UNIVERSITAS SULTAN AGENG TIRTAYASA

UNTIRTA Jawara
Jujur Adil Wibawa Amanah Religius Akuntabel

Algoritma dan Pemrograman

#11 Meeting

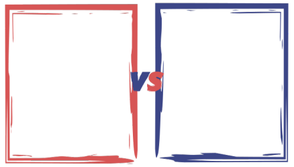
Metode Searching

Ferdian Bangkit Wijaya, S.Stat., M.Si
NIP. 19900520202406100

Pengertian

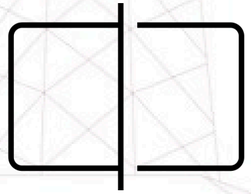
Kumpulan langkah sistematis yang dirancang untuk menemukan keberadaan atau lokasi suatu data spesifik (disebut target atau key) di dalam sekumpulan data.

Unsorted Search



Algoritma yang mencari data tanpa memerlukan data dalam kondisi terurut. Algoritma ini memeriksa data satu per satu.

Sorted Search



Algoritma yang memanfaatkan struktur data yang sudah terurut untuk menemukan data dengan jauh lebih efisien, seringkali dengan membuang sebagian besar data di setiap langkah.

Unsorted Search

1.Sequential Search

Mencari elemen dengan memeriksa setiap data secara berurutan, satu per satu, dari awal hingga akhir.

2. Hash Table Search

Menggunakan fungsi hash untuk menghitung indeks (posisi) data secara langsung, memungkinkan pencarian instan tanpa perbandingan.

3. Binary Search Tree (BST) Search

Mencari data dengan menelusuri struktur pohon, bergerak ke cabang kiri (nilai lebih kecil) atau kanan (nilai lebih besar) secara rekursif.

Sequential Search

Contoh 1: Target Ditemukan (Data Unik)
Vektor Awal: [7, 5, 1, 9, 3] (n = 5)

Target Dicari: 1

Langkah-langkah Iterasi:
i = 1: Cek vektor[1]. Apakah 7 == 1? Tidak. Lanjut.

i = 2: Cek vektor[2]. Apakah 5 == 1? Tidak. Lanjut.

i = 3: Cek vektor[3]. Apakah 1 == 1? Ya.

DITEMUKAN di indeks 3. Proses pencarian berhenti.

Contoh 2: Target Ditemukan (Data Duplikat)

- Vektor Awal: [7, 5, 9, 3, 5, 2] (n = 6)
- Target Dicari: 5

Langkah-langkah Iterasi:

1. i = 1: Cek vektor[1]. Apakah 7 == 5? Tidak. Lanjut.
2. i = 2: Cek vektor[2]. Apakah 5 == 5? Ya.
3. DITEMUKAN di indeks 2. Proses pencarian berhenti.

Hash Table Search

Contoh 1: Pencarian Ideal (Tanpa Tabrakan)
Vektor Awal: [7, 5, 1, 9, 3]

Fungsi Hash (Contoh): Gunakan fungsi sederhana indeks = (nilai % 5) + 1. (Modulo 5, lalu +1 agar 1-based).

Proses Membangun Tabel Hash (Setup):

7: (7 % 5) + 1 = 2 + 1 = 3. Simpan 7 di bucket [3].

5: (5 % 5) + 1 = 0 + 1 = 1. Simpan 5 di bucket [1].

1: (1 % 5) + 1 = 1 + 1 = 2. Simpan 1 di bucket [2].

9: (9 % 5) + 1 = 4 + 1 = 5. Simpan 9 di bucket [5].

3: (3 % 5) + 1 = 3 + 1 = 4. Simpan 3 di bucket [4].

Tabel Hash Final: [(1): [5], (2): [1], (3): [7], (4): [3], (5): [9]]

Target Dicari: 9

Iterasi Pencarian:
Hitung hash dari Target 9: indeks = (9 % 5) + 1 = 5.

Lompat langsung ke bucket indeks 5.

Periksa isi bucket [5]. Ditemukan 9.

DITEMUKAN. (Pencarian selesai dalam 1 langkah).

Hash Table Search

Contoh 2: Pencarian dengan Tabrakan (Collision)
Vektor Awal: [7, 5, 1, 9, 3, **14**]

Fungsi Hash: indeks = (nilai % 5) + 1

Proses Membangun Tabel Hash:

... (Sama seperti di atas)

14: (14 % 5) + 1 = 4 + 1 = 5.

Bucket [5] sudah berisi 9. Kita tambahkan 14 ke bucket yang sama (disebut chaining).

Tabel Hash Final: [(1): [5], (2): [1], (3): [7], (4): [3], (5): [9, 14]]

Target Dicari: 14

Iterasi Pencarian:
Hitung hash dari Target 14: indeks = (14 % 5) + 1 = 5.

Lompat langsung ke bucket indeks 5.

Isi bucket [5] adalah [9, 14]. Kita lakukan Sequential Search di dalam bucket ini.

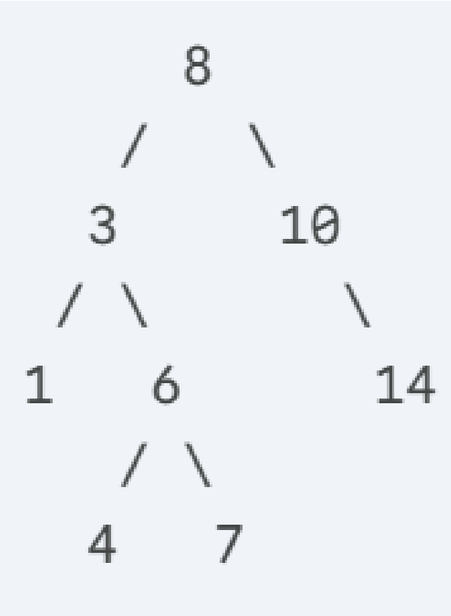
Cek elemen 1 di bucket: Apakah 9 == 14? Tidak.

Cek elemen 2 di bucket: Apakah 14 == 14? Ya.

DITEMUKAN.

Binary Search Tree (BST) Search

Vektor Input: [8, 10, 3, 14, 6, 1, 7, 4]



- Target Dicari: 7

Iterasi Pencarian (Menelusuri Jalur):

1. Mulai dari Akar (Root). Nilai 8.
2. Bandingkan: Target (7) < 8? Ya. Pindah KIRI ke simpul 3.
3. Bandingkan: Target (7) > 3? Ya. Pindah KANAN ke simpul 6.
4. Bandingkan: Target (7) > 6? Ya. Pindah KANAN ke simpul 7.
5. Bandingkan: Target (7) == 7? Ya.
6. DITEMUKAN.

Sorted Search

1.Binary Search

Mencari elemen pada data terurut dengan membandingkan target ke nilai tengah, lalu membuang setengah bagian pencarian secara berulang.

2. Interpolation Search

Menebak posisi elemen secara cerdas (interpolasi) pada data terurut, dengan asumsi data terdistribusi seragam (bukan sekadar membelah di tengah).

3. Jump Search

Mencari pada data terurut dengan "melompat" maju dalam interval tetap (blok), lalu melakukan pencarian berurutan di dalam blok yang relevan.

Binary Search

Contoh 1:

- Vektor Awal (HARUS TERURUT):
- [1, 3, 5, 7, 9, 11, 13] (n = 7)
- Target Dicari: 3

Gunakan tiga penunjuk (pointer): kiri, kanan, dan tengah.

Iterasi 1:

- kiri = 1, kanan = 7
- tengah = floor((1 + 7) / 2) = 4
- Cek vektor[tengah] → vektor[4] adalah 7.
- Perbandingan: Target (3) < (7). Target pasti ada di kiri.
- Buang bagian kanan. Atur ulang batas: kanan = tengah - 1 (yaitu 4 - 1 = 3).
- Ruang Pencarian Baru: [1, 3, 5] (Indeks 1 s/d 3)

Iterasi 2:

- kiri = 1, kanan = 3
- tengah = floor((1 + 3) / 2) = 2
- Cek vektor[tengah] → vektor[2] adalah 3.
- Perbandingan: Target (3) == (3).
- DITEMUKAN di indeks 2. Proses pencarian berhenti.

Interpolation Search

Contoh 1: Pencarian Ideal (Data Seragam)

- Vektor Awal: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100] (n = 10)
- Target Dicari: 90

Iterasi 1:

- L=1 (nilai 10), R=10 (nilai 100).
- Formula (1-based):
  pos = L + floor(((Target - V[L]) * (R - L)) / (V[R] - V[L]))
- pos = 1 + floor(((90 - 10) * (10 - 1)) / (100 - 10))
- pos = 1 + floor((80 * 9) / 90)
- pos = 1 + floor(720 / 90)
- pos = 1 + 8 = 9.
- Cek vektor[pos] → vektor[9] adalah 90.
- DITEMUKAN di indeks 9. (Hanya 1 iterasi).

Interpolation Search

Pencarian Kasus Buruk (Data Tidak Seragam)

- Vektor Awal: [1, 2, 3, 100, 101, 102] (n = 6)
- Target Dicari: 3

Iterasi 1:

- L=1 (nilai 1), R=6 (nilai 102).
- pos = 1 + floor(((3 - 1) * (6 - 1)) / (102 - 1))
- pos = 1 + floor((2 * 5) / 101)
- pos = 1 + floor(10 / 101)
- pos = 1 + 0 = 1.
- Cek vektor[1] (nilai 1). Target (3) > 1. Ruang pencarian baru: L = pos + 1 = 2.

Iterasi 2:

- L=2 (nilai 2), R=6 (nilai 102).
- pos = 2 + floor(((3 - 2) * (6 - 2)) / (102 - 2))
- pos = 2 + floor((1 * 4) / 100)
- pos = 2 + 0 = 2.
- Cek vektor[2] (nilai 2). Target (3) > 2. Ruang pencarian baru: L = pos + 1 = 3.

Iterasi 3:

- L=3 (nilai 3), R=6 (nilai 102).
- pos = 3 + floor(((3 - 3) * (6 - 3)) / (102 - 3))
- pos = 3 + 0 = 3.
- Cek vektor[3] (nilai 3).
- DITEMUKAN di indeks 3. (Membutuhkan 3 langkah, O(n), karena data tidak seragam).

Jump Search

Contoh 1: Target Ditemukan

- Vektor Awal: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31] (n = 16)
- Target Dicari: 27
- Setup: Ukuran lompatan m = sqrt{16} = 4.

Iterasi 1 (Lompatan):

- Cek indeks 1 * 4 = 4. vektor[4] adalah 7.
- Target (27) > 7. Lanjut lompat.

Iterasi 2 (Lompatan):

- Cek indeks 2 * 4 = 8. vektor[8] adalah 15.
- Target (27) > 15. Lanjut lompat.

Iterasi 3 (Lompatan):

- Cek indeks 3 * 4 = 12. vektor[12] adalah 23.
- Target (27) > 23. Lanjut lompat.

Iterasi 4 (Lompatan):

- Cek indeks 4 * 4 = 16. vektor[16] adalah 31.
- Target (27) < 31. STOP.
- Target, jika ada, pasti berada di antara indeks 12 (lompatan sebelumnya) dan indeks 16.

Iterasi 5 (Sequential Search):

- Lakukan Sequential Search di blok terakhir (mulai dari indeks 12 + 1 = 13):
- Cek vektor[13]: 25 == 27? Tidak.
- Cek vektor[14]: 27 == 27? Ya.
- DITEMUKAN di indeks 14.

UNIVERSITAS SULTAN AGENG TIRTAYASA

UNTIRTA Jawara
Jujur Adil Wibawa Amanah Religius Akuntabel

SEE YOU NEXT WEEK !

Ferdian Bangkit Wijaya, S.Stat., M.Si
NIP. 19900520202406100 1
ferdian.bangkit@untirta.ac.id