanamiz.

- aks holda, biz barqaror birlashma tartiblashdan foydalanamiz.

Dastur ishlash jarayonida:

- ikki o’lchovli vektorga havolani kiritish sifatida qabul qiluvchi ‘hybridSort2D‘ funksiyasini yarating.

- vektorning har bir satri uchun:

- chiziq o’lchamini tekshirish.

- tegishli saralash usulini qo’llang: kichik satrlar uchun ‘ std :: sort ‘ va kattalar uchun ‘ std :: stable_sort ‘.

```
#include <iostream>
#include <vektor>
#include <algoritm>
void hybridSort2D(std::vector<std::vector<int>>& matrix) {
for (std::vector<int>& row : matrix) {
      if ( row.size () < 10) {
         std :: sort ( row.begin (), row.end ()); // Kichik massivlar uchun tez
tartiblashdan foydalaning
} else {
        std :: stable_sort ( row.begin (), row.end ()); // Katta massivlar uchun
birlashtirish tartibidan foydalaning
            }
       }
}
int main() {
std::vector<std::vector<int>> matrix = {{5, 2, 9}, {1, 7, 3}, {8, 4, 6}};
hybridSort2D (matrix);
for (const std::vector<int>& row : matrix) {
```

```
for (int num: row) {
std::cout << num << " ";
}
std::cout << std::endl;
}
return 0;
}
```

5-topshiriq: Massivning turli qismlari uchun birlashtirib tartiblash va qo‘shish tartibidan foydalangan holda raqamlar qatorini tartiblang.

Masalaning qo‘yilishi: Massivning turli qismlarida birlashtirib tartiblash va kiritish tartibidan foydalanib, raqamlar qatorini tartiblang.

Masalaning ishlash algoritmi:

1. Massiv hajmini tekshiring.

2. Agar o’lcham chegara qiymatidan kichik yoki teng bo’lsa, biz kiritish tartiblashidan foydalanamiz.

3. Aks holda, massivni ikki qismga ajratamiz va har bir qismga rekursiv ravishda birlashtirish tartibini qo’llaymiz.

Dastur ishlash jarayonida:

Tartiblanayotgan hududning vektori va chegaralariga havolani kirituvchi ‘ mergeInsertionSort ‘ funksiyasini yarating.

- funksiya ichida massiv bo’limining o’lchamini tekshiramiz.

- agar o’lcham chegaradan kichik yoki teng bo’lsa, qo’shish tartibini qo’llang (‘ std::sort‘).

- aks holda, massivni ikkiga bo’ling va har bir yarmiga rekursiv birlashma tartiblash (‘ std :: stable_sort‘) qo’llang.

- keyin tartiblangan yarmini yana bitta massivga birlashtiramiz.

```
#include <iostream>
#include <vektor>
#include <algoritm>
void mergeInsertionSort (std::vector<int>& arr , int left, int right) {
   const int INSERTION_THRESHOLD = 10; // Kiritish tartibidan
foydalanish chegarasi
   if (right - left <= INSERTION_THRESHOLD) {
std::sort( arr.begin () + chap, arr.begin () + right + 1); // Tartiblash
qo’shimchalar Uchun kichik pastki massivlar
} else {
```

```
int mid = left + (right - left) / 2;
        mergeInsertionSort ( arr , left, mid);
        mergeInsertionSort ( arr , mid + 1, right);
std:: inplace_merge ( arr.begin () + left, arr.begin () + mid + 1, arr.begin ()
+ right + 1);
         }
}
int main() {
std::vector<int> arr = {5, 2, 9, 1, 7, 3, 8, 4, 6};
   mergeInsertionSort ( arr , 0, arr.size () - 1);
for (int num: arr ) {
        std :: cout << num << " ";
}
   return 0;
}
```

Mustaqil bajarish uchun topshiriqlar.

1. Kombinatsiyalangan algoritm yordamida butun sonlar massivini tartiblang.

2. Satrlar massivini kombinatsiya usuli yordamida uzunligining kamayishiga qarab tartiblang.

3. Birlashtirilgan algoritmlar yordamida tartiblangan massivdagi eng katta sonni toping.

4. Birlashtirilgan algoritmlar yordamida kasr sonlar massivini tartiblang.

5. Massivning birlashtirilgan algoritmlar yordamida tartiblanganligini tekshiring.

6. Birlashtirilgan algoritmlar yordamida tartiblangan massivning medianasini toping.

7. Birlashtirilgan algoritmlar yordamida fazodagi nuqtalar massivni koordinata boshidan masofa bo‘yicha tartiblang.

8. Birlashtirilgan algoritmlar yordamida tartiblangan massivdagi eng kichik sonni toping.

9. Birlashtirilgan algoritmlar yordamida maxsus sinf ob’ektlari massivini tartiblang.

10. Birlashtirilgan algoritmlar yordamida tartiblangan massivning barcha elementlari yig‘indisini toping.

11. Birlashtirilgan usullardan foydalanib, manfiy sonlar massivni kamayish tartibida tartiblang.

12. Birlashtirilgan algoritmlar yordamida tartiblangandan so’ng massivda bir xil elementlar mavjudligini tekshiring.

13. Birlashtirilgan algoritmlar yordamida tartiblangan massivdagi elementlar sonini toping.

14. Birlashtirilgan algoritmlar yordamida 1 dan 100 gacha bo’lgan oraliqdagi raqamlar massivini tartiblang.

15. Kombinatsiyalangan usullar yordamida tartiblangandan keyin massiv palindrom ekanligini tekshiring.

16. Birlashtirilgan algoritmlar yordamida tartiblangan massivning barcha elementlarining o‘rtacha arifmetik qiymatini toping.

17. Birlashtirilgan algoritmlar yordamida tub sonlar massivini tartiblang.

18. Birlashtirilgan usullardan foydalangan holda tartiblashdan so’ng massivdagi etishmayotgan qiymatlar sonini toping.

19. Birlashtirilgan algoritmlar yordamida sanalar massivini xronologik tartibda tartiblang.

20. Kombinatsiyalangan usullar yordamida tartiblangandan keyin massiv arifmetik progressiya ekanligini tekshiring.

21. Birlashtirilgan algoritmlar yordamida manfiy sonlar massivini absolyut qiymatning o‘sish tartibida tartiblang.

22. Birlashtirilgan algoritmlar yordamida tartiblangan massivning qo‘shni elementlari orasidagi maksimal farqni toping.

23. Birlashtirilgan algoritmlar yordamida harflar massivini alifbo tartibida tartiblang.

24. Birlashtirilgan usullar yordamida saralashdan so’ng massiv ortib boruvchi ketma-ketlik ekanligini tekshiring.

25. Birlashtirilgan algoritmlar yordamida mutlaq qiymatdagi sonlar massivini kamayish tartibida tartiblang.