**LAPORAN**



**ANALISIS SEARCHING**

Nama : Diaz Adha Asri Prakoso

NIM : 0102518007

Program Studi : Teknik Informatika

**UNIVERSITAS AL AZHAR INDONESIA**

**JAKARTA**

**2019**

**DAFTAR ISI**

[**BAB 1** 1](#_Toc6949193)

[**Sequensial Search dan Binary Search** 1](#_Toc6949194)

[1.1 Teori Searching dan Kompleksitas Algoritma Searching 1](#_Toc6949195)

[1.1.1 Sequential Search 1](#_Toc6949196)

[1.1.2 Kompleksitas algoritma Sequential Search 1](#_Toc6949197)

[1.1.3 Binary Search 2](#_Toc6949198)

[1.1.4 Kompleksitas algoritma Binary Search 2](#_Toc6949199)

[1.2 Deskripsi Tugas 2](#_Toc6949200)

[**BAB 2** 3](#_Toc6949201)

[**Source Program** 3](#_Toc6949202)

[**BAB 3** 17](#_Toc6949203)

[**Analisis Hasil Eksekusi Program** 17](#_Toc6949204)

[**Daftar Pustaka** 21](#_Toc6949205)

# **BAB 1**

# **Sequensial Search dan Binary Search**

* 1. **Teori Searching dan Kompleksitas Algoritma Searching**
* **Searching**

Pencarian(searhing) merupakan proses yang fundamental dalam pengolahan data. Proses pencarian adalah menemukan nilai(data) tertentu di dalam sekumpulan data yang bertipe sama.

Sebuah algoritma pencarian dijelaskan secara luas adalah sebuah algoritma yang menerima masukan berupa sebuah masalah dan menghasilkan sebuah solusi untuk masalah tersebut.

Algoritma pencarian (searching algorithm) adalah algoritma yang menerima sebuah argumen kunci dan dengan  langkah-langkah tertentu akan mencari rekaman dengan kunci tersebut.  Setelah proses pencarian dilaksanakan, akan diperoleh salah satu dari dua kemungkinan, yaitu data yang dicari ditemukan (successful) atau tidak ditemukan (unsuccessful).

* **Kompleksitas Algoritma Searching**

Kompleksitas dari suatu algoritma merupakan ukuran seberapa banyak pengulangan yang dibutuhkan algoritma tersebut untuk menyelesaikan masalah. Algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki kompleksitas yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan masalah, mempunyai kompleksitas yang tinggi.

* + 1. **Sequential Search**

Pencarian Sekuensial (sequential searching) atau pencarian berurutan sering disebut pencarian linear merupakan metode pencarian yang paling sederhana.

Pencarian beruntun adalah proses membandingkan setiap elemen larik satu per satu secara beruntun, mulai dari elemen pertama sampai elemen yang dicari ditemukan atau seluruh elemen sudah diperiksa.

* + 1. **Kompleksitas algoritma Sequential Search**

Tergantung dari jumlah perbandingan yang terjadi dalam perulangan saat melakukan pencarian data dalam sebuah list array. Pencarian nilai dalam algoritma ini membutuhkan waktu yang lama sehingga mempunyai kompleksitas yang tinggi.

* + 1. **Binary Search**

Metode ini digunakan untuk kebutuhan pencarian dengan waktu yang cepat. Prinsip pencarian dengan membagi data atas dua bagian / mencari nilai tengah. Data yang disimpan di dalam array harus sudah terurut.

* + 1. **Kompleksitas algoritma Binary Search**

Tergantung dari banyaknya data, dan data yang dimiliki adalah data yang sudah terurut, kasus terburuknya adalah jika data yang dicari tidak terdapat di array, maka algoritma ini bisa jadi memiliki kompleksitas yang tinggi.

* 1. **Deskripsi Tugas**
* Jumlah elemen array : n1 = 10, n2 = 100, n3 = 10000
* Isi array dengan fungsi randomize
* Eksekusi algoritma seq acak, seq acak sentinel, seq urut, binary search dengan :
* 10 x nilai ketemu
* 5 x nilai tidak ketemu (nilai yang dicari tidak ada di array)
* Simpulkan dan bandingkan kompleksitas algoritma searching yang telah dijalankan

# **BAB 2**

# **Source Program**

Source program sekuensial acak

|  |  |
| --- | --- |
| 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  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84 | #include <stdio.h>  #include <stdlib.h>  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Program Sekuensial\_acak  Deskripsi : Mencari sebuah nilai pada array secara sekuensial  IS : Array telah berisi nilai  FS : (1) Nilai tersebut ditemukan, outputnya = indeks elemen yang ditemukan  (2) Nilai, tersebut tidak ditemukan, pencarian telah sampai pada akhir elemen, outputnya = -1  Dibuat oleh : Diaz Adha Asri Prakoso  Tanggal : 21 April 2019  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //header procedure  void Searching\_Sekuensial\_Acak(int n, int N, int acak[10], int \*indeks);  void main (void)  //kamus  {  int acak[10001];  int cari, i, indeks, n;    //ALgoritma program utama \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  printf ("Program Pencarian Sekuensial \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    //Menampilkan isi array dulu  printf ("Isi array adalah ..... \n");  printf ("Masukkan jumlah array : ");  scanf ("%d", &n);  for (i=1; i<=n; i++)  {  acak[i] = rand() % n;  printf ("A[%d] = %d \n",i, acak[i]);  }  printf ("\n Masukkan Nilai yang dicari = ");  scanf ("%d", &cari);    Searching\_Sekuensial\_Acak(n, cari, acak, &indeks);      if (indeks != -1)  {  printf ("Nilai %d ditemukan pada indeks yang ke- %d", cari, indeks);  }  else //indeks == -1  {  printf ("Nilai %d tidak ditemukan", cari);  }    printf ("\n\n");  system ("PAUSE");  }  void Searching\_Sekuensial\_Acak(int n, int N, int acak[10],int \*indeks)  //Mencari sebuah nilai pada array secara sekuensial  // IS : Array telah berisi nilai  // FS : (1) Nilai tersebut ditemukan, outputnya = indeks elemen yang ditemukan  // (2) Nilai, tersebut tidak ditemukan, pencarian telah sampai pada akhir elemen, outputnya = -1  //KAMUS  { int i;  i = 1;  while (i<=n && acak[i] != N)  {  i = i + 1;  }  //i>n || acak[i] = N    if (i>n)  {  \*indeks = -1; //Nilai cari tidak ditemukan  }  else //i<=n  {  \*indeks = i; //Nilai cari ditemukan  }  } |

Source program sekuensial acak pakai sentinel

|  |  |
| --- | --- |
| 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  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89 | #include <stdio.h>  #include <stdlib.h>  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Program Sekuensial\_acak\_sentinel  Deskripsi : Mencari sebuah nilai pada array secara sekuensial menggunakan sentinel  IS : Array telah berisi nilai  FS : (1) Nilai tersebut ditemukan, outputnya = indeks elemen yang ditemukan  (2) Nilai, tersebut tidak ditemukan, pencarian telah sampai pada akhir elemen / ketemu sentinel, outputnya = -1  Dibuat oleh : Diaz Adha Asri Prakoso  Tanggal : 21 April 2019  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //header procedure  void Searching\_Sekuensial\_Acak(int n, int N, int acak[10], int \*indeks);  void main (void)  //kamus  {  int acak[10005];  int cari, i, indeks, n;    //ALgoritma program utama \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  printf ("Program Pencarian Sekuensial \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    //Menampilkan isi array dulu  printf ("Isi array adalah ..... \n");  printf ("Masukkan jumlah array : ");  scanf ("%d", &n);    for (i=1; i<=n; i++)  {  acak[i] = rand() % n;  printf ("A[%d] = %d \n",i, acak[i]);  }  acak[n+1] = -9999;  printf ("A[%d] = %d ", i, acak[n+1]);  printf ("\n Masukkan Nilai yang dicari = ");  scanf ("%d", &cari);    Searching\_Sekuensial\_Acak(n, cari, acak, &indeks);      if (indeks != -1)  {  printf ("Nilai %d ditemukan pada indeks yang ke- %d", cari, indeks);  }  else //indeks == -1  {  printf ("Nilai %d tidak ditemukan", cari);  }    printf ("\n\n");  system ("PAUSE");  }  void Searching\_Sekuensial\_Acak(int n, int N, int acak[10],int \*indeks)  //Mencari sebuah nilai pada array secara sekuensial  // IS : Array telah berisi nilai  // FS : (1) Nilai tersebut ditemukan, outputnya = indeks elemen yang ditemukan  // (2) Nilai, tersebut tidak ditemukan, pencarian telah sampai pada akhir elemen, outputnya = -1  //KAMUS  {  int i;  i = 1;  while (acak[i]!=-9999 && acak[i]!= N)  {  i = i + 1;  }  //acak[i]=-9999 || acak[i] = N    if (acak[i]==-9999)  {  \*indeks = -1; //Nilai cari tidak ditemukan  }  else //acak[i] != -9999  {  \*indeks = i; //Nilai cari ditemukan  }  } |

Source program sekuensial urut

|  |  |
| --- | --- |
| 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  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106 | #include <stdio.h>  #include <stdlib.h>  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Program Sekuensial\_acak\_urut  Deskripsi : Mencari sebuah nilai pada array secara sekuensial urut  IS : Array telah berisi nilai  FS : (1) Nilai tersebut ditemukan, outputnya = indeks elemen yang ditemukan  (2) Nilai, tersebut tidak ditemukan, pncarian telah sampai pada akhir elemen, outputnya = -1  Dibuat oleh : Diaz Adha Asri Prakoso  Tanggal : 21 April 2019  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //header procedure  void Searching\_Sekuensial\_Urut(int n, int N, int acak[10], int \*indeks);  void main (void)  //kamus  {  int acak[10001];  int cari, i, indeks, n, a, b, temp;    //ALgoritma program utama \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  printf ("Program Pencarian Sekuensial \n");  printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    //Menampilkan isi array dulu  printf ("Isi array adalah ..... \n");    printf ("Masukkan jumlah array : ");  scanf ("%d", &n);    for (i=1; i<=n; i++)  {  acak[i] = rand() % n;  printf ("A[%d] = %d \n",i, acak[i]);  }    for (a=1; a<=n; a++)  {  for (b=a+1; b <=n; b++)  {  if(acak[a]>acak[b])  {  temp = acak[b];  acak[b] = acak[a];  acak[a] = temp;  }  }  }  printf ("\nData setelah diurut : \n");  for (a=1; a<=n; a++)  {  printf ("A[%d] = %d\n", a, acak[a]);  }    printf ("\n Masukkan Nilai yang dicari = ");  scanf ("%d", &cari);    Searching\_Sekuensial\_Urut(n, cari, acak, &indeks);    if (indeks != -1)  {  printf ("Nilai %d ditemukan pada indeks yang ke- %d", cari, indeks);  }  else //indeks == -1  {  printf ("Nilai %d tidak ditemukan", cari);  }    printf ("\n\n");  system ("PAUSE");  }  void Searching\_Sekuensial\_Urut(int n, int N, int acak[10],int \*indeks)  //Mencari sebuah nilai pada array secara sekuensial  // IS : Array telah berisi nilai  // FS : (1) Nilai tersebut ditemukan, outputnya = indeks elemen yang ditemukan  // (2) Nilai, tersebut tidak ditemukan, pencarian telah sampai pada akhir elemen, outputnya = -1  //KAMUS  {  int i;  i = 1;  while (i<=n && acak[i]< N)  {  i = i + 1;  }  //i>n || A[i]>=N    if (acak[i]!=N)  {  \*indeks = -1; //Nilai cari tidak ditemukan  }  else //acak[i] == N  {  \*indeks = i; //Nilai cari ditemukan  }  } |

Source program binary search

|  |  |
| --- | --- |
| 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  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102 | #include <stdio.h>  #include <stdlib.h>  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Program Binary\_search  Deskripsi : Mencari sebuah nilai pada array secara binary search  IS : Array telah berisi nilai  FS : (1) Nilai tersebut ditemukan, outputnya = indeks elemen yang ditemukan  (2) Nilai, tersebut tidak ditemukan, pencarian telah sampai pada akhir elemen, outputnya = -1  Dibuat oleh : Diaz Adha Asri Prakoso  Tanggal : 21 April 2019  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //procedure untuk mencari angka dengan pencarian bagi dua  void biner ( int cari ,int n , int angka[], int \*c, int \*Temu)  {  //kamus  int akhir ,awal ,tengah, i, ketemu;    akhir = n;  awal = 0;  ketemu = -1;  i = 1;  //Perulangan untuk mencari angka nya dengan kondisi ketemu = -1 dan i <= n  while ( ketemu == -1 && i <=n )  {  tengah = ( awal + akhir ) / 2;    if ( angka[tengah] == cari )  {  ketemu = tengah;  }  else //angka[tengah] != cari  {  if ( angka[tengah] < cari )  {  awal = tengah + 1;  }  else //angka[tengah] >= cari  {  akhir = tengah - 1;  }  }  i++;  } //ketemu!=1 || i>n  \*c = i;  \*Temu = ketemu;  }  int main ()  { //kamus  int cari , i , n, a, b, temp, c, Temu;  int angka[10001];  printf("Berapa data yg ingin di tampilkan secara acak : ");  scanf("%d" , &n);  printf("\nInput angka secara terurut menaik\n");  for (i = 1; i <=n; i++)  {  angka[i] = rand() % n;  printf("A[%d] = %d\n ", i, angka[i]);  }    for (a=1; a<=n; a++)  {  for (b=a+1; b <=n; b++)  {  if(angka[a]>angka[b])  {  temp = angka[b];  angka[b] = angka[a];  angka[a] = temp;  }  }  }  printf ("\nData setelah diurut : \n");  for (a=1; a<=n; a++)  {  printf ("A[%d] = %d\n", a, angka[a]);  }    printf("Berapa angka yg ingin dicari: ");  scanf("%d", &cari);  biner ( cari , n , angka, &c, &Temu);  if ( Temu == - 1 )  {  printf("\nData tidak ditemukan !! \n");  }  else //Temu != -1  {  printf("\nAngka %d ditemukan di indeks %d\n", cari, Temu);  }  printf ("Nilai i = %d", c);  return 0;  } |

# **BAB 3**

# **Analisis Hasil Eksekusi Program**

Jumlah pengulangan pembandingan untuk n1 = 10 (Panjang Array)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Elemen yang dicari** | **Seq acak** | **Seq acak sentinel** | **Seq urut** | **Binary Search** |
| 1 | 1 | 1 | 1 | 2 | 3 |
| 2 | 7 | 2 | 2 | 7 | 5 |
| 3 | 4 | 3 | 3 | 4 | 2 |
| 4 | 0 | 4 | 4 | 1 | 4 |
| 5 | 9 | 5 | 5 | 10 | 5 |
| 6 | 4 | 6 | 3 | 4 | 2 |
| 7 | 8 | 7 | 7 | 8 | 3 |
| 8 | 8 | 7 | 7 | 8 | 3 |
| 9 | 2 | 9 | 9 | 3 | 4 |
| 10 | 4 | 3 | 3 | 4 | 2 |
|  |  |  |  |  |  |
| 1 | 20 | 11 | 11 | 11 | 11 |
| 2 | 40 | 11 | 11 | 11 | 11 |
| 3 | 50 | 11 | 11 | 11 | 11 |
| 4 | 25 | 11 | 11 | 11 | 11 |
| 5 | 100 | 11 | 11 | 11 | 11 |
|  | **Rata – rata** | 6.8 | 6.8 | 7.06 | 5.8 |

Jumlah pengulangan pembandingan untuk n2 = 100 (Panjang Array)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Elemen yang dicari** | **Seq acak** | **Seq acak sentinel** | **Seq urut** | **Binary Search** |
| 1 | 41 | 1 | 1 | 45 | 6 |
| 2 | 67 | 2 | 2 | 75 | 3 |
| 3 | 0 | 4 | 4 | 1 | 7 |
| 4 | 34 | 3 | 3 | 34 | 8 |
| 5 | 58 | 8 | 8 | 66 | 7 |
| 6 | 64 | 10 | 10 | 71 | 6 |
| 7 | 45 | 12 | 12 | 54 | 7 |
| 8 | 91 | 16 | 16 | 93 | 5 |
| 9 | 95 | 17 | 17 | 98 | 7 |
| 10 | 4 | 22 | 22 | 5 | 5 |
|  |  |  |  |  |  |
| 1 | 120 | 101 | 101 | 101 | 101 |
| 2 | 115 | 101 | 101 | 101 | 101 |
| 3 | 300 | 101 | 101 | 101 | 101 |
| 4 | 213 | 101 | 101 | 101 | 101 |
| 5 | 500 | 101 | 101 | 101 | 101 |
|  | **Rata - rata** | 40 | 40 | 69.8 | 37.73 |

Jumlah pengulangan pembandingan untuk n3 = 10000 (Panjang Array)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Elemen yang dicari** | **Seq acak** | **Seq acak sentinel** | **Seq urut** | **Binary Search** |
| 1 | 485 | 218 | 218 | 631 | 11 |
| 2 | 778 | 60 | 60 | 1006 | 13 |
| 3 | 923 | 7102 | 7102 | 1191 | 13 |
| 4 | 774 | 1215 | 1215 | 1003 | 15 |
| 5 | 951 | 2753 | 2753 | 1223 | 12 |
| 6 | 1270 | 4875 | 4875 | 1610 | 14 |
| 7 | 2813 | 199 | 199 | 3475 | 9 |
| 8 | 9497 | 659 | 659 | 9547 | 13 |
| 9 | 2192 | 7756 | 7756 | 2744 | 13 |
| 10 | 4087 | 4439 | 4439 | 4610 | 13 |
|  |  |  |  |  |  |
| 1 | 13000 | 10001 | 10001 | 10001 | 10001 |
| 2 | 12500 | 10001 | 10001 | 10001 | 10001 |
| 3 | 11000 | 10001 | 10001 | 10001 | 10001 |
| 4 | 30000 | 10001 | 10001 | 10001 | 10001 |
| 5 | 25000 | 10001 | 10001 | 10001 | 10001 |
|  | **Rata - rata** | 5285.4 | 5285.4 | 5136.3 | 3342.06 |

Rangkuman jumlah pembandingan untuk tiap algoritma

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **n** | **Seq acak** | **Seq acak sentinel** | **Seq urut** | **Binary Search** |
| **10** | 6.8 | 6.8 | 7.06 | 5.8 |
| **100** | 40 | 40 | 69.8 | 37.73 |
| **10000** | 5285.4 | 5285.4 | 5136.3 | 3342.06 |

Pertambahan jumlah pembandingan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Peningkatan n** | **Seq acak** | **Seq acak sentinel** | **Seq urut** | **Binary Search** |
| 10 kali | 33.2 | 33.2 | 62.74 | 31.93 |
| 100 kali | 5245.4 | 5245.4 | 5093.5 | 3304.33 |
| Kesimpulan Kompleksitas Algoritma | Memiliki kompleksitas yang tinggi, karena untuk mendapatkan nilai yang dicari di butuhkan pengulangan yang lumayan lama. | Memiliki kompleksitas yang tinggi, karena untuk mendapatkan nilai yang dicari di butuhkan pengulangan yang lumayan lama. | Memiliki kompleksitas yang sangat tinggi, karena untuk mendapatkan nilai yang dicari dibutuhkan pengulangan yang lebih lama di bandingkan dengan sekuensial acak dan sekuensial acak sentinel. | Memiliki kompleksitas yang rendah, karena untuk mendapatkan nilai yang dicari dibutuhkan pengulangan yang sedikit di bandingkan dengan sekuensial.. |

# **Daftar Pustaka**

Sinta,Dara. 2010. “*ALGORITMA SEARCHING”*

<http://darasinta.blogspot.com/2010/12/algoritma-searching.html> [Diakses tanggal 23 April 2019]

Gautama, Elliana. 2017. “Kompleksitas Algoritma”

<https://dosen.perbanas.id/kompleksitas-algoritma/> [Diakses tanggal 23 April 2019]