JOBSHEET 11

Searching: Linear Search & Binary Search

Fakultas	Proram Studi	Kode MK	Waktu
Teknik	Teknik Informatika	INF1.62.2014	2 x 50 Menit

TUJUAN PRAKTIKUM

1. Mahasiswa memahami konsep pencarian dengan metode linear search dan binary search.
2. Mahasiswa mampu mengimplementasikan konsep pencarian dalam pemrograman C menggunakan IDE.

HARDWARE & SOFTWARE

1. Personal Computer
2. DevC++ IDE

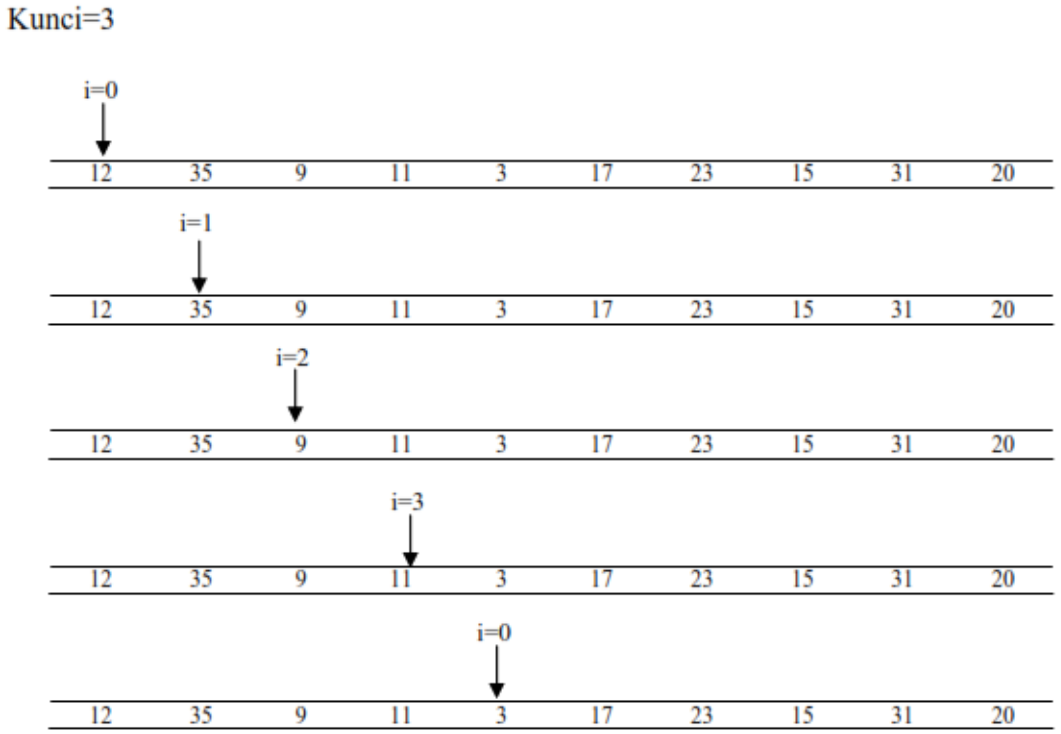
TEORI SINGKAT

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*).

Ada dua macam teknik pencarian yaitu pencarian sekuensial (*linear search*) dan pencarian biner (*binary search*). Perbedaan dari dua teknik ini terletak pada keadaan data. Pencarian sekuensial digunakan apabila data dalam keadaan acak atau tidak terurut. Sebaliknya, pencarian biner digunakan pada data yang sudah dalam keadaan urut.

A. Linear Search

Algoritma pencarian dapat dijelaskan sebagai berikut: pencarian dimulai dari data paling awal, kemudian ditelusuri dengan menaikkan indeks data, apabila data sama dengan kunci pencarian dihentikan dan diberikan nilai pengembalian *true*, apabila sampai indeks terakhir data tidak ditemukan maka diberikan nilai pengembalian *false*.

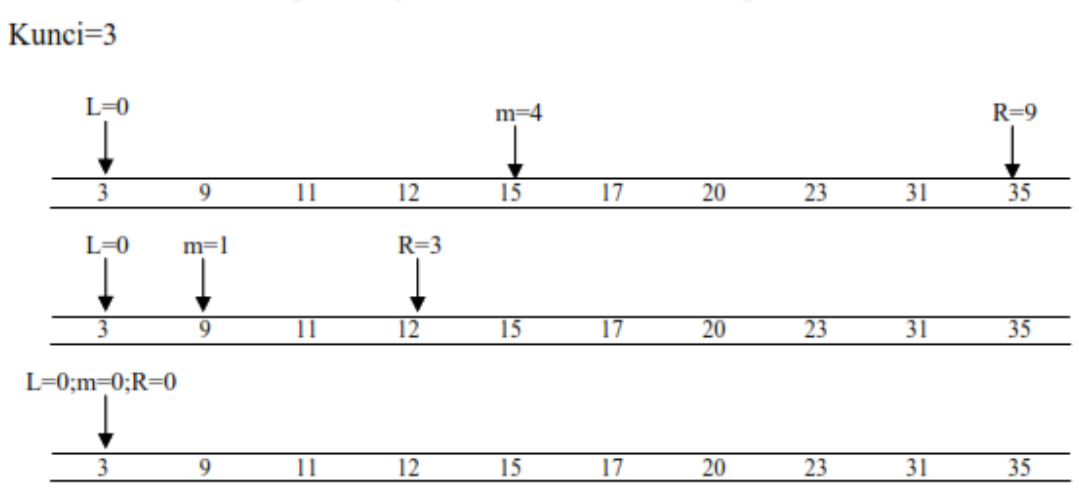


Algoritma pencarian berurutan dapat dituliskan sebagai berikut:

1. i ← 0
2. ketemu ← false
3. Selama (tidak ketemu) dan (i <= N) kerjakan baris 4
4. Jika (Data[i] = x) maka ketemu ← true, jika tidak i ← i + 1
5. Jika (ketemu) maka i adalah indeks dari data yang dicari, jika tidak data tidak ditemukan

B. Binary Search

Salah satu syarat agar pencarian biner dapat dilakukan adalah data sudah dalam keadaan urut. Dengan kata lain, apabila data belum dalam keadaan urut, pencarian biner tidak dapat dilakukan.



Algoritma binary search:

1. Data diambil dari posisi 1 sampai posisi akhir N
2. Kemudian cari posisi data tengah dengan rumus: (posisi awal + posisi akhir) / 2
3. Kemudian data yang dicari dibandingkan dengan data yang di tengah, apakah sama atau lebih kecil, atau lebih besar.
4. Jika lebih besar, maka proses pencarian dicari dengan posisi awal adalah posisi tengah + 1
5. Jika lebih kecil, maka proses pencarian dicari dengan posisi akhir adalah posisi tengah − 1
6. Jika data sama, berarti ketemu.

PERCOBAAN

1. Linear Search

```
#include <stdio.h>

int main()
{
  int array[100], search, c, n, count = 0;

  printf("Enter number of elements in array\n");
  scanf("%d", &n);
  printf("Enter %d numbers\n", n);

  for (c = 0; c < n; c++)
       scanf("%d", &array[c]);\
       printf("Enter a number to search\n");
       scanf("%d", &search);

  for (c = 0; c < n; c++) {
    if (array[c] == search) {
      printf("%d is present at location %d.\n", search, c+1);
      count++;
    }
  }
```

```
  if (count == 0)
        printf("%d isn't present in the array.\n", search);
  else
       printf("%d is present %d times in the array.\n", search, count);
       return 0;
}
```

2. Linear Search

```
#include <stdio.h>

int main()
{
  int array[100], search, c, n, count = 0;

  printf("Enter number of elements in array\n");
  scanf("%d", &n);

  printf("Enter %d numbers\n", n);

  for (c = 0; c < n; c++)
       scanf("%d", &array[c]);
       printf("Enter a number to search\n");
       scanf("%d", &search);

  for (c = 0; c < n; c++) {
       if (array[c] == search) {
       printf("%d is present at location %d.\n", search, c+1);
       count++;
    }
  }
  if (count == 0)
       printf("%d isn't present in the array.\n", search);
  else
       printf("%d is present %d times in the array.\n", search, count);

       return 0;
}
```

3. Linear Search

```
#include <stdio.h>

int search(int arr[], int n, int x)
{
   int i;
   for (i = 0; i < n; i++)
      if (arr[i] == x)
         return i;
   return -1;
}

int main(void)
{
   int arr[] = { 2, 3, 4, 10, 40 };
```

```
    int x = 10;
    int n = sizeof(arr) / sizeof(arr[0]);
    int result = search(arr, n, x);
    (result == -1)          ? printf("Element is not present in array")
                            : printf("Element is present at index %d", result);
    return 0;
}
```

4. Linear Search

```
#include <stdio.h>
#include <stdlib.h>
main(){
   int tabInt[10]={24,17,18,15,22,26, 13,21, 16, 28};
   int ketemu;
   int cariData;
   int i;
   printf("Masukkan data yang dicari = ");
   scanf("%d",&cariData);
   i=0;
   ketemu=0;
   while (i<10 && ketemu!=1){
      if (tabInt[i]==cariData){
      ketemu=1;
      }else{
         i++;
      }
   }
   if(ketemu==1){
      printf("Data %d terdapat pada kumpulan data\n",cariData );
   }else{
      printf("Data %d tidak terdapat pada kumpulan data\n",cariData );
   }
}
```

5. Binary Search

```
#include <stdio.h>
int main()
{
  int c, first, last, middle, n, search, array[100];

  printf("Enter number of elements\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  printf("Enter value to find\n");
  scanf("%d", &search);

  first = 0;
  last = n - 1;
  middle = (first+last)/2;
```

```
 while (first <= last) {
  if (array[middle] < search)
   first = middle + 1;
  else if (array[middle] == search) {
   printf("%d found at location %d.\n", search, middle+1);
   break;
  }
  else
   last = middle - 1;

  middle = (first + last)/2;
 }
 if (first > last)
  printf("Not found! %d isn't present in the list.\n", search);

 return 0;
}
```

6. Binary Search

```
#include <stdio.h>

int binarySearch(int arr[], int l, int r, int x)
{
   if (r >= l) {
      int mid = l + (r - l) / 2;

      if (arr[mid] == x)
         return mid;
      if (arr[mid] > x)
         return binarySearch(arr, l, mid - 1, x);
      return binarySearch(arr, mid + 1, r, x);
   }
   return -1;
}

int main(void)
{
   int arr[] = { 2, 3, 4, 10, 40 };
   int n = sizeof(arr) / sizeof(arr[0]);
   int x = 10;
   int result = binarySearch(arr, 0, n - 1, x);
   (result == -1)    ? printf("Element is not present in array")
                     : printf("Element is present at index %d",
                  result);
   return 0;
}
```

7. Binary Search

```
    #include <stdio.h>
    #include <stdlib.h>

       int main(){
       int tabInt[10] = {12,23,29,34,56,60,67,78,84,99};
```

```
    int i,j,k;
    int cariData,ketemu;
    printf("Masukkan data yang dicari = ");
    scanf("%d",&cariData);
    i = 0; j = 9;ketemu = 0;
      while((ketemu == 0) && (i<=j)){
        k = (int)(i + j) / 2;
        if(tabInt[k] == cariData){
        ketemu = 1;
      }
      else{
        if(tabInt[k] > cariData){
          j = k - 1;
        }
        else{
          i = k + 1;
        }
      }
    }
    if(ketemu==1){
      printf("Data %d terdapat pada kumpulan data\n",cariData );
    }else{
      printf("Data %d tidak terdapat pada kumpulan data\n",cariData );
    }
}
```

TUGAS

1. Implementasikan data mahasiswa dengan pencarian data menggunakan metode sequential search dan binary search.

   Data mahasiswa mempunyai 4 data yaitu:

   - NIM, bertipe bulat
   - Nama, bertipe string
   - TTL, bertipe string
   - IPK, bertipe float

   Kunci yang digunakan untuk pencarian data adalah berdasarkan NIM!

DAFTAR PUSTAKA

1. Kernighan, Brian W, & Ritchie, Dennis M. 1988. The Ansi C Programming Language Second Edition, Prentice-Hall.
2. Cipta Ramadhani. 2015. Dasar Algoritma & Struktur Data. Yogyakarta: ANDI.