Лаб№2.1

So'nggi darsda biz oddiy algoritm - chiziqli qidiruv yordamida qanday qilib ma'lum bir kalitni topishni o'rgandik. Shuningdek, biz ushbu qo'llanmada ketma-ket qidirish ob'ektni topish uchun bir xil algoritm emasligini eslatib o'tdik. Ushbu algoritmlardan biri bu ikkilik qidirish. Bu ikkilik qidirish deb ham ataladi.

Bugun biz uni o'rganamiz. Shuningdek, biz ushbu kabi savollarga javob berishga harakat qilamiz: ikkilik qidiruv dasturda qanday ishlaydi, uni sizning kodingizda ishlatishning ijobiy va salbiy tomonlari va qaysi ma'lumotlar tuzilmalarida foydalanishingiz mumkin va qaysi birida yo'q.

Ikkilik qidiruv nima?

Ikkilik qidirish - bu allaqachon bajarilgan qatorda ma'lum bir qiymatga ega bo'lgan elementni topadigan oddiy dasturga ega bo'lgan juda tez algoritm.

Yodda tutish juda muhim! Algoritm faqat tartiblangan massiv bilan to'g'ri ishlaydi. Va agar tasodifan siz uni ishlatishdan oldin qatorni saralashni unutgan bo'lsangiz, unda ko'p hollarda algoritm hisoblangan javob noto'g'ri bo'ladi.

C ++ da qanday qilib ikkilik qidiruvni yaratish

Keling, ikkilik qidiruvni misol bilan qanday ishlashini ko'rib chiqaylik, quyida keltirilgan misolda, 9-qatorda biz 10 ta elementdan iborat massiv massivini yaratdik va 12-qatorda foydalanuvchidan o'z katakchalarini klaviaturadan to'ldirishini so'radik.

20-qatorda biz foydalanuvchidan kalitni kiritishni so'raymiz (uni massivda topish kerak bo'ladi), so'ngra foydalanuvchi tomonidan kiritilgan kalit mavjudligini ikkilik qidirish bilan qatorni tekshiramiz. Agar biz kalitni massivdan topsak, u holda kalit joylashgan katakning indeksini chiqaramiz

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | #include <iostream>  #include <algorithm>    using namespace std;    int main() {    setlocale(LC\_ALL, "rus");      int arr[10]; // создали массив на 10 элементов    int key; // создали переменную в которой будет находиться ключ      cout << "Введите 10 чисел для заполнения массива: " << endl;      for (int i = 0; i < 10; i++) {      cin >> arr[i]; // считываем элементы массива    }      sort (arr, arr + 10); // сортируем с помощью функции sort (быстрая сортировка)      cout << endl << "Введите ключ: ";      cin >> key; // считываем ключ      bool flag = false;    int l = 0; // левая граница    int r = 9; // правая граница    int mid;      while ((l <= r) && (flag != true)) {      mid = (l + r) / 2; // считываем срединный индекс отрезка [l,r]        if (arr[mid] == key) flag = true; //проверяем ключ со серединным элементом      if (arr[mid] > key) r = mid - 1; // проверяем, какую часть нужно отбросить      else l = mid + 1;    }      if (flag) cout << "Индекс элемента " << key << " в массиве равен: " << mid;    else cout << "Извините, но такого элемента в массиве нет";      system("pause");    return 0;  } |

Ikkilik qidiruv 29-35 qatorlarda. 27-qatorda biz o'rta o'zgaruvchini yaratdik, u o'rta elementning indeksini ushlab turadi ([l, r] segmentidan). 27-qatorda biz segmentning o'rta elementini [l, r] o'zgaruvchiga, quyidagi formula bo'yicha o'qiymiz: (l + r) / 2 (unda l chap chegara, r o'ng chegara). 29-qatorda biz arr[mid] > keyshartini tekshiramiz:

Agar arr [mid] tugmachadan katta bo'lsa, u holda r qiymatiga mid beriladi. Chunki yuqori qismini tekshirish mantiqsiz, chunki kalit faqat o'rta indeks ostidagi kataklarda bo'lishi mumkin (agar bu massiv o'sish tartibida tartiblangan bo'lsa).

Agar arr [mid] tugmachadan kichik bo'lsa, u holda l qiymatiga mid beriladi. Chunki pastki qismni tekshirish mantiqsiz, chunki kalit faqat o'rta indeks ustidagi kataklarda joylashgan bo'lishi mumkin (agar bu massiv o'sish tartibida tartiblangan bo'lsa).

37-qatorda biz bayroqni tekshiramiz:

Agar bayroq rost bo'lsa, demak biz kalitni topdik, demak biz "massivdagi asosiy element indeksining o'rtasi" ni chiqaramiz.

Agar bayroq noto'g'ri bo'lsa, biz "Kechirasiz, lekin massivda bunday element yo'q" ni chiqaramiz.

Keling, ushbu kodni ishga tushiramiz:

bin\_search.cpp

**Напишите 10 чисел для заполнения массива:**

**17 20 26 31 44 54 55 65 77 93**

**Введите ключ : 54**

**Индекс элемента 54 в массиве равен: 5**

**Process returned 0 (0x0) execution time : 0.005 s**

**Press any key to continue.**

Agar qatorda bunday kalit bo'lmasa:

bin\_search.cpp

**Напишите 10 чисел для заполнения массива:**

**17 20 26 31 44 54 55 65 77 93**

**Введите ключ : 18**

**Извините, но такого элемента в массиве нет**

**Process returned 0 (0x0) execution time : 0.005 s**

**Press any key to continue.**

Ikkilik qidiruvni qanday yaxshilash mumkin

Ikkilik qidiruvni boshqa barcha algoritmlar singari optimallashtirish mumkin. Uni optimallashtirish uchun while holatini kamaytirish mumkin:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | while (l < r) {      mid = (l + r) / 2; // считываем срединный индекс отрезка [l,r]        if (arr[mid] > key) r = mid; // проверяем, какую часть нужно отбросить с поиска      else l = mid + 1;  }    r--; // уменьшаем на один    if (arr[r] == key) cout << "Индекс элемента " << key << " в массиве равен: " << r;  else cout << "Извините, но такого элемента в массиве нет"; |

Amalga oshirilgandan so'ng, r - 1 o'zgaruvchisi elementning soniga teng bo'ladi, agar bunday element, albatta, massivda bo'lsa. Aks holda, agar biror element bo'lmasa, unda r qatorda qator o'zgarishini kiritishingiz mumkin bo'lgan joy bo'ladi (bu printsip ikkilik qo'shimchalar bo'yicha saralashda ishlatiladi).

8-qatorda r ni oldindan birma-bir kamaytiring.

Ikkilik qidiruvdan foydalanishning ijobiy va salbiy tomonlari:

Algoritmni amalga oshirish etarli darajada oson;

Algoritmning tezkor ishlashi;

Kamchiliklari:

Massivni qidirish uchun buyurtma berish (saralash) kerak;

Ikkilik qidiruv har qanday ma'lumot elementiga (indeks bo'yicha) murojaat qilishi kerak. Bu shuni anglatadiki, bog'langan ro'yxatlarga asoslangan barcha ma'lumotlar tuzilmalaridan foydalanish mumkin emas;

Massivlar yoki vektorlar bilan ishlashda doimo ikkilik qidiruvdan foydalanishni maslahat beramiz. Odatda dasturchilar funktsiyalar yordamida ikkilik qidiruvdan foydalanadilar.