Algoritmai

Dvejetainė paieška

Dvejetainės paieškos algoritmas

Principas:

- Pradinė sąlyga: Masyvas turi būti surikiuotas didėjančia arba mažėjančia tvarka.
- 2. Pirmas žingsnis: Randame vidurinį masyvo elementą.
- 3. Patikrinimas:
- - Jei vidurinis elementas yra tas, kurio ieškome, paieška baigta.
- - Jei ieškomas elementas yra mažesnis už vidurinį, toliau ieškome tik kairinėje pusėje.
- - Jei ieškomas elementas yra didesnis už vidurinį, toliau ieškome tik dešinėje pusėje.
- 4. **Pakartojimas:** Tęsiame procesą su atitinkama puse tol, kol randame elementą arba paaiškėja, kad elemento nėra (jei sumažėja paieškos diapazonas iki nulio).

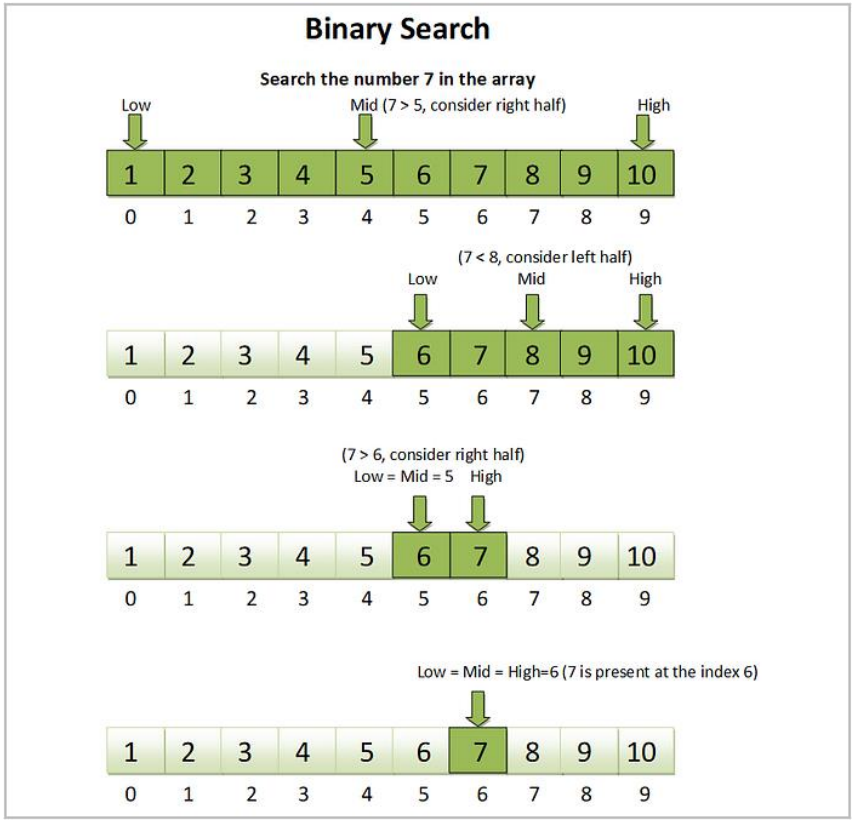
Pseudokodas

```
funkcija DvejetainėPaieška(masyvas, ieškomasElementas):
    pradžia = 0
    pabaiga = masyvoDydis - 1

    kol pradžia ≤ pabaiga:
        vidurys = (pradžia + pabaiga) // 2

        jei masyvas[vidurys] == ieškomasElementas:
            grąžinti vidurys
        jei masyvas[vidurys] < ieškomasElementas:
            pradžia = vidurys + 1
        kitaip:
            pabaiga = vidurys - 1

    grąžinti -1  // Elementas nerastas
```



Šaltinis: https://dolly-desir.medium.com/algorithms-binary-search-2656c7eb5049

Laiko sudėtingumas

O(log n),

kur **n** yra masyvo dydis. Kiekviename žingsnyje masyvas dalijamas per pusę, todėl *algoritmas yra efektyvus dideliems masyvams*

Python įgyvendinimas

```python
def binary_search(arr, target):
    left, right = 0, len(arr) - 1

    while left <= right:
        mid = (left + right) // 2  # Randame vidurį

        if arr[mid] == target:
            return mid  # Elementas rastas

        elif arr[mid] < target:
            left = mid + 1  # Ieškome dešinėje pusėje

        else:
            right = mid - 1  # Ieškome kairėje pusėje

    return -1  # Elementas nerastas

# Pavyzdys:
arr = [1, 3, 5, 7, 9, 11, 13]
target = 7

result = binary_search(arr, target)
if result != -1:
    print(f'Elementas rastas indeksu {result}')
else:
    print('Elementas nerastas')
```

C++ įgyvendinimas

```cpp
int binary_search(const std::vector<int>& arr, int target) {
    int left = 0, right = arr.size() - 1;

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

        if (arr[mid] == target)
            return mid;  // Elementas rastas

        else if (arr[mid] < target)
            left = mid + 1;  // Ieškome dešinėje pusėje

        else
            right = mid - 1;  // Ieškome kairėje pusėje
    }

    return -1;  // Elementas nerastas
}

int main() {
    std::vector<int> arr = {1, 3, 5, 7, 9, 11, 13};
    int target = 7;

    int result = binary_search(arr, target);
    if (result != -1) {
        std::cout << "Elementas rastas indeksu " << result << std::endl;
    } else {
        std::cout << "Elementas nerastas" << std::endl;
    }
}
```

Paaiškinimas

Python:

- binary_search funkcija priima masyvą arr ir ieškomą elementą target.
- Naudojamas ciklas, kuris kartoja, kol left (kairė riba) yra mažesnė arba lygi right (dešinė riba).
- Kiekvienoje iteracijoje randamas vidurinis elementas mid ir lyginamas su ieškomu elementu.
- Jei randamas ieškomas elementas, funkcija grąžina jo indeksą. Jei nerandama, grąžinama -1.

C++:

- Veikimas yra toks pat kaip Python versijos, tik su C++ sintakse.
- Funkcija taip pat grąžina elemento indeksą, jei randama, arba -1, jei nerandama.
- Naudojamas vector<int> kaip pagrindinė duomenų struktūra.

Apibendrinimas

- **Efektyvumas:**

Dvejetainė paieška yra itin efektyvus algoritmas, nes jo sudėtingumas yra $O(\log n)$O(logn), o tai reiškia, kad paieška atliekama labai greitai, ypač dideliuose masyvuose.

- **Veikimas tik su surikiuotais masyvais:**

Algoritmas veikia tik tada, kai duomenys yra iš anksto surikiuoti, todėl prieš pradedant paiešką būtina pasirūpinti, kad masyvas būtų tinkamai sutvarkytas.

- **Svarbiausi aspektai:**

Greitas paieškos laikas: Dėl mažėjančios paieškos erdvės per kiekvieną iteraciją paieška atliekama labai efektyviai, net ir su dideliais duomenų rinkiniais.

Plačiai naudojamas: Dvejetainės paieškos algoritmas yra svarbus įrankis programinės įrangos inžinerijoje, dažnai naudojamas duomenų struktūrose, paieškos sistemose ir daugelyje kitų sričių.