KEMENTRIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
DIREKTORAT JENDERAL PENDIDIKAN TINGGI, RISET, DAN TEKNOLOGI
DIREKTORAT PEMBELAJARAN DAN KEMAHASISWAAN

Kampus Merdeka
INDONESIA JAYA

ALGORITMA DAN PEMROGRAMAN

[Komang Aryasa| [Pertemuan 21 dan 22]

Outline

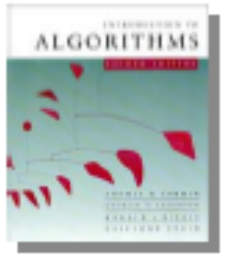
- Pencarian Bagi Dua
- Teknik Shell Sort
- Teknik Bubble Sort
- Teknik penyisipan

Pencarian Bagi Dua

- Pencarian bagi-dua (binary search) adalah teknik yang diterapkan hanya pada elemen larik yang telah terurut (sorted).
- Pencarian beruntun memiliki satu kekurangan yaitu dalam kasus terburuk (elemen yang dicari berada pada posisi terakhir) maka pencarian harus dilakukan sepanjang larik. Semakin banyak elemen maka semakin lama pencarian harus dilakukan.

- Proses pencarian bagi dua dilakukan sebagai berikut:

1. Andaikan jumlah elemen adalah m, maka tetapkan indeks= m/2, sehingga larik terbagi dua, yaitu bagian kiri dengan indeks dari 1 sampai m/2, dan bagian kanan dengan indeks m/2 hingga m.
2. Periksa dulu apakah x = A[indeks], bila ya berarti elemen ditemukan. Bila tidak teruskan ke langkah berikutnya.
3. Periksa apakah x > A[indeks]. Bila ya maka cari disisi kanan. Bila tidak maka cari disisi kiri.
4. Teruskan pencarian pada sisi yang tepat dengan mengambil indeks tengah dari sisi tersebut, sampai elemen ditemukan atau tidak sama sekali.



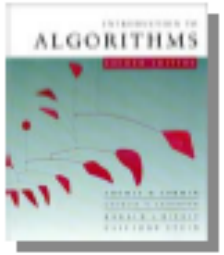
Binary search

Find an element in a sorted array:

1. ***Divide:*** Check middle element.
2. ***Conquer:*** Recursively search 1 subarray.
3. ***Combine:*** Trivial.

Example: Find 9

3	5	7	8	9	12	15

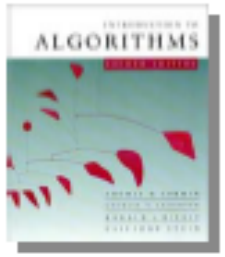


Binary search

Find an element in a sorted array:

1. ***Divide:*** Check middle element.
2. ***Conquer:*** Recursively search 1 subarray.
3. ***Combine:*** Trivial.

Example: Find 9



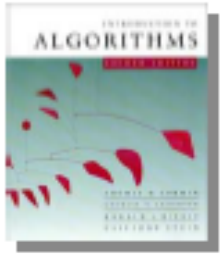
Binary search

Find an element in a sorted array:

1. ***Divide:*** Check middle element.
2. ***Conquer:*** Recursively search 1 subarray.
3. ***Combine:*** Trivial.

Example: Find 9

3 5 7 8 9 12 15



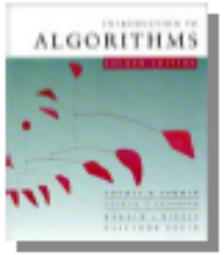
Binary search

Find an element in a sorted array:

1. ***Divide:*** Check middle element.
2. ***Conquer:*** Recursively search 1 subarray.
3. ***Combine:*** Trivial.

Example: Find 9

3 5 7 8 9 12 15



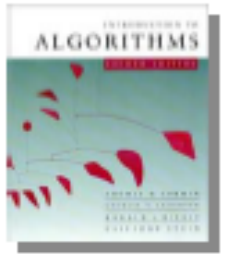
Binary search

Find an element in a sorted array:

1. ***Divide:*** Check middle element.
2. ***Conquer:*** Recursively search 1 subarray.
3. ***Combine:*** Trivial.

Example: Find 9

3 5 7 8 9 12 15



Binary search

Find an element in a sorted array:

1. ***Divide:*** Check middle element.
2. ***Conquer:*** Recursively search 1 subarray.
3. ***Combine:*** Trivial.

Example: Find 9

3 5 7 8 9 12 15

- **Contoh:** andaikan larik A = [7 10 13 16 18 21 76 81], dan x = 10.
- Pada contoh ini jumlah elemen m = 8, sehingga indeks = 8/2 = 4, dan A[4] = 16. Namun x tidak sama dengan A[4].
- Periksa apakah x > A[4], atau x > 16?, jawabannya tidak, periksa sisi kiri.
- Pada sisi kiri indeks = (1 + 4) / 2 = 2, sehingga A[2] = 10.
- Ternyata x=A[2] sehingga elemen ditemukan dalam dua langkah.

Algoritma Binary Search

{ pencarian elemen dengan metoda bagi dua **}**

Deklarasi

integer m=10;

integer A[m], x;

integer idx1, idx2, indeks;

boolean ketemu;

Deskripsi

idx1 ← 1;

idx2 ← m;

ketemu ← **false;**

write (“Masukkan data yang dicari : “**)**;

read(x**);**

write(“Baca larik yang sudah sort”**);**

for idx1=1 to m

read(A[idx1]**);**

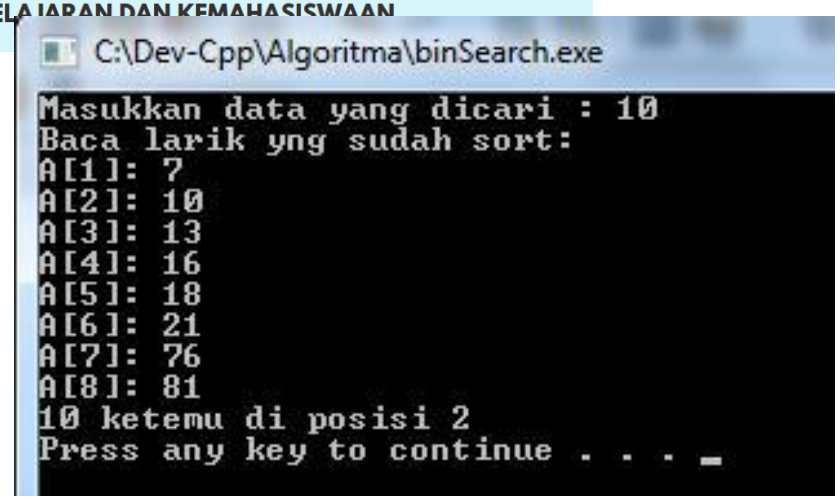
endfor;

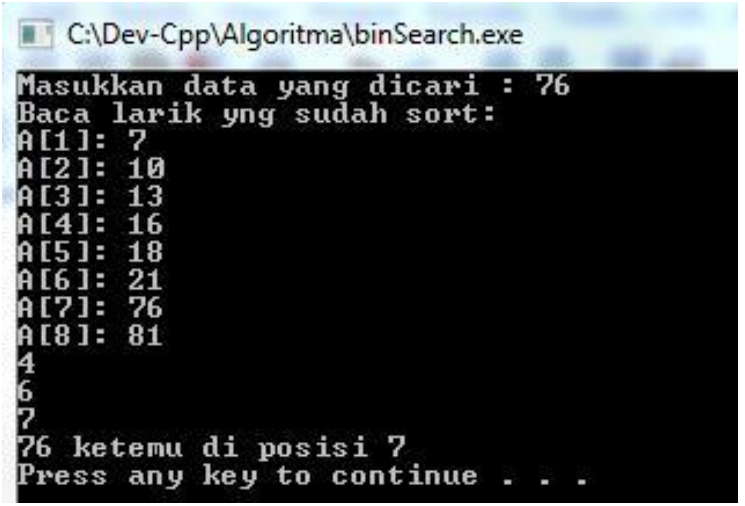
```
while ( !ketemu && ( idx1 <= idx2 ) ) do
    { menghitung titik tengah }
    indeks ← (idx1 + idx2) / 2;
    if ( x = A[indeks] )
        then ketemu ← true;
        else if ( x < A[indeks] )
                then idx2 ← indeks - 1; { sisi kiri }
                else idx1 ← indeks + 1; { sisi kanan }
             endif.
    endif.
endwhile.
if ( ketemu )
    then write ( " x ketemu di posisi : ", indeks );
    else write ( " x tidak ditemukan " );
endif.
```

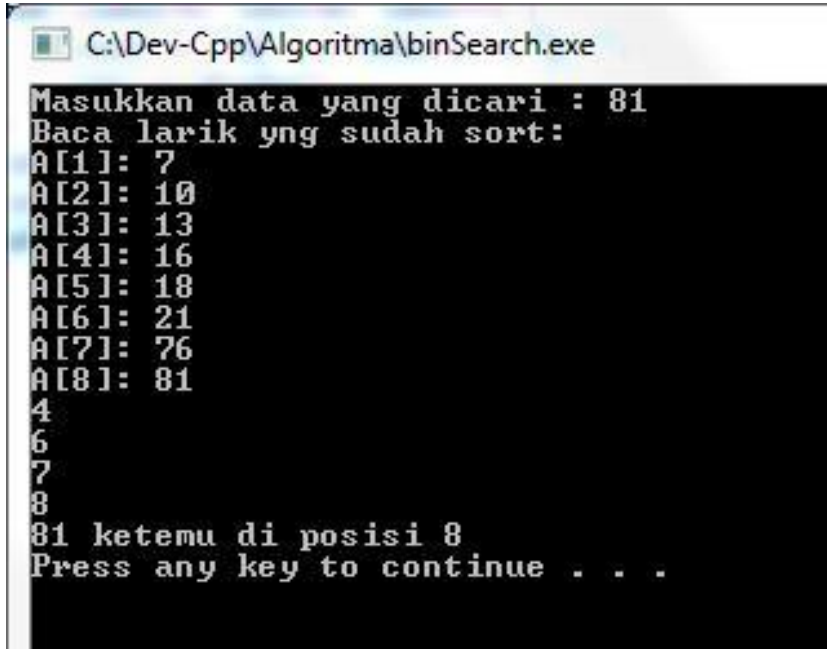
BinSearch.cpp

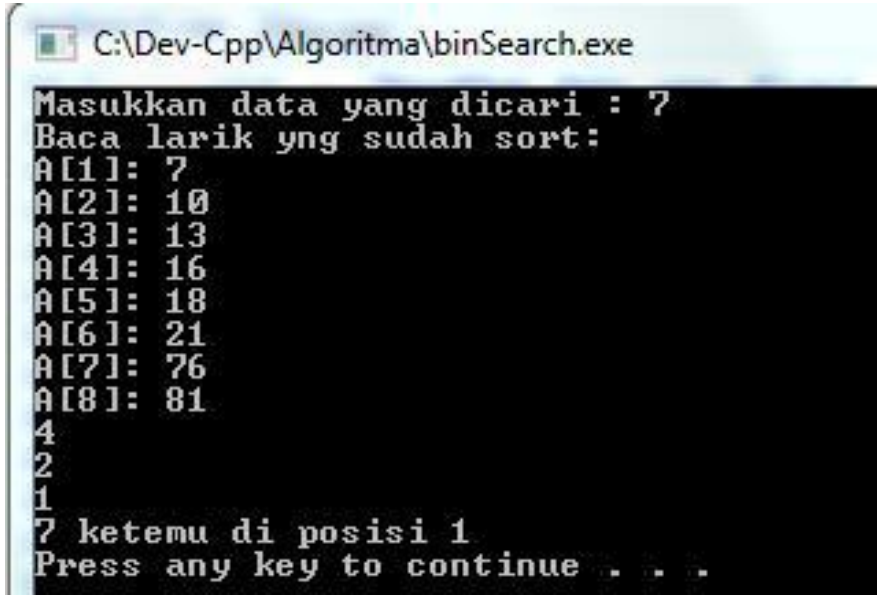
```
//binSearch.cpp
#include <iostream>
#include <math.h>
using namespace std;
const int m=8;
int A[m], x;
int idx1, idx2, indeks;
bool ketemu;

int main() {
        ketemu = false;
        cout << "Masukkan data yang dicari : ";
        cin >> x;
        cout << "Baca larik yng sudah sort:\n";
        for (idx1=0; idx1<m; idx1++) {
                cout << "A[" << idx1+1 <<"]: ";
                cin >> A[idx1];
        }
        idx1=0;
        idx2=m-1;
```

```
                while (!ketemu && (idx1 <= idx2)) {
                        indeks = (idx1 + idx2)/2;
                        cout << indeks+1 << '\n';
                        if (x == A[indeks]) {
                                ketemu = true;
                        }
                        else {
                                if (x < A[indeks])
                                        idx2 = indeks - 1;  // sisi kiri
                                else
                                        idx1 = indeks + 1;  // sisi kanan
                        }
                }
                if (ketemu)
                        cout << x << " ketemu di posisi " << indeks+1 << '\n';
                else
                        cout << x << " tidak ditemukan \n";

                system("PAUSE");
                return 0;
}
```


C:\Dev-Cpp\Algoritma\binSearch.exe
Masukkan data yang dicari : 10
Baca larik yng sudah sort:
A[1]: 7
A[2]: 10
A[3]: 13
A[4]: 16
A[5]: 18
A[6]: 21
A[7]: 76
A[8]: 81
10 ketemu di posisi 2
Press any key to continue . . .


C:\Dev-Cpp\Algoritma\binSearch.exe
Masukkan data yang dicari : 76
Baca larik yng sudah sort:
A[1]: 7
A[2]: 10
A[3]: 13
A[4]: 16
A[5]: 18
A[6]: 21
A[7]: 76
A[8]: 81
4
6
7
76 ketemu di posisi 7
Press any key to continue . . .


C:\Dev-Cpp\Algoritma\binSearch.exe
Masukkan data yang dicari : 81
Baca larik yng sudah sort:
A[1]: 7
A[2]: 10
A[3]: 13
A[4]: 16
A[5]: 18
A[6]: 21
A[7]: 76
A[8]: 81
4
6
7
8
81 ketemu di posisi 8
Press any key to continue . . .


C:\Dev-Cpp\Algoritma\binSearch.exe
Masukkan data yang dicari : 7
Baca larik yng sudah sort:
A[1]: 7
A[2]: 10
A[3]: 13
A[4]: 16
A[5]: 18
A[6]: 21
A[7]: 76
A[8]: 81
4
2
1
7 ketemu di posisi 1
Press any key to continue . . .

Pencarian dengan HASH

- Beberapa aplikasi memerlukan teknik pencarian yang sangat cepat sementara ketiga teknik yang telah disajikan diatas kemungkinan tidak memenuhi keperluan. Berikut ini akan disajikan suatu teknik yang berbasis pada suatu fungsi Hash. Fungsi Hash adalah fungsi matematis yang mengubah (konversi) elemen yang dicari menjadi suatu bilangan, dimana bilangan ini menunjukkan lokasi atau alamat dimana elemen tersebut disimpan.

- Inti dari pencarian berbasis fungsi Hash terletak pada bentuk fungsi Hash yang digunakan.
- Fungsi matematis ini harus dipilih sedemikian rupa sehingga tidak menimbulkan “tabrakan” (collision) yang terlalu banyak.
- Tabrakan adalah kejadian dimana dua elemen berbeda memiliki nilai Hash yang sama. Apabila terjadi tabrakan maka pada sistem pencarian harus ditambahkan tabel collision untuk menampung elemen-elemen yang mengalami tabrakan.

- Langkah yang harus ditempuh untuk meng-implementasi tabel Hash adalah sebagai berikut:
  1. tetapkan ukuran dari tabel Hash, ambil angka prima yang melampaui jumlah data yang akan disimpan
  2. pilih suatu fungsi Hash
  3. tempatkan elemen-elemen kedalam tabel sesuai dengan hasil fungsi Hash

- Ketika pencarian elemen akan dilakukan maka langkahnya sebagai berikut:
  1. hitung nilai fungsi Hash dari elemen
  2. langsung ke lokasi pada tabel sesuai dengan nilai Hash-nya.

- Contoh: andaikan ada 10 data yang akan disimpan, maka dapat dipilih array dengan indeks N=11 atau N=13 (prima > 10).
- Fungsi hash yang digunakan adalah: fungsi mod N, sehingga lokasi data x dalam array adalah: indeks = x mod N
- Contoh: x=45 disimpan pada: 45 mod 11=1,
- 73 disimpan pada : 73 mod 11 = 7,
- 54 disimpan pada : 54 mod 11=10,
- 38 disimpan pada : 38 mod 11 = 5
- 88 disimpan pada : 88 mod 11 = 0
- 19 disimpan pada : 19 mod 11 = 8
- 57 disimpan pada : 57 mod 11 = 2
- 72 disimpan pada : 72 mod 11 = 6
- 91 disimpan pada : 91 mod 11 = 3
- 15 disimpan pada : 15 mod 11 = 4

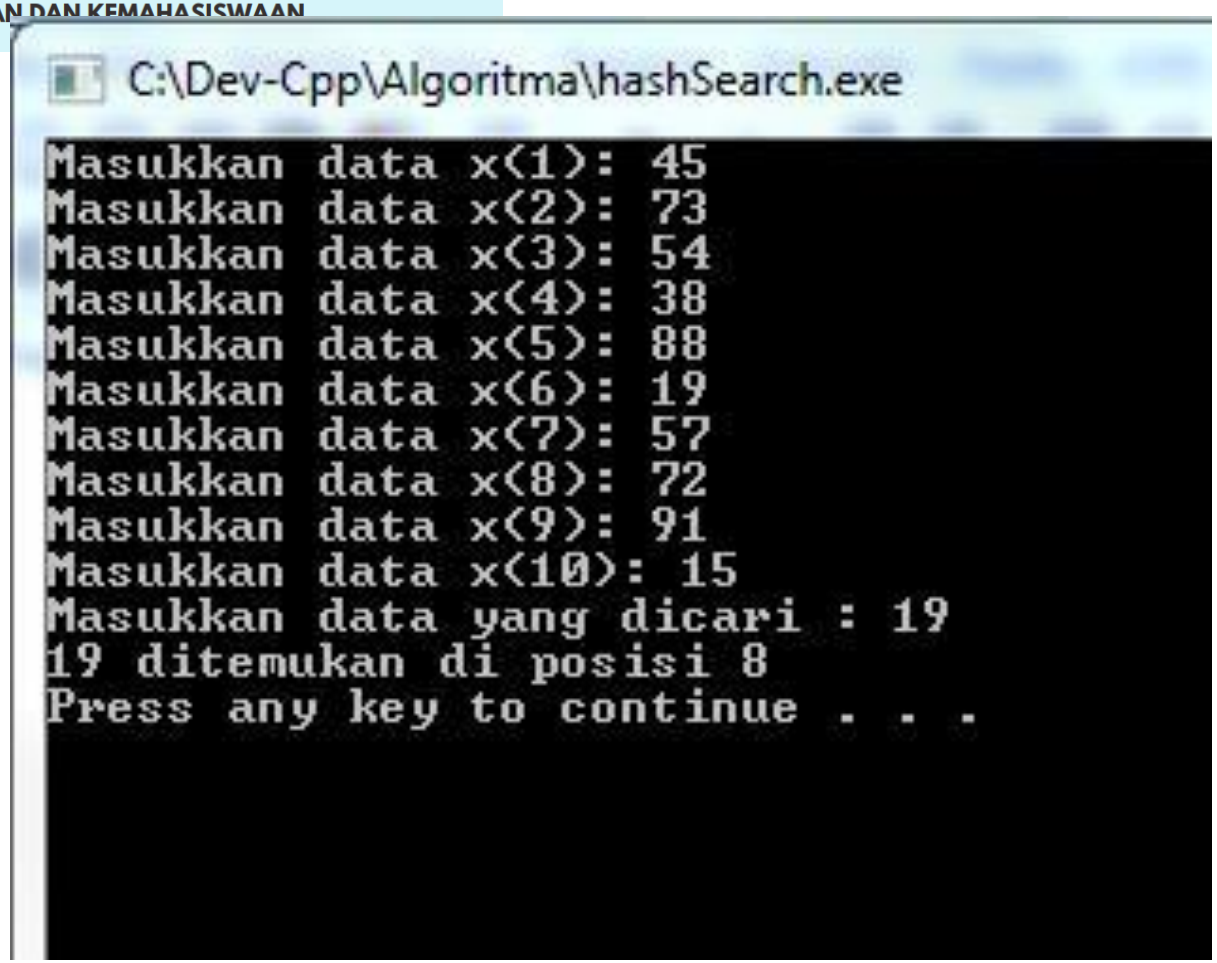
Algoritma hashSearch

```
Deklarasi:
        integer indeks, size=10, N=11;
        integer x, idx, A[N];
        fungsi myhash(x)→ integer;
Deskripsi:
        { penempatan data }
        untuk idx=1 s/d size:
                masukkan x,
                indeks = myhash(x)
                A[indeks] = x;
        { pencarian data}
        masukkan data yang dicari x,
        indeks = myhash(x)
        if (x = A[indeks])
        then tampil("data ditemukan")
        else tampil("data tdk ditemukan")
```

hashSearch.cpp

```
//hashSearch.cpp
#include <iostream>
using namespace std;
const int size=10, N=11;
int indeks, x, idx, A[N];
int myhash(int x);

int main() {
		// penempatan data
		for (idx=0; idx < size; idx++) {
			cout << "Masukkan data x" << '(' << idx+1 << "): ";
			cin >> x;
			indeks = myhash(x);
			A[indeks] = x;
		}
		// pencarian data
		cout << "Masukkan data yang dicari : ";
		cin >> x;
		indeks = myhash(x);
```

```
                if (x == A[indeks])
                        cout << x << " ditemukan di posisi " << indeks
<< '\n';
        else
                cout << x << " tidak ditemukan ! \n";

        system("PAUSE");
        return 0;
}

int myhash(int x) {
        int ix;

        ix = x % N;
        return ix;
}
```


C:\Dev-Cpp\Algoritma\hashSearch.exe
Masukkan data x(1): 45
Masukkan data x(2): 73
Masukkan data x(3): 54
Masukkan data x(4): 38
Masukkan data x(5): 88
Masukkan data x(6): 19
Masukkan data x(7): 57
Masukkan data x(8): 72
Masukkan data x(9): 91
Masukkan data x(10): 15
Masukkan data yang dicari : 19
19 ditemukan di posisi 8
Press any key to continue . . .