Perbandingan Algoritma Pencarian Kunci di dalam Himpunan Terurut Melalui *Linear Search* dan *Binary Search*

Biolardi Yoshogi (13509035)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
vsio@students.itb.ac.id

***Abstract*—Ada banyak teknik yang dapat digunakan ketika mencari data (kunci) di dalam suatu himpunan data. Di makalah ini dipersempit sehingga yang dibahas adalah data yang terurut saja dan teknik yang digunakan ada dua. Pencarian kunci di suatu data terurut, dapat digunakan teknik pencarian linear atau biner. Akan tetapi, manakah dari kedua teknik tersebut yang lebih baik. Untuk menemukannya, digunakan kompleksitas waktu dari kompleksitas algoritma berupa nilai N dengan satuan kali.**

***Index Terms*— biner, linear, pencarian, terurut**

I. PENDAHULUAN

Dalam mencari suatu data terurut di dalam himpunan data, bisa dilakukan pencarian. Untuk memudahkan pencarian tersebut, bisa digunakan suatu algoritma sehingga pencarian dilakukan secara otomatis. Dalam praktik di dunia nyata, pencarian di dalam data yang berurut, digunakan misalnya untuk mencari nama di suatu daftar berdasarkan nomor indeks, daftar absen, dan masih banyak lagi. Untuk melakukan pencarian tersebut ada beberapa cara. Akan tetapi, di dalam makalah ini hanya di bahas dengan dua cara yaitu *Linear Search* (Pencarian Linear) dan *Binary Search* (Pencarian Biner). Kembali kepada teknik tadi, hal tersebut memang efektif untuk mencari suatu data di dalam data terurut. Akan tetapi, dari kedua algoritma tersebut, manakah yang lebih mangkus untuk kasus pencarian data terurut? Apakah *Linear Search* atau *Binary Search*?

II. TEORI DASAR

Linear Search

Linear Search adalah metode untuk menemukan nilai tertentu pada sebuah larik dengan cara memeriksa elemennya satu per satu secara berurutan hingga nilai (kunci) yang dicari ditemukan. Jika ditemukan, pencarian akan dihentikan.[1]

Kompleksitas waktunya adalah O(n)

Secara ringkasnya:

1. Pertama atur ke posisi 0
2. Uji apakah posisi tersebut memiliki nilai yang dicari
3. Jika iya, mengembalikan nilai
4. Jika tidak, atur ke posisi selanjutnya yaitu 1
5. Ulangi 2 hingga 4 sehingga posisinya = n-1.
6. Jika posisinya = n-1 tetapi nilai belum ditemukan maka nilai tidak ditemukan atau bisa mengembalikan pesan nilai tidak ditemukan

Contoh:
Di dalam sebuah larik, ada data: 2 3 5 7 11 13 17.
Kunci = 5

1. Pertama atur ke posisi 0 (memiliki nilai = 2)
2. Uji apakah posisi tersebut memiliki nilai yang dicari (5 != 2)
3. Jika iya, mengembalikan nilai (Tidak sama)
4. Jika tidak, atur ke posisi selanjutnya yaitu 1 (memiliki nilai = 3)

5. Uji apakah posisi tersebut memiliki nilai yang dicari (5 != 3)
6. Jika iya, mengembalikan nilai (Tidak sama)
7. Jika tidak, atur ke posisi selanjutnya yaitu 1 (memiliki nilai = 5)

8. Uji apakah posisi tersebut memiliki nilai yang dicari (5 != 5)
9. Jika iya, mengembalikan nilai (sama: Posisi = 3)

Hasil: Posisi = 3

Binary Search

Binary Search adalah metode untuk menemukan suatu posisi suatu nilai di dalam array yang berurutan. Di setiap langkah, algoritma ini membandingkan kunci masukan dengan nilai tengah suatu array. Jika ditemukan, maka akan mengembalikan posisi nilai input

masukan. Jika tidak, secara rekursif akan terus mencari dengan membandingkan kunci masukan dengan sub-larik yang mana sub-larik tersebut adalah setengah dari ukuran asal larik yang mana psosinya dimulai dari posisi median. [2]

Secara ringkasnya:

1. Pertama dicari nilai median
2. Kemudian dibandingkan apakah merupakan nilai yang dicari
3. Jika iya, akan mengembalikan posisi nilai tersebut
4. Jika bukan, potong setengah array tersebut dimulai dari median menjadi sub-larik
   - Jika kunci lebih kecil, maka dari posisi 0 hingga posisi median dikurangi satu
   - Jika kunci lebih besar, maka dari posisi median ditambah satu hingga posisi ukuran array dikurangi satu.
5. Ulangi langkah 1 dengan masukan larik adalah sub-larik yang telah didapat tadi

Kompleksitas waktunya adalah O(log n).

Contoh:

Data masukan: 2 3 5 7 11 13 17
Nilai yang ingin dicari => 5
Langkah:

1. Pertama dicari nilai median yaitu 7 di posisi 4
2. Kemudian dibandingkan apakah merupakan nilai yang dicari (5 < 7)
3. Jika iya, akan mengembalikan posisi nilai tersebut (belum sama)
4. Jika bukan, potong setengah array tersebut dimulai dari median
   - Jika kunci lebih kecil, maka dari posisi 0 hingga posisi median dikurangi satu
   - Jika kunci lebih besar, maka dari posisi median ditambah satu hingga posisi ukuran array dikurangi satu.

   Dalam kasus ini adalah lebih kecil sehingga yang diambil adalah {2,3,5}

5. Dicari nilai median yaitu 3
6. Kemudian dibandingkan apakah merupakan nilai yang dicari (5 > 3)
7. Jika iya, akan mengembalikan posisi nilai tersebut (belum sama)
8. Jika bukan, potong setengah array tersebut dimulai dari median
   - Jika kunci lebih kecil, maka dari posisi 0 hingga posisi median dikurangi satu
   - Jika kunci lebih besar, maka dari posisi median ditambah satu hingga posisi ukuran array dikurangi satu.

   Dalam kasus ini adalah lebih kecil sehingga yang diambil adalah {5}

9. Dicari nilai median yaitu 5

   Kemudian dibandingkan apakah merupakan nilai yang dicari (5 = 5)
10. Jika iya, akan mengembalikan posisi nilai tersebut (sama sehingga yang dikembalikan adalah posisi nilai 5 tadi yaitu 3.

Hasil yang didapat adalah 3 yang menunjukkan posisi 3..

Kompleksitas Waktu

Kuantitas waktu yang digunakan oleh algoritma untuk menjalankan suatu fungsi berdasarkan input.

Untuk menentukannya digunakan Notasi Big-Oh dan N kali.

III. METODE PEMECAHAN MASALAH

Untuk menemukan perbandingan apakah *linear search* atau *binary search* lebih baik, akan dilakukan beberapa cara dengan melihat beberapa faktor:

1. Ukuran input data
2. Nilai n dalam satuan kali

Ukuran input data yang dimaksud adalah banyaknya data di dalam sebuah larik yang akan dimasukkan.

Contohnya:

1. {1,2,3,4,5}, ukuran 5
2. {111,22,33,55,6,4,3,22,3,4,4,4,5,5}, ukuran 14
3. {}, ukuran 0

Nilai n dalam satuan kali adalah jumlah eksekusi yang dilakukan pada sebuah operasi. Biasanya bagian yang dilirik:

1. Operasi penjumlahan
2. Operasi pengurangan
3. Operasi perbandingan

Kedua faktor tersebut akan menentukan apakah kedua algoritma tersebut lebih mangkus atau tidak.

Asumsi yang digunakan:

1. Data sudah terurut
2. Bahasa yang digunakan = Action Script 3
3. Perhitungan nilai n dilakukan pada operasi dasar (perbandingan dalam kasus ini)
4. Fungsi pencarian median diasumsikan = 1 kali
5. Algoritma yang digunakan, bersifat rekursif

IV. EKSPERIMEN DAN HASILNYA

Input: Larik, Kunci
Output: N kali

Eksperimen 1

- Input
  Larik = {2, 4, 6, 9, 11,15,17}
  Jumlah Data = 7
  Kunci = 2

- Output
  N linear = 2
  N biner = 6

Eksperimen 2

- Input
  Larik = {}
  Jumlah Data = 0
  Kunci = 2

- Output
  N linear = 1
  N biner = 1

Eksperimen 3

- Input
  Larik = {0, 3, 6, 9, 12}
  Jumlah Data = 5
  Kunci = 12

- Output
  N linear = 10
  N biner = 6

Eksperimen 4

- Input
  Larik = {0, 5, 10}
  Jumlah Data = 3
  Kunci = 12 (Tidak ditemukan)

- Output
  N linear = 7
  N biner = 5

Eksperimen 5

- Input
  Larik = {, 0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99, 102, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 135, 138, 141, 144, 147}
  Jumlah Data = 50
  Kunci = 3

- Output
  N linear = 4
  N biner = 12

Eksperimen 6

- Input
  Larik = {, 0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99, 102, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 135, 138, 141, 144, 147}

  Jumlah Data = 50
  Kunci = 3

- Output
  N linear = 101
  N biner = 13

Eksperimen 7

- Input
  Larik = 0, 25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 475, 500, 525, 550, 575, 600, 625, 650, 675, 700, 725, 750, 775, 800, 825, 850, 875, 900, 925, 950, 975, 1000, 1025, 1050, 1075, 1100, 1125, 1150, 1175, 1200, 1225, 1250, 1275, 1300, 1325, 1350, 1375, 1400, 1425, 1450, 1475, 1500, 1525, 1550, 1575, 1600, 1625, 1650, 1675, 1700, 1725, 1750, 1775, 1800, 1825, 1850, 1875, 1900, 1925, 1950, 1975, 2000, 2025, 2050, 2075, 2100, 2125, 2150, 2175, 2200, 2225, 2250, 2275, 2300, 2325, 2350, 2375, 2400, 2425, 2450, 2475}

  Jumlah Data = 100
  Kunci = 1025

- Output
  N linear = 84
  N biner = 41

Eksperimen 8

- Input
  Larik = {, 0, 25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450,

475, 500, 525, 550, 575, 600, 625, 650, 675, 700, 725, 750, 775, 800, 825, 850, 875, 900, 925, 950, 975, 1000, 1025, 1050, 1075, 1100, 1125, 1150, 1175, 1200, 1225, 1250, 1275, 1300, 1325, 1350, 1375, 1400, 1425, 1450, 1475, 1500, 1525, 1550, 1575, 1600, 1625, 1650, 1675, 1700, 1725, 1750, 1775, 1800, 1825, 1850, 1875, 1900, 1925, 1950, 1975, 2000, 2025, 2050, 2075, 2100, 2125, 2150, 2175, 2200, 2225, 2250, 2275, 2300, 2325, 2350, 2375, 2400, 2425, 2450, 2475}

Jumlah Data = 100
Kunci = 1111 (Tidak ditemukan)

- Output
  N linear = 201
  N biner = 15

Eksperimen 9

Larik = {0, 75, 150, 225, 300, 375, 450, 525, 600, 675, 750, 825, 900, 975, 1050, 1125, 1200, 1275, 1350, 1425, 1500, 1575, 1650, 1725, 1800, 1875, 1950, 2025, 2100, 2175, 2250, 2325, 2400, 2475, 2550, 2625, 2700, 2775, 2850, 2925, 3000, 3075, 3150, 3225, 3300, 3375, 3450, 3525, 3600, 3675, 3750, 3825, 3900, 3975, 4050, 4125, 4200, 4275, 4350, 4425, 4500, 4575, 4650, 4725, 4800, 4875, 4950, 5025, 5100, 5175, 5250, 5325, 5400, 5475, 5550, 5625, 5700, 5775, 5850, 5925, 6000, 6075, 6150, 6225, 6300, 6375, 6450, 6525, 6600, 6675, 6750, 6825, 6900, 6975, 7050, 7125, 7200, 7275, 7350, 7425, 7500, 7575, 7650, 7725, 7800, 7875, 7950, 8025, 8100, 8175, 8250, 8325, 8400, 8475, 8550, 8625, 8700, 8775, 8850, 8925, 9000, 9075, 9150, 9225, 9300, 9375, 9450, 9525, 9600, 9675, 9750, 9825, 9900, 9975, 10050, 10125, 10200, 10275, 10350, 10425, 10500, 10575, 10650, 10725, 10800, 10875, 10950, 11025, 11100, 11175, 11250, 11325, 11400, 11475, 11550, 11625, 11700, 11775, 11850, 11925, 12000, 12075, 12150, 12225, 12300, 12375, 12450, 12525, 12600, 12675, 12750, 12825, 12900, 12975, 13050, 13125, 13200, 13275, 13350, 13425, 13500, 13575, 13650, 13725, 13800, 13875, 13950, 14025, 14100, 14175, 14250, 14325, 14400, 14475, 14550, 14625, 14700, 14775, 14850, 14925}

Jumlah Data = 200
Kunci = 75

- Output
  N linear = 401
  N biner = 17

Eksperimen 10

- Input

Larik = {0, 25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 475, 500, 525, 550, 575, 600, 625, 650, 675, 700, 725, 750, 775, 800, 825, 850, 875, 900, 925, 950, 975, 1000, 1025, 1050, 1075, 1100, 1125, 1150, 1175, 1200, 1225, 1250, 1275, 1300, 1325, 1350, 1375, 1400, 1425, 1450, 1475, 1500, 1525, 1550, 1575, 1600, 1625, 1650, 1675, 1700, 1725, 1750, 1775, 1800, 1825, 1850, 1875, 1900, 1925, 1950, 1975, 2000, 2025, 2050, 2075, 2100, 2125, 2150, 2175, 2200, 2225, 2250, 2275, 2300, 2325, 2350, 2375, 2400, 2425, 2450, 2475, 2500, 2525, 2550, 2575, 2600, 2625, 2650, 2675, 2700, 2725, 2750, 2775, 2800, 2825, 2850, 2875, 2900, 2925, 2950, 2975, 3000, 3025, 3050, 3075, 3100, 3125, 3150, 3175, 3200, 3225, 3250, 3275, 3300, 3325, 3350, 3375, 3400, 3425, 3450, 3475, 3500, 3525, 3550, 3575, 3600, 3625, 3650, 3675, 3700, 3725, 3750, 3775, 3800, 3825, 3850, 3875, 3900, 3925, 3950, 3975, 4000, 4025, 4050, 4075, 4100, 4125, 4150, 4175, 4200, 4225, 4250, 4275, 4300, 4325, 4350, 4375, 4400, 4425, 4450, 4475, 4500, 4525, 4550, 4575, 4600, 4625, 4650, 4675, 4700, 4725, 4750, 4775, 4800, 4825, 4850, 4875, 4900, 4925, 4950, 4975, 5000, 5025, 5050, 5075, 5100, 5125, 5150, 5175, 5200, 5225, 5250, 5275, 5300, 5325, 5350, 5375, 5400, 5425, 5450, 5475, 5500, 5525, 5550, 5575, 5600, 5625, 5650, 5675, 5700, 5725, 5750, 5775, 5800, 5825, 5850, 5875, 5900, 5925, 5950, 5975, 6000, 6025, 6050, 6075, 6100, 6125, 6150, 6175, 6200, 6225, 6250, 6275, 6300, 6325, 6350, 6375, 6400, 6425, 6450, 6475, 6500, 6525, 6550, 6575, 6600, 6625, 6650, 6675, 6700, 6725, 6750, 6775, 6800, 6825, 6850, 6875, 6900, 6925, 6950, 6975, 7000, 7025, 7050, 7075, 7100, 7125, 7150, 7175, 7200, 7225, 7250, 7275, 7300, 7325, 7350, 7375, 7400, 7425, 7450, 7475}

Jumlah Data = 300
Kunci = 1111

- Output
  N linear = 601
  N biner = 19

V. ANALISIS

Dari hasil eksperimen di dapat bahwa pada pencarian linear, dihasilkan nilai n yang relatif kecil jika jumlah data masukannya juga relatif kecil. Akan tetapi, ketika jumlah data masukannya relatif besar, nilai N-nya pun relatif menjadi lebih besar.

Pada pencarian biner, nilai N yang dihasilkan relatif besar jika jumlah input datanya relatif kecil. Akan tetapi,

nilai N-nya relatif jauh lebih kecil ketika jumlah input datanya besar.

Dengan melihat nilai N pada kedua metoda tersebut, terlihat bahwa N linear jauh lebih besar daripada N biner saat jumlah masukannya besar. Bisa dibandingkan, perbedaan tersebut mencapai lebih dari 100%..

V. KESIMPULAN

Dari hasil analisis, dapat disimpulkan bahwa pencarian biner jauh lebih mangkus daripada pencarian linear terutama jika kasus terburuk yang mana akan terjadi pemborosan yang relatif lebih banyak jika menggunakan pencarian linear. Hal ini terlihat dari selisih jumlah N yang begitu tajam ketika kunci diletakkan pada posisi yang begitu relatif sulit dicari atau keadaan terburuk (yaitu kunci ada di akhir larik atau tidak ditemukan sama sekali). Jadi, sebaiknya pencarian biner digunakan pada data yang relatif jauh lebih

Akan tetapi, hasil ini masih tidak memperhitungkan faktor lain seperti apakah datanya terurut atau tidak. Kemudian bagaimana nilai N pada fungsi-fungsi eksternal yang telah digunakan dan bagaimana bahasa pemroses operasi-operasi dasarnya. Akan tetapi, dari hasil tadi didapat bahwa pencarian biner lebih mangkus daripada pencarian linear untuk kasus pencarian kunci di suatu himpunan data terutama himpunan data yang jumlahnya relatif begitu banyak.

REFERENCES

[1] Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L. (1990). Introduction to Algorithms (1st ed.). MIT Press and McGraw-Hill. ISBN 0-262-03141-8.

[2] Donald Knuth (1997). The Art of Computer Programming. 3: Sorting and Searching (3rd ed.). Addison-Wesley. pp. 396–408. ISBN 0-201-89685-0.

[3] Sipser, Michael (2006). Introduction to the Theory of Computation. Course Technology Inc. ISBN 0-619-21764-2.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Desember 2012