1-1KIDT-21-guruh talabasi

Mamatov Otamurodning

Algoritm va berilganlar strukturasi

Fanidan Mustaqil ishi

Chiziqli algoritmlari. Binar qidiruv algoritmi.

Chiziqli izlash algoritmi (gradient descent) ma'lumotlar tahlili va o'rganishda keng ishlatiladigan algoritm hisoblanadi. Bu algoritm, bir funksiya minimumini topish uchun ishlatiladi. Chiziqli izlash algoritmi, funksiyaning gradientini hisoblash yordamida ishlaydi va undan foydalanib, funksiyaning minimum qiymatiga yaqinroq bo'lgan nuktani aniqlashga yordam beradi.

Bir funksiyaning gradienti, o'nlik funksiyaning har bir o'zgaruvchisiga nisbatan qanday o'zgarishi kuzatiladi. Chiziqli izlash algoritmi, funksiyaning qiymati o'zgarishining eng katta ko'rsatkichini topishga harakat qiladi va funksiyaning minimum qiymatiga yaqinroq bo'lgan nuktani aniqlashga harakat qiladi.

Binar qidiruv algoritmi (binary search) esa, bir tartiblangan ro'yxatda kerakli elementni qidirish uchun ishlatiladigan algoritm hisoblanadi. Bu algoritm, ro'yxatdagi elementlarni tartiblash talab qiladi.

Binar qidiruv algoritmi, ro'yxatning o'rta elementini tanlaydi va qidiruvning kerakli bo'lagini tanlash uchun o'rtasidagi elementni tekshiradi. Agar kerakli element o'rta elementdan kichik bo'lsa, algoritm o'rta elementdan oldingi bo'limni qidirishga harakat qiladi. Aks holda, o'rta elementdan keyingi bo'limni qidirishga harakat qiladi. Binar qidiruv algoritmi, kerakli elementni topish uchun eng yaxshi algoritmlardan biri hisoblanadi, chunki u kerakli elementni topishda juda tez ishlaydi.

\*Chiziqli izlash algoritmi (Gradient Descent Algorithm)\*\*

**1. Funksiyaning minimum qiymatini topish:**

Gradient Descent algoritmi, funksiyaning minimum qiymatini topish uchun ishlatiladi. Bu algoritm, gradientni hisoblash, funksiyani o'zgaruvchilarga nisbatan yonlendirmoq va funksiyani minimal qiymatga qarab ishlatish orqali ishlaydi.

Sizning so'rovingiz uchun, C++ da funksiyaning minimum qiymatini topish uchun bir nechta algoritmlar mavjud. Men sizga "Gradient Descent" algoritmini tavsiya qilaman, bu algoritm funksiyaning minimum qiymatini topish uchun juda kuchli va oddiy bir algoritm hisoblanadi.

Gradient Descent algoritmi funksiyaning minimum qiymatini topish uchun qo'llaniladigan eng oddiy algoritmlardan biri hisoblanadi. Bu algoritm, funksiyaning qiymatini kamaytirib, funksiyaning minimum qiymatiga yaqinlashadi. Algoritmda funksiyaning qiymati va uning gradienti (yani, funksiyaning qaysi yo'nalishda kamayish ko'rsatishi) hisoblanadi. Gradientni hisoblash uchun, funksiyaning har bir muhitdagi qiymatlari olinadi. Gradientni olishdan keyin, funksiyani kamaytirishda kerak bo'lgan qadam aniqlanadi va funksiyaning qiymati yangi qiymatga yaqinlashadi. Bu jarayon, funksiyaning minimum qiymatiga yaqinlashishga yordam beradi.

Algoritmda funksiyaning qiymatini kamaytirish va gradientni hisoblash uchun matematik formulalaridan foydalaniladi. C++ da Gradient Descent algoritmini yozish uchun, funksiyani yozing va gradientni hisoblash uchun matematik formulalarni yozing. So'ng, gradientni hisoblash va funksiyani kamaytirish jarayonini yozing. Algoritmda qadam qadamiga yaqinlashish uchun, har bir qadamda gradientni hisoblash va funksiyani kamaytirish kerak bo'ladi. Bunda, algoritmning to'g'ri ishlashi uchun, qadam qadamiga kamayish koeffitsiyenti aniqlanadi. Bu koeffitsiyentni o'zgartirib, algoritmdan eng yaxshi natijalarni olish mumkin.

**2. Gradientning hisoblanishi:**

Gradient, funksiyani o'zgaruvchilarga nisbatan necha bosqichda o'zgarishi haqida ma'lumot beradi. Gradientning hisoblanishi uchun, chiziqli izlash algoritmi o'zgaruvchilarni belgilash va funksiyani hisoblash kerak.

3. Chiziqli izlash algoritmini amalga oshirish:

Chiziqli izlash algoritmini amalga oshirish uchun, boshlang'ich nuqta tanlash kerak va gradientni hisoblash kerak. Gradientni hisoblaganidan so'ng, o'zgaruvchilarni yangilash va funksiyani hisoblash kerak. Yangilash jarayoni, gradientni ishlatib, funksiyani minimal qiymatga yaqinlashishga qaratiladi.

\*\*Binar qidiruv algoritmi (Binary Search Algorithm)\*\*

1. Tartiblash:

Binar qidiruv algoritmi, tartiblangan ro'yxatda element qidirish uchun ishlatiladi. Buning uchun, ro'yxat elementlari tartiblangan holatda bo'lishi kerak.

**2. O'rta elementni tanlash:**

Ro'yxatdagi o'rta elementni tanlash uchun, ro'yxatni yarim yarim bo'lish kerak.

O'rta arifmetikni topish uchun, A va B ni yig'indisini olib, yig'indini 2 ga bo'lib, natijani ekranga chiqaring. Bu usul quyidagicha yozilishi mumkin:

#include <iostream>

using namespace std;

int main() {

int A, B;

double mean;

cout << "A ni kiriting: ";

cin >> A;

cout << "B ni kiriting: ";

cin >> B;

mean = (A + B) / 2.0;

cout << "O'rta arifmetik: " << mean << endl;

return 0;

}

Bu dastur foydalanuvchidan A va B sonlarini kiritishni so'raydi, keyin A va B ni yig'indisini 2 ga bo'lib, o'rta arifmetikni hisoblaydi va natijani ekranga chiqaradi.

3. Qidirilayotgan elementni tekshirish:

Qidirilayotgan elementni tekshirish uchun, o'rta elementni qidirilayotgan element bilan solishtirish kerak.

4. Qidirilayotgan elementni topish:

Qidirilayotgan elementni topish uchun, ro'yxatni yarim yarim bo'lish va qidirilayotgan elementni tekshirish jarayonini takrorlash kerak.

Binar qidiruv algoritmi, saralanmagan ro'yxatda kerakli elementni topish uchun ishlatiladigan oddiy algoritm hisoblanadi. Ushbu algoritmni ishlatish uchun ro'yxatni qidirib topish kerak bo'lgan elementlar to'g'ridan-to'g'ri saralgan bo'lishi kerak. Masalan, ro'yxatni kichikdan katta bo'lgan tartibda saralash mumkin.

Binar qidiruv algoritmi quyidagicha ishlaydi:

1. Ro'yxatni saralang.

2. Qidirayotgan elementni ro'yxatning o'rtasidan boshlang.

3. Ro'yxatning o'rtasidagi elementni tekshirib, qidirayotgan elementdan katta yoki kichik bo'lsa, qidirayotgan elementni o'rtasidan o'ng tomoniga ko'chirib yuboriladi.

4. Agar qidirayotgan element katta bo'lsa, qidirayotgan elementni o'rtasidan chap tomoniga ko'chirib yuboriladi.

5. Qidirayotgan element ro'yxatda topilguncha yoki ro'yxatni ko'proq buzilguncha qilishdan to'xtatiladi.

**Binar qidiruv algoritmi** o'nlik, ikkilik yoki boshqa sonlar ustida ishlaydi. Ushbu algoritmni amalga oshirish uchun, ro'yxat saralgan bo'lishi kerak. Bunday holatda, birinchi qadamda ro'yxatni saralash kerak bo'ladi.

Quyidagi kodda, C++ tilida binar qidiruv algoritmi ko'rsatilgan:

#include <iostream>

using namespace std;

int binarySearch(int arr[], int n, int x) {

int left = 0;

int right = n - 1;

while (left <= right) {

int mid = (left + right) / 2;

if (arr[mid] == x) {

return mid;

}

if (arr[mid] < x) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return -1;

}

int main() {

int arr[] = {2, 5, 10, 13, 18, 20};

int n = sizeof(arr) / sizeof(arr[0]);

int x = 13;

int result = binarySearch(arr, n, x);

if (result == -1) {

cout << "Element topilmadi" << endl;

} else {

cout << "Element indeksi: " << result << endl;

}

return 0;

}

Bu kodda, "binarySearch" funksiyasi ro'yxat, ro'yxat uzunligi va qidirayotgan elementni qabul qiladi. Algoritmda ro'yxatni saralash kerak, shuning uchun "arr" ro'yxatida elementlar kichikdan katta saralgan. Dasturda "n" o'zgaruvchisi ro'yxat uzunligini saqlaydi. "x" o'zgaruvchisi esa qidirayotgan elementni anglatadi.

Funksiyada, qidirayotgan elementni o'rtasiga qo'yilgan "mid" indeksi hisoblanadi. Agar ro'yxatdagi "mid" element qidirayotgan elementga teng bo'lsa, "mid" indeksi qaytariladi. Agar "mid" element qidirayotgan elementdan kichik bo'lsa, qidirayotgan elementni o'rtasidan o'ng tomoniga ko'chirib yuboriladi. Aks holda, qidirayotgan elementni o'rtasidan chap tomoniga ko'chirib yuboriladi.

Keyingi qadamda, funksiya "result" o'zgaruvchisida qidirayotgan elementning indeksi saqlanadi. Agar qidirayotgan element topilmagan bo'lsa, "result" o'zgaruvchisiga "-1" qiymati tayinlanadi. Natijada, "if" va "else" operatorlari yordamida, natija ekranga chiqariladi.

"Hello Uzbekistan" so'zining h harfini binar qidiruv tizimi yordamida topish uchun quyidagi kodni ishlatishingiz mumkin:

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

string s = "Hello Uzbekistan";

int n = s.size();

sort(s.begin(), s.end()); // ro'yxatni saralash

int l = 0, r = n-1, mid;

while (l <= r) {

mid = (l + r) / 2;

if (s[mid] == 'h') {

cout << "h harfi indeksi: " << mid << endl;

break;

}

else if (s[mid] < 'h')

l = mid + 1;

else

r = mid - 1;

}

return 0;

}

Ushbu kodda, "Hello Uzbekistan" so'zi ro'yxatga saqlanadi va ro'yxat saralandi. Keyin, binar qidiruv algoritmi yordamida 'h' harfi qidiriladi. Harf topilganda, shu harfning indeksi konsolga chiqariladi. Agar 'h' harfi ro'yxatda topilmagan bo'lsa, yuqorida ko'rsatilgan binar qidiruv algoritmlaridan biri ishlatilishi mumkin.

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

int a[] = {2, 4, 3, 1, 6};

int b[] = {7, 6, 5, 4, 3};

int c[] = {1, 2, 3, 4, 5, 6};

int n = sizeof(a) / sizeof(a[0]);

sort(c, c + 6); // c ro'yxatini saralas

int l = 0, r = n-1, mid;

while (l <= r) {

mid = (l + r) / 2;

if (c[mid] == a[mid] + b[n-mid-1]) {

cout << "a va b ro'yxatlari " << mid+1 << "-chi elementdan ajratilgan holda tartiblangan" << end

break;

}

else if (c[mid] < a[mid] + b[n-mid-1])

l = mid + 1;

else

r = mid - 1;

}

return 0;

}

Ushbu kodda, berilgan ro'yxatlardan a va b ro'yxatlari yig'indisini c ro'yxati deb qabul qilamiz. Keyin, c ro'yxatini saralaymiz. Binar qidiruv algoritmi yordamida, a va b ro'yxatlari ajratilgan holda tartiblangan to'plamni topishga harakat qilamiz. Topilganda, tartiblangan indeks konsolga chiqariladi.

Agar a va b ro'yxatlari ajratilgan holda tartiblangan to'plam topilmagan bo'lsa, yuqorida ko'rsatilgan binar qidiruv algoritmlaridan biri ishlatilishi mumkin.

#include <iostream>

#include <algorithm>

using namespace std;

int binary\_search(int arr[], int n, int x) {

int low = 0, high = n - 1;

while (low <= high) {

int mid = (low + high) / 2;

if (arr[mid] == x) {

return mid;

} else if (arr[mid] < x) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return -1;

}

int main() {

int a[] = {9, 8, 7, 6, 5};

int b[] = {3, 4, 5, 6, 7};

int c[] = {4, 7, 2, 5, 6};

int n = sizeof(c) / sizeof(c[0]);

sort(c, c + n);

int index\_a = binary\_search(c, n, a[0]);

int index\_b = binary\_search(c, n, b[0]);

cout << "a to'plamidagi eng kichik element tartibida " << index\_a << " indeksda joylashgan." << endl;

cout << "b to'plamidagi eng kichik element tartibida " << index\_b << " indeksda joylashgan." << endl;

return 0;

}

Bu kodda, binary\_search funksiyasi ro'yxat, ro'yxat uzunligi va qidirilayotgan elementni qabul qiladi. Ushbu funksiya binar qidiruv algoritmini ishlatadi va qidirilayotgan elementni topganda, indeksni qaytaradi. Agar element topilmagan bo'lsa, -1 qiymatini qaytaradi.

main funksiyasida, a, b va c ro'yxatlari berilgan. n o'zgaruvchisi c ro'yxati uzunligini saqlaydi. c ro'yxatini sort funksiyasi orqali tartiblash kerak. binary\_search funksiyasi yordamida, a va b ro'yxatidagi eng kichik elementlar c ro'yxatida qayerda joylashganligi aniqlanadi va natijada indekslar chiqariladi.

Natijada, programma a va b ro'yxatidagi eng kichik elementlarning tartibida joylashgan indekslarini chiqaradi.

#include <iostream>

#include <algorithm>

using namespace std;

int binary\_search(int arr[], int n) {

int low = 0, high = n - 1;

while (low < high) {

int mid = (low + high + 1) / 2;

if (arr[mid] <= arr[high]) {

low = mid;

} else {

high = mid - 1;

}

}

return arr[high];

}

int main() {

int A[] = {2, 3, 6, 8, 9};

int n = sizeof(A) / sizeof(A[0]);

int max\_element = binary\_search(A, n);

cout << "Eng katta element: " << max\_element << endl;

return 0;

}

binary\_search funksiyasi, ro'yxatni va ro'yxat uzunligini qabul qiladi. Ushbu funksiya binar qidiruv algoritmini ishlatadi va ro'yxatdagi eng katta elementni topadi.

main funksiyasida, A ro'yxati berilgan va n o'zgaruvchisi ro'yxat uzunligini saqlaydi. binary\_search funksiyasi yordamida, A ro'yxatidagi eng katta element aniqlanadi. Natijada, dastur eng katta elementni chiqaradi.

#include <iostream>

#include <algorithm>

using namespace std;

int binary\_search(int arr[], int n) {

int low = 0, high = n - 1;

while (low < high) {

int mid = (low + high) / 2;

if (arr[mid] >= arr[low]) {

low = mid + 1;

} else {

high = mid;

}

}

return arr[low];

}

int main() {

int A[] = {2, 3, 6, 8, 9};

int n = sizeof(A) / sizeof(A[0]);

int min\_element = binary\_search(A, n);

cout << "Eng kichik element: " << min\_element << endl;

return 0;

}

A = [2,3,6,8,9] berilgan ro'yxatdagi elementlar sonini toping

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

int A[] = {2, 3, 6, 8, 9};

int n = sizeof(A) / sizeof(A[0]);

int min = A[0];

int max = A[n-1];

cout << "Minimum element: " << min << endl;

cout << "Maximum element: " << max << endl;

return 0;

}

Quyidagi kod C++ dasturlash tilida berilgan A=26 va B=20 sonlarining o'rta qiymatini topadi (binar qidiruv algoritmi orqali):

#include <iostream>

using namespace std;

double findMid(double a, double b) {

return (a + b) / 2.0;

}

double findAverage(double a, double b) {

double mid = findMid(a, b);

return mid;

}

int main() {

double A = 26;

double B = 20;

double mid = findAverage(A, B);

cout << "The average of " << A << " and " << B << " is " << mid << endl;

return 0;

}

double findMid(double a, double b) funksiyasi 2 ta sonning o'rta qiymatini topish uchun ishlatiladi.

double findAverage(double a, double b) funksiyasi esa findMid funksiyasidan foydalanib, 2 ta sonning o'rta qiymatini topadi.

double A = 26; va double B = 20; qiymatlar beriladi.

double mid = findAverage(A, B); findAverage funksiyasi yordamida A va B o'rtasidagi qiymat topiladi va mid o'zgaruvchiga saqlanadi.

cout << "The average of " << A << " and " << B << " is " << mid << endl; o'rta qiymat chiqariladi.

Natijada, kod A=26 va B=20 sonlari uchun o'rta qiymatni hisoblaydi va konsolga chiqaradi.