**TUGAS 1**

**DESAIN DAN ANALISIS ALGORITMA**



**Disusun oleh:**

**Adrian Maulana Muhammad (06111540000099)**

**Departemen Matematika**

**Fakultas Sains dan Analitika Data**

**Institut Teknologi Sepuluh Nopember**

**Surabaya**

**2020**

1. *A* = {2, 4, 5, 10, 15, 20}

*B* = {1, 3, 7, 11}

*C* = {1, 2, 3, 4, 5, 7, 10, 11, 15, 20}, *C* = *A* U *B*

Penyelesaian:

* + Permasalahan:

Penggabungan elemen-elemen integer dari array *A* dan *B* menjadi elemen-elemen integer yang sudah tersortir untuk array yang baru, yaitu array *C*.

* + Input:
    - Array *A* yang beranggotakan 6 elemen integer, yaitu 2, 4, 5, 10, 15, dan 20.
    - Array *B* yang beranggotakan 4 elemen integer, yaitu 1, 3, 7, dan 11.
  + Output:

Array *C* yang beranggotakan elemen-elemen yang sudah tersortir dari Array *A* dan array *B.*

* + Strategi:
    1. Inisialisasi array *C*
    2. Menyelidiki elemen dari array *A* dan *B*.
    3. Membandingkan elemen yang diselidiki.
    4. Memasukan elemen yang sesuai ke array *C*.
  + Pseudocode:

combineSort(A, B):

* + 1. **Start**
    2. **Int[]** C
    3. **Int** a,b = 0
    4. **For** i < C.length **do**
    5. **If** (a < A.length) && (b < B.length) **do**
    6. **If** B[b} < A[a] **do**
    7. C[i] = B[b]
    8. b++
    9. **Else**
    10. C[i] = A[a]
    11. a++
    12. **End** **if**
    13. **Else if** b < B.length **do**
    14. C[i] = B[b]
    15. b++
    16. **Else**
    17. C[i] = A[a]
    18. a++
    19. **End if**
    20. **End for**
    21. **Return** C
    22. **End**
  + Source Code Program:

package tugasdaa;

public class TugasDAA {

public static void main(String[] args) {

// inisialisasi array A dan B

int[] A = {2, 4, 5, 10, 15, 20};

int[] B = {1, 3, 7, 11};

// panggil method dan masukan hasil ke array C

int[] C = combineSort(A, B);

// print array C

for(int i = 0; i < C.length; i++){

System.out.print(C[i] + " ");

}

}

// method combineSort untuk menggabung dan sortir array

public static int[] combineSort(int[] A, int[] B){

// inisialisasi Array C

int[] C = new int[A.length+B.length];

// mendefinisi indeks elemen untuk array A dan B

int a = 0, b = 0;

// mengisi array C

for(int i = 0; i < C.length; i++){

// jika elemen dari array A dan B masih ada untuk diselidiki

if(a < A.length && b < B.length){

// jika elemen dari array B lebih kecil

if(B[b] < A[a]){

// masukan elemen ke array C

C[i] = B[b];

b++;

}

// jika elemen dari array A lebih kecil

else{

// masukan elemen ke array C

C[i] = A[a];

a++;

}

}

// jika sudah tidak ada elemen dari array A untuk diselidiki

// tapi masih ada elemen dari array B untuk diselidiki

else if(b < B.length){

// masukan elemen ke array C

C[i] = B[b];

b++;

}

// jika sudah tidak ada elemen dari array B untuk diselidiki

// tapi masih ada elemen dari array A untuk diselidiki

else{

// masukan elemen ke array C

C[i] = A[a];

a++;

}

}

return C;

}

}

* + Output Program:



1. Mencari elemen puncak pada array.

Penyelesaian:

* + Permasalahan:

Pandang sederetan *n* buah bilangan {a1, a2, …, an}. Ingin dicari bilangan max/puncak yang merupakan bilangan terbesar dari *n* buah sederetan bilangan tersebut.

* + Input:
    - Array *A* dengan *n* elemen bilangan.
    - Indeks batas awal dari array
    - Indeks batas akhir dari array
  + Output:

Bilangan puncak = max{a1, a2,…, an}.

* + Strategi:
    1. Bagi array A menjadi dua sub array yang hampir sama besar.
    2. Array bagian pertama (kiri) dicari nilai maksimalnya (a).
    3. Array bagian kedua (kanan) dicari nilai maksimalnya (b).
    4. Bandingkan hasil dari kedua sub array.
    5. Bila a > b maka maksimalnya adalah a, selain itu maksimalnya adalah b.
  + Pseudocode:

maxArray{A, start, end):

* + 1. **Start**
    2. **Int** max
    3. **If** start == end **do**
    4. max = A[start]
    5. **Else if** start + 1 == end **do**
    6. **If** A[start] < A[end] **do**
    7. max = A[end]
    8. **Else**
    9. max = A[start]
    10. **End if**
    11. **Else**
    12. **Int** mid = start + (end - start)/2
    13. **Int** left = **maxArray**(A, start, mid)
    14. **Int** right = **maxArray**(A, mid + 1, end)
    15. **If** left > right **do**
    16. max = left
    17. **Else**
    18. max = right
    19. **End if**
    20. **End** **if**
    21. **Return** max
    22. **End**
  + Source Code Program:

package tugasdaa;

public class TugasDAA {

public static void main(String[] args) {

// inisialisasi array A

int[] A = {0, 5, 2, 3, 9, 4, 8, 6, 7};

// inisiasi batas awal dan akhir

int start = 0;

int end = A.length-1;

// panggil method

int res = maxArray(A, start, end);

// print result

System.out.println("Array:");

for(int i = 0; i < A.length; i++){

System.out.print(A[i] + " ");

}

System.out.println('\n' + "Elemen puncak: " + '\n' + res);

}

// method untuk mencari elemen puncak dengan divide & conquer

public static int maxArray(int[] A, int start, int end){

// inisialisasi variable elemen puncak

int max;

// jika tersisa 1 elemen pada array

if(start == end){

max = A[start];

// jika tersisa 2 elemen pada array

} else if(start + 1 == end){

// membandingkan 2 elemen tersebut

if(A[start] < A[end]){

max = A[end];

} else{

max = A[start];

}

// jika terdapat beberapa elemen pada array

} else{

// membagi array menjadi 2 sub-array

int mid = start + (end - start)/2;

// rekursif terhadap sub-array

int left = maxArray(A, start, mid);

int right = maxArray(A, mid+1, end);

// membandingkan elemen puncak dari kedua sub-array

if(left > right)

max = left;

else

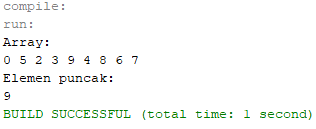
max = right;

}

return max;

}}

* + Hasil Output Program:



1. Implementasi Merge Sort

Penyelesaian:

* + Permasalahan:

Pandang sederetan *n* buah bilangan (a1, a2,…, an). Ingin dilakukan pengurutan (urut naik) sehingga menjadi (aj1, aj2,…, ajn) dengan aj1 < aj2 < … < ajn.

* + Input:

Array input dengan *n* elemen bilangan.

* + Output:

Array output dengan elemen terurut naik.

* + Strategi:
    1. Bagi array A menjadi dua sub-array yang hampir sama besar.
    2. Array bagian pertama (kiri) diurut.
    3. Array bagian kedua (kanan) diurut.
    4. Menggabung kedua sub-array menjadi satu array yang terurut.
  + Source Code Program:

package tugasdaa;

import java.util.Arrays;

public class TugasDAA {

public static void main(String[] args) {

// array input

int arr[] = {12, 11, 13, 5, 6, 7};

System.out.println("Array Input");

printArray(arr);

// array output

TugasDAA ob = new TugasDAA();

ob.sort(arr, 0, arr.length-1);

System.out.println("\nArray Output");

printArray(arr);

}

public void merge(int[] arr, int start, int mid, int end){

/\* ukuran dari dua subarray \*/

int n1 = mid - start + 1;

int n2 = end - mid;

/\* array kosong (temp) \*/

int L[] = new int [n1];

int R[] = new int [n2];

/\* masukan elemen ke array temp \*/

for (int i=0; i<n1; ++i)

L[i] = arr[start + i];

for (int j=0; j<n2; ++j)

R[j] = arr[mid + 1+ j];

/\* Gabung kedua array temp \*/

// inisialisasi indeks untuk sub-array

int i = 0, j = 0;

// inisialisasi indeks array yang sudah digabung

int k = start;

while (i < n1 && j < n2){

if (L[i] <= R[j]){

arr[k] = L[i];

i++;

} else{

arr[k] = R[j];

j++;

} k++;

}

/\* jika masih ada elemen dari subarray kiri \*/

while (i < n1){

arr[k] = L[i];

i++;

k++;

}

/\* jika masih ada elemen dari subarray kanan \*/

while (j < n2){

arr[k] = R[j];

j++;

k++;

}

}

// Main function that sorts arr[l..r] using

// merge()

void sort(int arr[], int l, int r){

if (l < r){

// Find the middle point

int m = (l+r)/2;

// Sort first and second halves

sort(arr, l, m);

sort(arr , m+1, r);

// Merge the sorted halves

merge(arr, l, m, r);

}

}

/\* A utility function to print array of size n \*/

static void printArray(int arr[]){

int n = arr.length;

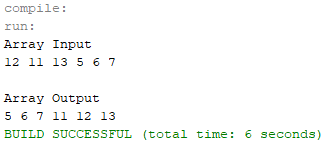
for (int i=0; i<n; ++i)

System.out.print(arr[i] + " ");

System.out.println();

}

* Hasil Output Program:



* Sudah membaca bab tentang analisa algoritma.