BINARY SEARCH QIDIRISH ALGORITMI

АЛГОРИТМ БИНАРНОГО ПОИСКА

BINARY SEARCH ALGORITHM

Abdullayev Shaxboz Solijon o‘g‘li
Shaxbozfardu2023@gmail.com
FarDU Axborot texnologiyalari kafedrasi katta o‘qituvchisi
Jo’rayeva Madinabonu Abdukarim qizi
Farg‘ona davlat universiteti
Axborot tizimlariva texnologiyalari yo‘nalishi 1-kurs talabasi
madinabonuj888@gmail.com

Annotatsiya: *Mazkur maqolada Binary Search (ikkilik qidiruv) algoritmining nazariy va amaliy asoslari yoritilgan. Ikkilik qidiruv — bu tartiblangan massiv yoki ro‘yxatda ma’lum bir elementni topish uchun ishlatiladigan samarali algoritm bo‘lib, uning ishlash murakkabligi O(log n) ga teng. Maqolada algoritmning ishlash prinsipi, uni dasturlash tillarida implementatsiya qilish usullari, hamda chiziqli qidiruv bilan solishtirilgan holdagi afzalliklari va cheklovlari tahlil qilinadi. Shuningdek, algoritmning real hayotdagi qo‘llanilishi va uning optimallashtirish imkoniyatlari haqida ham ma’lumotlar keltirilgan.*

Аннотация: *В данной статье рассматриваются теоретические и практические основы алгоритма бинарного поиска (Binary Search). Бинарный поиск — это эффективный алгоритм для нахождения элемента в отсортированном массиве или списке, обладающий логарифмической сложностью O(log n). В работе подробно описан принцип действия алгоритма, методы его реализации на различных языках программирования, а также проведено сравнение с линейным поиском по эффективности и ограничениям. Кроме того, рассмотрены примеры применения алгоритма в реальных задачах и возможности его оптимизации.*

Abstract: *This article explores the theoretical and practical foundations of the Binary Search algorithm. Binary Search is an efficient algorithm used to locate a*

specific element in a sorted array or list, with a time complexity of O(log n). The paper outlines the working principle of the algorithm, implementation techniques in various programming languages, and provides a comparative analysis with linear search in terms of performance and limitations. Furthermore, real-world applications and potential optimization strategies of the algorithm are also discussed.

Kalit so‘zlar: *Binary Search, ikkilik qidiruv, tartiblangan massiv, qidiruv algoritmlari, algoritmik samaradorlik, logarifmik murakkablik, iteratsiya, rekursiya, massivlar bilan ishlash, chiziqli qidiruv, optimallashtirish, real hayotdagi qo‘llanma, algoritm tahlili.*

Ключевые слова: *Бинарный поиск, отсортированный массив, алгоритмы поиска, алгоритмическая эффективность, логарифмическая сложность, итерация, рекурсия, работа с массивами, линейный поиск, оптимизация, практическое применение, анализ алгоритмов.*

Keywords: *Binary Search, sorted array, search algorithms, algorithmic efficiency, logarithmic complexity, iteration, recursion, array manipulation, linear search, optimization, real-world application, algorithm analysis.*

KIRISH

Axborot texnologiyalari jadal rivojlanib borayotgan hozirgi davrda katta hajmdagi ma’lumotlar bilan samarali ishlash muhim vazifalardan biriga aylangan. Dasturlashda ayniqsa tezkorlik talab qilinadigan muhitlarda ma’lumotlar ichidan kerakli elementni izlash (qidirish) operatsiyasi ko‘p uchraydi. Bu kabi operatsiyalarni optimal bajarish uchun turli algoritmlar mavjud bo‘lib, ular orasida Binary Search (ikkilik qidiruv) algoritmi o‘zining yuqori samaradorligi bilan ajralib turadi.

Binary Search algoritmi faqat oldindan tartiblangan massiv yoki ro‘yxatlar bilan ishlaydi. Uning asosiy g‘oyasi — izlanayotgan elementni topish uchun har safar qidiruv oraliq diapazonini yarmiga qisqartirib borishdir. Bu yondashuv O(log n) murakkablikka ega bo‘lib, chiziqli qidiruv (O(n))ga nisbatan ancha tez va resurs tejamkor hisoblanadi. Masalan, bir million elementdan iborat massivda kerakli elementni Binary Search yordamida bor-yo‘g‘i 20 ga yaqin taqqoslash bilan topish mumkin.

Mazkur algoritmning afzalliklari nafaqat nazariy, balki amaliy dasturlashda ham o‘z aksini topadi. Dasturiy ta’minotlarda, bazaviy ma’lumotlar tuzilmalari, qidiruv

tizimlari, foydalanuvchi interfeyslari, o‘yinlar va hatto sun’iy intellekt algoritmlarida Binary Search keng qo‘llaniladi. Shu sababli, ushbu algoritmni chuqur o‘rganish nafaqat algoritmik fikrlashni rivojlantiradi, balki dasturiy mahsulotlarni optimallashtirishda muhim ahamiyat kasb etadi.

Ushbu maqolada Binary Search algoritmining ishlash tamoyillari, uning iterativ va rekursiv implementatsiyalari, boshqa qidiruv algoritmlari bilan taqqoslanishi, afzalliklari va cheklovlari, shuningdek, real hayotdagi qo‘llanilish holatlari atroflicha tahlil qilinadi. Maqola dasturlashni yangi o‘rganayotgan talabalar, algoritmshunoslar hamda tizimli dasturlovchilar uchun foydali nazariy va amaliy bilim manbai bo‘lib xizmat qiladi.

BINARY SEARCH ALGORITMI VA UNING ASOSIY ELEMENTLARI

Zamonaviy dasturlash tizimlarida ma’lumotlarga ishlov berish tezligi muhim omillardan biri hisoblanadi. Ma’lumotlar ustida tezkor izlash (qidiruv) amallarini bajarish uchun turli xil algoritmlar ishlab chiqilgan. Ulardan eng samaralilaridan biri bu — Binary Search (ikkilik qidiruv) algoritmidir. Bu algoritm ma’lumotlar tartiblangan bo‘lsa, izlanayotgan elementni juda tez va ishonchli topish imkonini beradi.

Binary Search algoritmi asosida "bo‘lish va izlash" (divide and conquer) strategiyasi yotadi. Ya’ni, algoritm har bir bosqichda qidiruv oraliq diapazonini yarmiga qisqartirib boradi. Shu tariqa izlanayotgan elementning joylashgan oraliq hududi minimal taqqoslashlar yordamida aniqlanadi. Bu yondashuv algoritmga O(log n) darajadagi vaqt murakkabligini ta’minlaydi — bu esa uni katta hajmdagi ma’lumotlar bilan ishlashda juda foydali qiladi.

Binary Search algoritmining asosiy tarkibiy qismlari:

1. Tartiblangan massiv yoki ro‘yxat – Binary Search faqat oldindan saralangan (ko‘payish yoki kamayish tartibida) ma’lumotlar to‘plamida ishlaydi.

2. Boshlang‘ich va oxirgi indekslar (low va high) – Qidiruv boshlanadigan oraliq chegaralari.

3. O‘rta indeks (mid) – Har bir bosqichda oraliqning markaziy elementi hisoblanadi. Bu element bilan qidirilayotgan qiymat solishtiriladi.

4. Shartli taqqoslash – Mid qiymati izlanayotgan qiymatdan katta yoki kichik ekanligiga qarab, qidiruv oraliqning yuqori yoki pastki yarmiga o‘tkaziladi.

Binary Search algoritmining ikki asosiy implementatsiyasi mavjud:

Iterativ usul – *While* yoki *for* sikllari yordamida oraliq doirasini yangilab borish orqali amalga oshiriladi.

Rekursiv usul – Har bir qadamda funksiyani *o‘z-o‘ziga chaqirish* orqali qidiruv davom ettiriladi. Bu yondashuv kodni yanada *ixcham* va tushunarli qiladi, lekin chuqur rekursiya resurs sarfini oshirishi mumkin.

Binary Search quyidagi bosqichlarda ishlaydi:

1. Qidiruv boshlanishida low = 0, high = n-1 qilib olinadi.
2. mid = (low + high) / 2 hisoblanadi.
3. Agar a[mid] == x, unda qidiruv yakunlanadi – element topilgan bo‘ladi.
4. Agar a[mid] > x, unda qidiruv yuqori yarmini tashlab, high = mid - 1 qilib belgilanadi.
5. Agar a[mid] < x, unda pastki yarmi tashlab, low = mid + 1 bo‘ladi.
6. Qidiruv oraliq bo‘shaguncha davom etadi (low > high bo‘lmaguncha).

Binary Search algoritmining afzalliklari:

Juda katta hajmdagi ma’lumotlar to‘plamida tez ishlaydi.

Hisobiy murakkabligi O(log n) bo‘lgani uchun juda samarali.

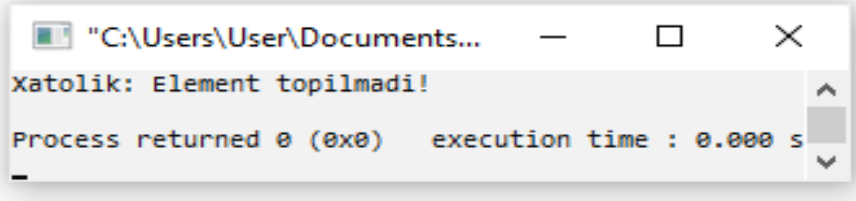
Massivning tartiblanganligi sababli, qidiruv qadamlar soni keskin kamayadi.

Biroq bu algoritm faqat saralangan ma’lumotlar bilan ishlashini talab qiladi. Agar massiv tartiblanmagan bo‘lsa, avval uni saralash kerak bo‘ladi, bu esa umumiy hisobda qo‘shimcha vaqt talab etadi.

Binary Search nafaqat nazariy jihatdan, balki amaliy dasturlashda ham muhim rol o‘ynaydi. Masalan, qidiruv tizimlari, elektron kataloglar, ma’lumotlar bazalari, algoritmik o‘yinlar, foydalanuvchi interfeyslarida ko‘p hollarda ushbu algoritmdan foydalaniladi. Shu sababli uni chuqur o‘rganish — dasturchi uchun zarur bo‘lgan fundamental ko‘nikmalardan biridir.

Misol : Binary Search va noto‘g‘ri indeks xatosi

```cpp
#include <iostream>
#include <stdexcept>
using namespace std;

int binarySearch(int arr[], int size, int target) {
    int left = 0, right = size - 1;

    while (left <= right) {
        int mid = (left + right) / 2;

        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    // Element topilmasa, xatolik chiqariladi
    throw runtime_error("Element topilmadi!");
}

int main() {
    int sonlar[] = {2, 4, 6, 8, 10};
    int n = 5;

    try {
        int indeks = binarySearch(sonlar, n, 7); // 7 massivda yo'q
        cout << "Element indeksi: " << indeks << endl;
    } catch (const exception &e) {
        cout << "Xatolik: " << e.what() << endl;
    }

    return 0;
}
```



Izoh: Ushbu dasturda binarySearch funksiyasi ikkilik qidiruv algoritmi yordamida massivda berilgan sonni izlaydi. Agar qidirilayotgan son (bu yerda 7) massivda topilmasa, throw operatori orqali "Element topilmadi!" degan xatolik xabari tashlanadi.

a-holat. try bloki orqali funksiya chaqiriladi.

b-holat. Agar element topilsa, uning indeksi chiqariladi.

c-holat. Agar topilmasa, catch bloki orqali foydalanuvchiga aniq va tushunarli xabar beriladi: Xatolik: Element topilmadi!.

Bu misol orqali Binary Search algoritmida xatolik yuzaga kelganda dasturni to‘xtatmasdan, foydalanuvchiga xatolik haqida ma’lumot berish usuli ko‘rsatilgan.

C++ DASTURLASH TILIDA BINARY SEARCH VA ENG KO‘P UCHRAYDIGAN XATOLIKLAR

Binary Search — tartiblangan massivda ma’lum bir qiymatni tez va samarali izlash uchun keng qo‘llaniladigan algoritmdir. Ushbu algoritm yuqori samaradorlikka ega bo‘lsa-da, noto‘g‘ri indekslar, yaroqsiz ma’lumotlar yoki kutilmagan holatlar tufayli turli xatoliklar yuzaga kelishi mumkin. Shuning uchun, bunday holatlarni to‘g‘ri boshqarish uchun exception handling (favqulodda holatlarni boshqarish) mexanizmlaridan foydalanish muhimdir.

C++ dasturida Binary Search jarayonida yuzaga kelishi mumkin bo‘lgan asosiy xatoliklar:

1. Nolga bo‘lish xatosi

Garchi Binary Search algoritmida bevosita nolga bo‘lish kam uchrasa-da, boshqa hisoblashlar bilan birgalikda bunday holat yuzaga kelishi mumkin. Masalan, indeksni noto‘g‘ri hisoblash holatlari.

2. Massiv chegarasidan chiqish xatosi

Binary Search noto‘g‘ri tashkil etilganda left, right yoki mid indekslari massiv chegarasidan chiqib ketishi mumkin, bu esa dasturning to‘xtab qolishiga olib keladi.

3. Ko‘rsatkich (pointer) xatolari

Dynamic memory (dinamik xotira) asosida tashkil etilgan massivlar bilan ishlaganda noto‘g‘ri yoki bo‘sh pointerlarga murojaat qilish xavfi mavjud.

4. Yaroqsiz ma’lumot kiritish xatolari

Binary Search ishlashi uchun massiv oldindan tartiblangan bo‘lishi kerak. Foydalanuvchi noto‘g‘ri yoki tartiblanmagan ma’lumot kiritsa, natijalar noto‘g‘ri bo‘ladi.

5. Xotira to‘lib qolish (memory overflow)

Juda katta massivlar yoki rekursiv qidiruv algoritmlarida stek to‘lib qolishi mumkin.

Exception handling roli:

Favqulodda holatlar bilan ishlash orqali dastur: xatoliklarni oldindan aniqlay oladi, to‘xtab qolmasdan davom etadi, foydalanuvchiga aniq va tushunarli xabarlar beradi, dastur ishonchliligini oshiradi.

Shuning uchun, Binary Search kabi algoritmlarni yozishda try-catch bloklari orqali xatoliklarni ushlash va ularni foydalanuvchiga tushunarli tarzda yetkazish tavsiya etiladi.

Xulosa

Zamonaviy dasturlashda algoritmlarning tezligi, ishonchliligi va barqaror ishlashi muhim ahamiyat kasb etadi. Ayniqsa, katta hajmdagi ma’lumotlar bilan ishlovchi dasturlarda samarali qidiruv usullaridan foydalanish — bu tizim unumdorligini sezilarli darajada oshiradi. Shunday algoritmlardan biri bu — Binary Search bo‘lib, u tartiblangan massivlar ichida ma’lum bir elementni logarifmik vaqtda topish imkonini beradi. Bu esa uni chiziqli qidiruvdan (Linear Search) ancha samarali qiladi. Biroq, hatto mukammal algoritm ham noto‘g‘ri qo‘llanilsa yoki kutilmagan holatlar hisobga olinmasa, xatoliklar yuzaga kelishi mumkin. C++ dasturlash tilida bu kabi xatolarni aniqlash, ularga munosib javob qaytarish va dastur barqarorligini ta’minlash uchun exception handling mexanizmlaridan keng foydalaniladi. Ayniqsa, Binary Search algoritmida massiv chegarasidan chiqish, noto‘g‘ri indekslar bilan ishlash, yaroqsiz ma’lumotlar kiritilishi kabi xatoliklar throw-catch orqali boshqarilishi mumkin. Shu jihatdan qaraganda, algoritmik yechimlar bilan bir qatorda xatoliklarni boshqarish tizimini to‘g‘ri tashkil qilish dasturiy ta’minotning sifat ko‘rsatkichlaridan biri hisoblanadi. Bu nafaqat foydalanuvchiga qulay interfeys yaratadi, balki tizimni ishonchli va xavfsiz ishlashiga ham kafolat bo‘ladi.

Xulosa qilib aytganda, Binary Search algoritmini ishlab chiqishda algoritmning mantiqiy to‘g‘riligi, massiv bilan ishlashda ehtiyotkorlik, va favqulodda holatlar bilan ishlashni to‘g‘ri yo‘lga qo‘yish zaruriy hisoblanadi. Bularning barchasi yagona maqsadga — barqaror, tezkor va foydalanuvchi uchun qulay dastur yaratishga xizmat qiladi.

Foydalanilgan adabiyotlar

1. Stroustrup, B. The C++ Programming Language. 4th Edition. Addison-Wesley, 2013. (C++ dasturlash tili asoschilari tomonidan yozilgan, chuqur nazariy va amaliy asoslarga ega manba)

2. Sedgewick, R., & Wayne, K. Algorithms. 4th Edition. Addison-Wesley, 2011. (Algoritmlar, xususan, Binary Search kabi qidiruv usullarining nazariy va amaliy yoritilishi)

3. Lafore, R. Object-Oriented Programming in C++. 4th Edition. Sams Publishing, 2002. (C++ tilidagi obyektga yo‘naltirilgan dasturlash, exception handling va misollar bilan)

4. Skiena, S. S. The Algorithm Design Manual. 2nd Edition. Springer, 2008. (Qidiruv, tartiblash va boshqa algoritmik yondashuvlarning amaliy tahlili)

5. Tanenbaum, A. S. Structured Computer Organization. 6th Edition. Pearson, 2012. (Dastur tuzilmalari va kompyuter tizimlari haqida chuqur ma’lumotlar)

6. C++ Reference. https://en.cppreference.com (C++ standart kutubxona va exception handling mexanizmlari bo‘yicha rasmiy qo‘llanma)

7. GeeksforGeeks. Binary Search and Exception Handling tutorials. https://www.geeksforgeeks.org (Algoritmlar va C++ tilidagi exception mexanizmlari bo‘yicha misollar va tushuntirishlar)