MODUL MATA KULIAH

ANALISIS DAN DESAIN ALGORITMA

PG167 – 3 SKS

FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BUDI LUHUR

JAKARTA
SEPTEMBER 2019

TIM PENYUSUN

Dr. Achmad Solichin, S.Kom., M.T.I
Atik Ariesta, S.Kom., M.Kom
Ita Novita, S.Kom., M.T.I

UNIVERSITAS BUDI LUHUR
FAKULTAS TEKNOLOGI INFORMASI

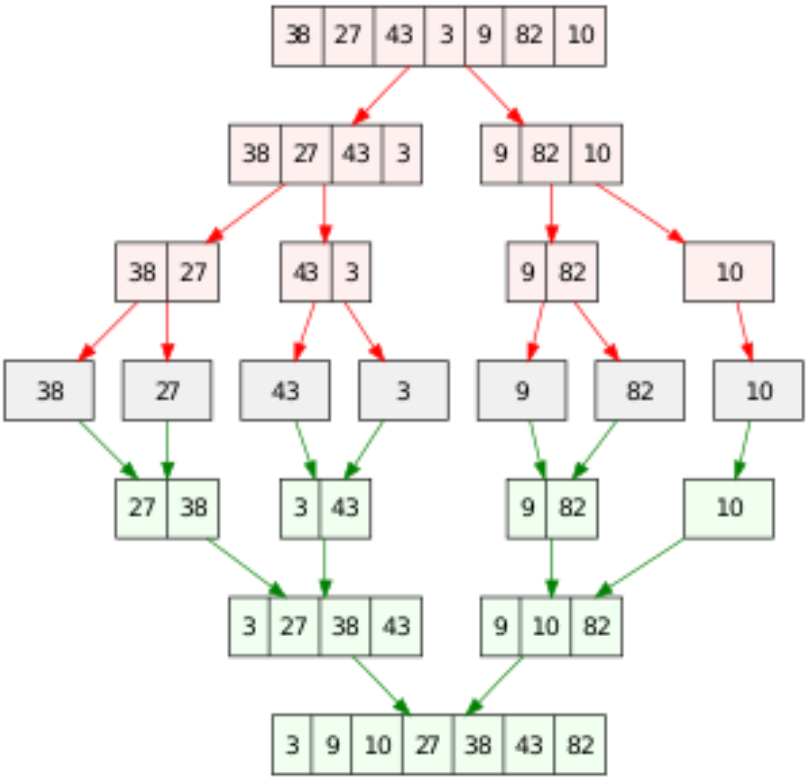
MODUL PERKULIAHAN #14
PEMECAHANAN (SPLIT) ARRAY SATU DIMENSI

Capaian Pembelajaran	:	Mahasiswa dapat memahami konsep manipulasi array satu dimensi, dan melakukan penggabungan dua atau lebih array menjadi array satu dimensi.
Sub Pokok Bahasan	:	14.1. Konsep dasar pemecahan (split) array. 14.2. Pemecahan satu buah array menjadi dua buah array. 14.3. Aplikasi pemecahan (split) dan penggabungan (merge) pada array.
Daftar Pustaka	:	1. Gaddis, nd.2011. Starting Out with C++ from Control Structures through Objects .8th. Boston: Addison-Wesley. 2. Institue of Distance & Open Learning, n.d. UNIT l Algorithms, Flowcharts & Program Design in: INTRODUCTION TO C++. p. 205 3. Sjukani,Moh .2014. Algoritma (Algoritma & Struktur Data 1) Dengan C, C++, dan Java Edisi 9", Mitra Wacana Media.

PEMECAHAN (SPLIT) ARRAY SATU DIMENSI

14.1. KONSEP PEMECAHAN (SPLIT) ARRAY

Split array merupakan operasi memecah suatu array menjadi beberapa array lain, baik dalam dimensi yang sama maupun berbeda. Pemecahan (split) array terkadang disebut juga dengan istilah pembagian (divide) array. Dalam pemrograman, operasi pemecahan array sering digunakan untuk menangani manipulasi data pada array yang berukuran besar. Misalnya, untuk melakukan pengurutan (sorting) array dalam jumlah besar, dapat dilakukan dengan memecah array menjadi beberapa array yang berukuran yang lebih kecil, lalu melakukan pengurutan pada setiap array, yang selanjutnya digabungkan kembali sebagai array utuh yang terurut. Gambar 1 menyajikan ilustrasi algoritme pengurutan Merge Sort yang memanfaatkan operasi pemecahan (split) dan penggabungan (merge) array.



Gambar 1. Ilustrasi algoritma merge-sort yang memanfaatkan operasi pemecahan (split) dan penggabungan (merge) array
(Sumber: https://en.wikipedia.org/wiki/Merge_sort)

14.2. PEMECAHAN ARRAY SATU DIMENSI MENJADI DUA ARRAY

Pada bagian ini akan dijelaskan teknik pemecahan suatu array menjadi dua buah array berbeda. Untuk mempermudah penjelasan, perhatikan contoh soal 14.1 berikut ini.

CONTOH SOAL 14.1 – PEMECAHAN ARRAY

Sudah ada array **A** satu dimensi yang dibuat dengan **int A[12],** sudah ada isinya dengan ilustrasi sebagai berikut:

	0	1	2	3	4	5	6	7	8	9	10	11
A	12	17	10	15	25	11	7	25	16	22	14	5

Sudah ada array B[12] dan C[12] yang (dianggap) masih belum ada isinya. Susun program untuk memindahkan setiap isi array A yang bernilai **GANJIL** ke array **B** dan isi array A yang bernilai **GENAP** ke array **C**, sedemikian hingga isi array B dan C menjadi sebagai berikut:

B	17	15	25	11	7	25	5					
C	12	10	16	22	14							

Analisis dan Penyelesaian Soal

Untuk menyelesaikan soal 14.1 di atas, pertama-tama kita harus dapat melakukan penelusuran pada array A[]. Selanjutnya menambahkan kondisi sesuai dengan permintaan soal tersebut untuk menyalin data ke array B[] dan C[]. Berikut ini algoritme atau kerangka pikir untuk menyelesaikan soal 14.1 di atas:

1. Inisialisasi array A[] dan isinya
2. Inisialisasi array B[] tanpa isi (kosong)
3. Inisialisasi array C[] tanpa isi (kosong)
4. Lakukan penelusuran isi array A[], jika nilainya GANJIL maka pindahkan nilainya secara berurutan ke array B[]. Namun jika nilainya GENAP maka pindahkan nilai tersebut secara berurutan ke array C[].
5. Cetak isi array B[] dan C[] untuk memastikan bahwa isi array sudah benar.

Berdasarkan algoritme (kerangka pikir) di atas, selanjutnya dapat diimplementasikan ke dalam bentuk program seperti pada Program 1 berikut ini.

Program 1. Pemecahan satu array menjadi dua array.

```
#include <stdio.h>
int main()
{
    int A[12] = {12,17,10,15,25,11,7,25,16,22,14,5};
    int B[12] = {0};
    int C[12] = {0};
    int I, J;

    //cetak isi array A
    printf("Isi Array A : ");
    for (I=0; I<12; I++) {
        printf("%3i", A[I]);
    }

    J = 0;
    for (I=0; I<12; I++) {
        if (A[I]%2==1) {
            B[J] = A[I];
            J++;
        }
    }

    J = 0;
    for (I=0; I<12; I++) {
        if (A[I]%2==0) {
            C[J] = A[I];
            J++;
        }
    }

    //cetak isi array B
    printf("\nIsi Array B : ");
    for (I=0; I<12; I++) {
        printf("%3i", B[I]);
    }

    //cetak isi array C
    printf("\nIsi Array C : ");
    for (I=0; I<12; I++) {
        printf("%3i", C[I]);
    }
    return 0;
}

```

Penjelasan Program 1:

- Pada baris 4-7 dilakukan deklarasi array A[] lengkap dengan isinya, array B[] dan array C[] masih kosong, serta beberapa variabel yang digunakan dalam program.
- Pada baris 9-13, program melakukan pencetakan isi dari array A[] agar saat dijalankan kita dapat mengetahui isi array aslinya.
- Baris 15-21 merupakan proses penelusuran isi dari array A[], disertai pemeriksaan kondisi apakah nilai / bilangan bernilai GANJIL (perhatikan baris ke-17). Jika bernilai ganjil, maka pindahkan nilai tersebut ke array B[] secara berurutan. Perhatikan variabel yang digunakan untuk melakukan penelusuran A[] berbeda dengan variabel yang digunakan untuk mengisi array B[]. *Mengapa harus berbeda? Bisakah kalo dibuat satu variabel saja?*
- Baris 23-29 merupakan proses penelusuran isi dari array A[], disertai pemeriksaan kondisi apakah nilai / bilangan bernilai GENAP (perhatikan baris ke-25). Jika bernilai genap, maka salin nilai tersebut ke array C[] secara berurutan. Perhatikan variabel yang digunakan untuk melakukan pengisian ke array B[], yaitu J, harus diberikan nilai awal dengan J=0 (baris ke-23). *Mengapa demikian? Bagaimana jika perintah tersebut dihilangkan?*
- Bagian akhir program, yaitu baris 31-41, dilakukan pencetakan isi array B[] dan C[] untuk memastikan bahwa seluruh isi array A[]telah berhasil disalin dengan benar.

Perhatikan kembali program 14.1 di atas! Seperti sudah dijelaskan di atas, baris ke 15-29 merupakan proses penelusuran dan penyalinan array A[] ke array B[] dan C[]. Proses penelusuran berlangsung dua kali. Bagian program tersebut sebenarnya dapat digabungkan, sehingga program hanya memerlukan satu kali penelusuran array A[] dan secara komputasi menjadi lebih efektif.

Program 2 berikut ini merupakan potongan program yang menggabungkan proses penyalinan isi array A[] ke array B[] dan C[]. Yang perlu diperhatikan adalah penggunaan variabel sebagai pengatur indeks array B[] dan C[] harus berbeda, karena keduanya merupakan array berbeda.

Program 2. Potongan program yang menggabungkan proses penelusuran array

```
15      J = 0; K = 0;
16      for (I=0; I<12; I++) {
17          if (A[I]%2==1) {
18              B[J] = A[I];
19              J++;
20          } else {
21              C[K] = A[I];
22              K++;
23          }
24      }
```

14.3. CONTOH APLIKASI PEMECAHAN DAN PENGGABUNGAN ARRAY

Aplikasi yang melibatkan proses pemecahan dan penggabungan array sebenarnya banyak sekali. Pada modul ini dicontohkan penerapan konsep pemecahan (split) dan penggabungan (merge) array pada algoritma pengurutan Merge-Sort. Algoritma pengurutan tersebut sudah diilustrasikan pada Gambar 1 di atas. Anda dapat mencoba Program 3 berikut ini. Namun demikian, untuk penjelasan rinci mengenai algoritma pengurutan Merge-Sort tidak akan dijelaskan pada modul ini, akan tetapi akan dijelaskan pada matakuliah Algoritma dan Struktur Data 1.

Program 3. Contoh Algoritma Pengurutan Merge-Sort.

```
1   #include "stdio.h"
2   int A[10];
3   void merge(int,int,int);
4   void merge_sort(int low,int high)
5   {
6       int mid;
7       if(low<high) {
8           mid=(low+high)/2;
9           merge_sort(low,mid);
10          merge_sort(mid+1,high);
11          merge(low,mid,high);
12      }
13  }
14  void merge(int low,int mid,int high)
15  {
16      int h,i,j,b[10],k;
17      h=low;
18      i=low;
19      j=mid+1;
```

```
20      while((h<=mid)&&(j<=high)) {
21          if(A[h]<=A[j]) {
22              b[i]=A[h]; h++;
23          } else {
24              b[i]=A[j]; j++;
25          }
26          i++;
27      }
28      if(h>mid) {
29          for(k=j;k<=high;k++) {
30              b[i]=A[k]; i++;
31          }
32      } else {
33          for(k=h;k<=mid;k++) {
34              b[i]=A[k]; i++;
35          }
36      }
37      for(k=low;k<=high;k++)
38          A[k]=b[k];
39  }
40
41  int main()
42  {
43      int num = 10,i, M;
44      printf("MERGE SORT\n");
45      //input
46      printf("Input 10 bilangan:\n");
47      for (M=0; M<10; M++) {
48          scanf("%i", &A[M]);
49      }
50
51      printf("Sebelum pengurutan");
52      for (M=0; M<10; M++) {
53          printf("%4i", A[M]);
54      }
55
56      merge_sort(0,num-1);
57
58      printf("\n\nSetelah pengurutan");
59      for (M=0; M<10; M++) {
60          printf("%4i", A[M]);
61      }
62      return 0;
63  }
```

SOAL LATIHAN

Soal 1.

Sudah ada array **NILAI** satu dimensi yang dibuat dengan **int NILAI[12],** sudah ada isinya yang merupakan nilai mahasiswa, dengan ilustrasi sebagai berikut:

	0	1	2	3	4	5	6	7	8	9	10	11
NILAI	60	80	55	90	75	40	50	85	70	65	45	55

Sudah ada array **LULUS**[12] dan **GAGAL**[12] yang (dianggap) masih belum ada isinya. Susun program untuk memindahkan setiap isi array **NILAI** yang lebih besar atau sama dengan 60 ke array **LULUS** dan sebaliknya ke array **GAGAL**. Selanjutnya tampilkan isi array LULUS dan GAGAL serta jumlah mahasiswa yang LULUS dan GAGAL!

Soal 2.

Sudah ada array **A** satu dimensi yang dibuat dengan **int A[12],** sudah ada isinya yang merupakan nilai mahasiswa, dengan ilustrasi sebagai berikut:

	0	1	2	3	4	5	6	7	8	9	10	11
A	60	80	55	90	75	40	50	85	70	65	45	55

Sudah ada array **B**[12] dan **C**[12] yang (dianggap) masih belum ada isinya. Susun program untuk menghitung rata-rata nilai mahasiswa dari array A. Selanjutnya periksa isi array A. Jika isi array A lebih dari nilai rata-rata, maka pindahkan nilainya ke array B. Sedangkan jika isi array A kurang dari nilai rata-rata, maka pindahkan ke array C. Tampilkan isi array A, B dan C!

KESIMPULAN

Konsep pemecahan (split) array penting untuk dipelajari, karena menjadi dasar bagi berbagai algoritme untuk menyelesaikan berbagai permasalahan. Salah satu penerapan konsep pemecahan array, bersama dengan konsep penggabungan array, adalah algoritma pengurutan Merge-Sort seperti sudah diilustrasikan pada Gambar 1 dan diterapkan pada Program 3.

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS BUDI LUHUR

Jl. Raya Ciledug, Petukangan Utara, Pesanggrahan

Jakarta Selatan, 12260

Telp: 021-5853753 Fax : 021-5853752

http://fti.budiluhur.ac.id