1 - AMALIY MASHG‘ULOT. *HISOBLASH MODELLARI, ALGORITMLARI VA ULARNING MURAKKABLIGI.*

Ishdan maqsad: Talabalarni hisoblash modellari, algoritmlari va ularning murakkabliklari bilan tanishtirish. C++ tilida algoritmlarga oid misollar yaratish.

Nazariy qism. Hisoblash modellari va algoritmlari zamonaviy axborot texnologiyalari dunyosining asosiy komponentlari hisoblanadi. Ular kompyuterlarning ishlashi, dasturiy ta’minot va turli xil muammolarni - oddiy matematik operatsiyalardan tortib murakkab ma’lumotlarni tahlil qilish va sun’iy intellektgacha bo’lgan muammolarni hal qilish uchun asos bo’ladi.

Bundan tashqari, hisoblash modellari va algoritmlarini tushunish yangi texnologiyalar va innovatsiyalarni ishlab chiqish uchun juda muhimdir. Bu bizga yanada samarali va kuchli dasturiy mahsulotlar yaratish va murakkab muammolarni tezroq va samaraliroq hal qilish imkonini beradi.

Hisoblash modellari va algoritmlari bilan tanishishdan oldin, keling, ushbu kursda ko’rib chiqiladigan mavzularni qisqacha ko’rib chiqaylik.

- Algoritmlar asoslari va ularning tasnifi. Biz algoritmlar nima ekanligini, ular qanday ishlashini va ularni turli yo’llar bilan qanday tasniflash mumkinligini ko’rib chiqamiz.

- Ma’lumotlar tuzilmalari. Kursning ushbu bo’limi massivlar, ro’yxatlar, daraxtlar va grafiklar kabi asosiy ma’lumotlar tuzilmalarini o’rganish va ularni turli muammolarni hal qilishda qo’llashga bag’ishlangan.

- Algoritmik tahlil. Biz algoritmlarning samaradorligini bajarish vaqti va resurslardan foydalanish nuqtai nazaridan baholashni o’rganamiz.

- Saralash va qidirish. Keling, turli xil saralash va qidirish algoritmlarini, ularning qo’llanilishi va samaradorligini ko’rib chiqaylik.

- Rekursiya va dinamik dasturlash. Rekursiya va dinamik dasturlash tushunchalari va ularni murakkab masalalarni yechishda qo‘llanilishini o‘rganamiz.

Hisoblash modeli - bu hisoblash tizimining ishlash tamoyillarini tavsiflovchi abstraksiya. Bu kompyuterlar va boshqa hisoblash qurilmalari qanday ishlashini tushunishga imkon beruvchi kontseptual model. Kompyuter fanida hisoblash modellarining rolini ortiqcha baholab bo’lmaydi. Ular algoritmlar, dasturiy ta’minot va texnik vositalarni ishlab chiqish, algoritmlarning samaradorligi va murakkabligini tahlil qilish uchun asos bo’lib xizmat qiladi.

Algoritm - muayyan masalani hal qilish uchun mo’ljallangan ko’rsatmalar ketma-ketligi. Bu dasturlash va kompyuter fanida asosiy tushunchadir, chunki u kompyuterda amalga oshirilishi mumkin bo’lgan muammolarni hal qilish usulidir. Dasturlashda algoritmlarning ahamiyatini e’tiborsiz qoldirib bo’lmaydi. Ular dasturiy ta’minotni yaratish uchun asos bo’lib xizmat qiladi, chunki har bir dastur oxir-oqibatda aniq muammolarni hal

qiladigan algoritmlar to'plamidir. Algoritmlarni bilish dasturchilarga massivlarni saralash va elementlarni qidirishdan tortib, ma'lumotlarni qayta ishlash va sun'iy intellekt kabi murakkabroq operatsiyalargacha bo'lgan keng ko'lamli muammolarni samarali hal qilish imkonini beradi.

Bundan tashqari, algoritmlar kompyuter fanida asosiy rol o'ynaydi, chunki ular bizga muammolarni hal qilishning turli usullarini tahlil qilish va solishtirish imkonini beradi. Algoritmlarni tushunish dasturlarning samaradorligini aniqlashga va muayyan muammolar uchun eng mos echimlarni tanlashga yordam beradi.

Asosiy algoritmlarga misollar:

- Saralash: massiv yoki ro'yxatdagi elementlarni raqamlar tartibi yoki alifbo tartibi kabi ma'lum bir mezon bo'yicha tartibga soluvchi algoritmlar. Misollar pufakchali tartiblash, birlashtirib tartiblash va tez saralash.

- Qidiruv: massiv yoki ma'lumotlar strukturasida berilgan elementni topadigan algoritmlar. Masalan, chiziqli qidiruv va ikkilik qidiruv.

- Grafik algoritmlari: grafiklar bilan ishlaydigan algoritmlar, masalan, eng qisqa yo'lni qidirish, chuqurlikdan birinchi va kenglikdan birinchi grafani o'tkazish, minimal kenglikdagi daraxtlarni qidirish va boshqalar.

Bu dasturlash va kompyuter fanida qo'llaniladigan asosiy algoritmlarning kichik ro'yxati. Turli xil muammolarni hal qilish uchun mo'ljallangan boshqa ko'plab algoritmlar mavjud. Algoritmning murakkabligi uni bajarish uchun zarur bo'lgan resurslar miqdorining o'lchovidir. Vaqt murakkabligi kirish ma'lumotlarining hajmiga qarab algoritmni bajarish uchun qancha vaqt ketishini baholaydi, kosmik murakkablik esa algoritmni bajarish uchun qancha xotira kerakligini baholaydi.

Algoritmning vaqt murakkabligi kiritilgan ma'lumotlarning hajmidan kelib chiqib, uni bajarish uchun zarur bo'lgan vaqtni aniqlaydi. Bu algoritm samaradorligini baholashga imkon beruvchi muhim xususiyatdir.

Algoritmning vaqt murakkabligini baholash odatda eng yomon holatda bajariladigan amallar sonini tahlil qilish orqali amalga oshiriladi. Vaqt murakkabligi katta O kabi asimptotik belgilar bilan ifodalanishi mumkin.

Vaqt murakkabligi har xil bo'lgan algoritmlarga misollar:

1. Chiziqli qidiruv:

- Vaqt murakkabligi: O(n) eng yomon holatda, Ō(1) eng yaxshi holatda.
- Tavsif: Kirish massivining har bir elementi bo'ylab o'zi qidirayotgan elementni topguncha takrorlanadi.

2. Ikkilik qidiruv:

- Vaqt murakkabligi: O(log n) eng yomon holatda, O(1) eng yaxshi holatda.
- Tavsif: tartiblangan massiv bilan ishlaydi, uni yarmiga bo'linadi va kerakli element topilguncha elementlarni tekshiradi.

3. Pufakcha tartiblash:

- Vaqt murakkabligi: O(n^2) eng yomon va o'rtacha holatda, O(n) eng yaxshi holatda.

- Tavsif: qo'shni elementlarni taqqoslash va almashtirish orqali massivni ko'p marta takrorlaydi.

4. Birlashtirish tartibi:

- Vaqt murakkabligi: O(n log n) eng yomon, o'rtacha va eng yaxshi holatlarda.

- Tavsif: Massivni ikki yarmiga ajratadi, ularni alohida tartiblaydi va keyin tartiblangan yarmini birlashtiradi.

5. Floydning eng qisqa yo'llarni topish algoritmi:

- Vaqt murakkabligi: O(n^3).

- Tavsif: Grafikdagi barcha cho'qqi juftlari orasidagi eng qisqa yo'llarni topish uchun dinamik dasturlashdan foydalanadi.

Ushbu algoritmlarning turli vaqt murakkabliklarini va ularning kirish ma'lumotlari hajmiga bog'liqligini ko'rsatadi. Algoritmlarning vaqt murakkabligini tushunish muammolarni hal qilishning eng samarali usullarini tanlash va dasturiy ta'minotning ishlashini baholash imkonini beradi.

Amaliy qism.

Bu yerda C++ tilida yechimlari, to'liq tavsiflari va xulosalari bilan "Algoritmlarning murakkabligi" mavzusidagi beshta misollar keltirilgan:

1-topshiriq: Massivdagi barcha elementlarning yig'indisini toping.

Masalaning qo'yilishi: Raqamlar massivini oladigan va ularning yig'indisini qaytaruvchi funksiya yozing.

Yechim:

```
#include <iostream>
using namespace std;
int sumArray (int arr [], int n) {
        int sum = 0;
         (int i = 0; i < n; ++ i ) {
        summa += arr [ i ];
}
return summa;
}
int main() {
        int arr [] = {1, 2, 3, 4, 5};
        int n = sizeof ( arr ) / sizeof ( arr [0]);
      cout << "Masiv elementlari yig'indisi: " << sumArray (arr, n) << endl;
      return 0;
}
```

Natija: Bu algoritm O(n) chiziqli murakkablikka ega, bu yerda n massivdagi elementlar soni. Barcha elementlarning yig'indisi massivdan bir marta o'tishda hisoblanadi.

2-topshiriq: Massivdagi eng kichik elementni toping.

Masalaning qo‘yilishi: Raqamlar massivini oladigan va eng kichik elementni qaytaruvchi funksiya yozing.

Yechim:

```
#include <iostream>
#include <climits>
using namespace std;
int findMin (int arr [], int n) {
        int min = INT_MAX;
        (int i = 0; i < n; ++ i) {
if (arr [ i ] < min) {
        min = arr [ i ];
        }
}
return min;
}
int main() {
        int arr [] = {5, 3, 8, 1, 9};
        int n = sizeof ( arr ) / sizeof ( arr [0]);
      cout << "Minimal element: " << findMin (arr, n) << endl;
      return 0;
}
```

Natija: Bu algoritm O(n) chiziqli murakkablikka ham ega, chunki u minimal elementni topish uchun massivni bir marta bosib o‘tadi.

3-topshiriq: Massivdagi manfiy elementlar sonini hisoblang.

Masalaning qo‘yilishi: Raqamlar massivini oladigan va manfiy elementlar sonini qaytaruvchi funksiya yozing.

Yechim:

```
#include <iostream>
using namespace std;
int countNegatives (int arr [], int n) {
        int soni = 0;
(int i = 0; i < n; ++ i) {
if ( arr [ i ] < 0) {
        count++;
        }
}
return soni;
}
int main() {
int arr [] = {5, -3, 8, -1, 9};
int n = sizeof ( arr ) / sizeof ( arr [0]);
  cout << "Salbiy elementlar soni: " << countNegatives ( arr , n) << endl ;
      return 0;
```

```
}
```

Natija: Bu algoritm O(n) chiziqli murakkablikka ham ega, chunki u massivni bir marta bosib o‘tadi va manfiy elementlar sonini hisoblaydi.

4-topshiriq: Massivni kamayish tartibida tartiblang.

Masalaning qo‘yilishi: Raqamlar massivini kamayish tartibida tartiblovchi funksiya yozing.

Yechim:

```
#include <iostream>
#include <algoritm>
using namespace std;
void sortKamayuvchi (int arr [], int n) {
sort( arr , arr + n, kattaroq<int>());
}
int main() {
int arr [] = {5, 3, 8, 1, 9};
int n = sizeof ( arr ) / sizeof ( arr [0]);
      sortKamayuvchi (arr, n);
      cout << "Masiv kamayish tartibida tartiblangan:";
(int i = 0; i < n; ++ i) {
          cout << " " << arr [ i ];
}
      cout << endl;
      return 0;
}
```

Natija: bu yerda algoritm (log n) murakkablikka ega, bu yerda n massivdagi elementlar soni.

5-topshiriq: Raqam tub ekanligini tekshiring.

Masalaning qo‘yilishi: Raqamni qabul qiladigan va agar u tub bo‘lsa, rost, aks holda noto‘g‘ri bo‘lsa, qaytaradigan funksiyani yozing .

Yechim:

```
#include <iostream>
using namespace std;
bool isPrime (int n) {
if (n <= 1) {
return false;
}
(int i = 2; i * i <= n; ++ i) {
if (n % i == 0) {
return false;
        }
}
return true;
}
```

```
int main() {
int soni = 29;
if ( Prime (son)) {
        cout << num << "asosiy hisoblanadi." << endl;
} else {
        cout << num << "asosiy emas." << endl;
}
     return 0;
}
```

Natija: Bu sonning birlamchiligini tekshirish algoritmi O(sqrt (n)) ga teng murakkablikka ega, bunda n tekshirilayotgan sondir.

Mustaqil bajarish uchun topshiriqlar

1. Saralanmagan massivdagi eng kichik elementni toping.
2. Massivdagi barcha elementlarning yig‘indisini toping.
3. Birlashtirish usuli yordamida butun sonlar massivini tartiblang.
4. Raqam tub ekanligini tekshiring.
5. Massivda ma’lum bir elementning paydo bo‘lish sonini hisoblang.
6. Substrat satrda mavjudligini tekshiring.
7. Ikkilik qidiruv usuli yordamida tartiblangan massivdagi elementni qidirish.
8. Massivdagi elementlarning o‘rtacha qiymatini toping.
9. Massivdagi musbat elementlar sonini hisoblang.
10. Massivdagi inversiyalar sonini hisoblang.
11. Massivdagi maksimal elementni topish.
12. Massivdagi ikkita sonning eng katta ko‘paytmasini toping.
13. Massiv qatorlarini leksikografik tartibda tartiblang.
14. Massivdagi eng kichik k sonni toping.
15. Massiv palindrom ekanligini tekshiring.
16. Pufakcha usuli yordamida butun sonlar massivini tartiblang.
17. Kvadrat matritsaning bosh diagonalidagi elementlar yig‘indisini toping.
18. Satr boshqa satrning anagrammasi ekanligini tekshiring.
19. Butun sonlar massivining medianasini toping.
20. Massivning o’sish tartibida tartiblanganligini tekshiring.
21. Massivni kamayish tartibida tartiblang.
22. Berilgan n sonigacha tub sonlar sonini sanang.
23. Satrda barcha noyob belgilar mavjudligini tekshiring.
24. Ikki sonning eng katta umumiy bo‘luvchisini toping.
25. Satrning palindrom ekanligini tekshiring.