Mata Kuliah : Dasar Dasar Pemrograman (Praktek)
Kode Mata Kuliah : KBTI4104
Waktu : Jumat (8.20 – 11.40)
Jumlah SKS : 4 SKS
Nama Dosen : Ani Rahmani
Minggu ke : 7 (Tujuh)
Tanggal : 30-10-2015
Judul Materi : Macam Macam Algoritma Sorting

Algoritma Sorting atau pengurutan digunakan untuk mengolah data dan mengurutkannya secara *ascending* atau *descending* sesuai kebutuhan. Algoritma Sorting terbagi menjadi 9 macam, yaitu Bubble Sort, Selection Sort, Insertion Sort, Shell Sort, Merge Sort, Quick Sort, Heap Sort, Bucket Sort, Radix Sort.

Contoh : Mengurutkan secara *Ascending*

A	4	3	6	3	10	23	5	7	8

a. Bubble Sort

Bubble sort dilakukan dengan membandingkan masing-masing item dalam suatu list secara berpasangan, menukar item dilakukan sampai akhir list secara berurutan.

6 5 3 1 8 7 2 4

Berikut contoh algoritma Bubble Sort dengan menggunakan data array A diatas

Algoritma

```
Begin
  |  i,j : integer
  |  for  i ← 1 to 8 do
  |    |  for  j ← i + 1 to 9 do
```

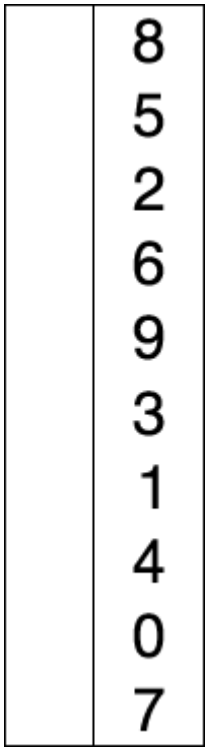
```
  |   |   |  if (A[ i ] > A[ j ])
  |   |   |   | then  A[ i ] ←A[ i ] + A[ j ]
  |   |   |   |       A[ j ] ← A[ i ] – A[ j ]    // proses swaping
  |   |   |   |       A[ i ] ← A[ i ] – A[ j ]
  |   |   |  endif
  |   |  endfor
  |  endfor
End
```

Bahasa C

```
#include <stdio.h>
int main(){
   int i,j;
   for(i=0;i<8;i++){
      for(j=i+1;j<9;j++){
         if(A[i]>A[j]){
            A[i]=A[i]+A[j];
            A[j]=A[i]-A[j];
            A[i]=A[i]-A[j];
         }
      }
   }
   return 0;
}
```

b. Selection Sort

Cara kerja Selection Sort yaitu memilih elemen dengan nilai paling rendah dan menukar elemen yang terpilih dengan elemen ke-i. nilai dari i dimulai dari 1 ke n adalah jumlah total elemen dikurangi 1.



Berikut contoh algoritma Selection Sort

Algoritma

```
Begin
   |   i,j,k,Min : integer
   |   for i ← 1 to 8 do
   |   |   Min ← A[ i ]
   |   |   k ← i
   |   |   for  j ← i  to  9 do
   |   |   |   if (Min > A[ j ])
   |   |   |   |  then  Min ← A[ j ]
   |   |   |   |          k ← j
```

```
  |   |   | endif
  |   | endfor
  |   | if (k =/= i )
  |   |   | then  A[ i ] ← A[ i ] + A[ k ]
  |   |   |       A[ k ] ← A[ i ] – A[ k ]
  |   |   |       A[ i ] ←A[ i ] – A[ k ]
  |   | endif
  | endfor
End
```

Bahasa C

```
#include <stdio.h>
int main(){
   int i,j,k,Min;
   for(i=0;i<8;i++){
      Min=A[i];
      k=i;
      for(j=i;j<9;j++){
         if(Min>A[j]){
            Min=A[j];
            k=j;
         }
      }
      if(k!=i){
         A[i]=A[i]+A[k];
         A[k]=A[i]-A[k];
         A[i]=A[i]-A[k];
      }
```

```
    }
    return 0;
}
```

c. Insertion Sort

Cara kerja insertion sort sebagaimana namanya.Pertama-tama, dilakukan proses iterasi, dimana di setiap iterasi insertion sort memindahkan nilai elemen,kemudian menyisipkannya berulang-ulang sampai ketempat yang tepat. Begitu seterusnya dilakukan. Dari proses iterasi, seperti biasa, terbentuklah bagian yang telah di-sorting dan bagian yang belum.

Berikut contoh algoritma insertion sort.

Algoritma

Begin

| c,d : integer

| for c ← 2 to 9 do

| | d ← c

| | while (d>1 AND A[d]<A[d-1]) do

| | | A[d] ← A[d]+ A[d-1]

| | | A[d-1] ← A[d] – A[d-1]

| | | A[d] ← A[d] – A[d-1]

| | | d ← d - 1

```
  |   | endwhile
  | endfor
end
```

Bahasa C

```
#include <stdio.h>
int main()
{
  int c, d;
  for (c = 1 ; c <= 8; c++) {
    d = c;

    while ( d > 0 && array[d] < array[d-1]) {
      A[d]  =A[d]+a[d-1];
      A[d-1]=A[d]-A[d-1];
      A[d]  =A[d]-A[d-1];
      d--;
    }
  }
  return 0;
}
```

d. Shell Sort

Shell sort merupakan metode pengurutan yang hampir sama dengan insertion sort, dimana pada setiap nilai i dalam n/i item diurutkan. Pada setiap pergantian nilai, i dikurangi sampai 1 sebagai nilai terakhir.

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
input data:	62	83	18	53	07	17	95	86	47	69	25	28
after 5-sorting:	17	28	18	47	07	25	83	86	53	69	62	95
after 3-sorting:	17	07	18	47	28	25	69	62	53	83	86	95
after 1-sorting:	07	17	18	25	28	47	53	62	69	83	86	95

Berikut contoh algoritma shell sort.

Algoritma

```
Begin
   |  i,j,m : integer
   |  break : Boolean
   |  break ← .False.
   |  m ← 9 div 2
   |  while (m>0) do
   |     |  for j ← m to 9 do
   |     |     |  i ← j – m
   |     |     |  while (i >= 0 AND break=.False.) do
   |     |     |     |  if (A[i + m + 1]>=A[i + 1])
   |     |     |     |     |  then break← .True.
   |     |     |     |     |  else  A[i + 1] ← A[ i + 1] + A[i + m + 1]
   |     |     |     |     |        A[i + m + 1] ← A[i + 1] – A[i + m + 1]
   |     |     |     |     |        A[i + 1] ← A[i + 1] – A[i + m + 1]
   |     |     |     |  endif
   |     |     |     |  i ← i – m
   |     |     |  endwhile
   |     |  endfor
   |     |  m ← m div 2
   |  endwhile
End
```

Bahasa C

```
#include <stdio.h>

int main(){
```

```
    int i,j,m,temp;

    for(m = 9/2;m>0;m/=2){
       for(j=m;j<9;j++){
            for(i=j-m;i>=0;i-=m){
                if(L[i+m]>=L[i]) break;
                else{
                    temp = L[i];
                    L[i] = L[i+m];
                    L[i+m] = temp;
                }
            }
        }
    }

    return 0;

}
```

e. Merge Sort

Merge Sort merupakan jenis pengurutan yang dirumuskan dalam 3 tahap berpola divide-and-conquer.

- Divide = Memilah elemen – elemen dari rangkaian data menjadi dua bagian.
- Conquer = setiap bagian dengan memanggil prosedur merge sort secara rekursif

Kombinasi = Mengkombinasikan dua bagian tersebut secara rekursif untuk mendapatkan rangkaian data yang berurutan. Proses rekursi berhenti jika mencapai elemen dasar. Hal ini terjadi jika bagian yang akan diurutkan menyisakan tepat satu elemen. Sisa pengurutan satu elemen tersebut menandakan bahwa bagian tersebut telah terurut sesuai rangkaian yagn dikehendaki.

6 5 3 1 8 7 2 4

Berikut ini contoh algoritma merge sort

Algoritma	**Modul yang dipanggil**	
Begin | MergeSort(1,9) end	Procedure MergeSort(low:int,high:int) | mid : int | if (low<high) | | then mid ←(low+high)div 2 | | MergeSort(low,mid) | | MergeSort(mid+1,high) | | Merge(low,mid,high) | endif endProcedure	Procedure Merge(low:int,mid:int,high:int) | h,i,j,k : int | b[50] : int | h←low | i ← low | j ← mid +1 | while (h<=mid AND j<=high) | | if (A[h+1]<A[j+1]) | | | then b[i +1]←A[h+1] | | | h ← h+1 | | | else b[i +1]←A[j +1] | | | j ← j+1 | | endif | | i ← i + 1 | | if (h>mid) | | | then for k ← j to high do | | | | b[i + 1]←A[k+1] | | | | i ← i + 1 | | | else for k←h to mid do | | | | b[i+1]←A[k+1] | | | | i ← i + 1 | | endif | | for k←low to high do | | | A[k+1]← b[k+1] | | endfor

		| endwhile end

Bahasa C
 	
#include <iostream>
using namespace std;
void MergeSort(**int** low,**int** high);
void Merge(**int** low,**int** mid,**int** high);
 	
int main(){
MergeSort(1,9);
return 0;
}
	void MergeSort(**int** low, **int** high){
	int mid;
	if(low<high){
	mid=(low+high)/2;
	MergeSort(low,mid);
	MergeSort(mid+1,high);
	Merge(low, mid, high);
	}
	}
		void Merge(**int** low, **int** mid, **int** high){
		int h,i,j,k,b[50];
		h=low;
		i=low;
		j=mid+1;
		while((h<=mid)&&(j<=high)){
		if(A[h]<A[j]){
		b[i]=A[h];
		h++;
		}**else**{
		b[i]=A[j];
		j++;
		}
		i++;
		}
		
		if(h>mid){
		
		for(k=j;k<=high;k++){
		b[i]=A[k];
		i++;
		}
		}**else**{
		for(k=h;k<=mid;k++){
		b[i]=A[k];
		i++;

		}
		}
		
		for(k=low;k<=high;k++){
		A[k]=b[k];
		}
		}

f. Quick Sort

Quick sort merupakan metode pengurutan dengan algoritma berdasarkan pola divide-and-conquer. Algoritma ini hanya memiliki 2 langkah sebagai berikur :

- Divide = bisa dikatakan Memilah rangkaian data menjadi dua sub-rangkaian A[p…q-1] dan A[q+1…r] dimana setiap elemen A[p…q-1] adalah kurang dari atau sama dengan A[q] dan setiap elemen pada A[q+1…r] adalah lebih besar atau sama dengan elemen pada A[q]. A[q] disebut sebagai elemen pivot. Perhitungan pada elemen q merupakan salah satu bagian dari prosedur pemisahan.
- Conquer = dengan cara Mengurutkan elemen pada sub-rangkaian secara rekursif. Pada algoritma quicksort, langkah ”kombinasi” tidak di lakukan karena telah terjadi pengurutan elemen – elemen pada sub-array

berikut contoh quick sort :

6 5 3 1 8 7 2 4

Algoritma	**Modul yang dipanggil**
Begin | quicksort(A,0,8)	Procedure quicksort(A[]:int,kiri:int,kanan:int) | tmp,i,j,pivot : int

end	| i ← kiri | j ← kanan | pivot←A[(kiri+kanan)div 2 +1] | while (i<=j) | | while (A[i + 1]<pivot) | | | i ← i + 1 | | endwhile | | while (A[j+1]>pivot) | | | j ← j – 1 | | endwhile | | if (i<=j) | | | then tmp ← A[i + 1] | | | A[i + 1] ← A[j + 1] | | | A[j + 1] ← tmp | | | i ← i + 1 | | | j ← j – 1 | | endif | endwhile | if (kiri < j) | | then quicksort(A,kiri,j) | endif | if (i < kanan) | | then quicksort(A,i,kanan) | endif end
Bahasa C	`void quicksort(int*A,int kiri,int kanan){`

```
#include <stdio.h>

void quicksort(int*A,int kiri,int kanan);


int main(){

   quicksort(A,0,8);

   return 0;

}
```

```
   int tmp,i=kiri,j=kanan;

   int pivot=A[(kiri+kanan)/2];

   while(i<=j){

      while(A[i]<pivot){

         i++;

      }

      while(A[j]>pivot){

         j--;

      }

      if(i<=j){

         tmp=A[i];

         A[i]=A[j];

         A[j]=tmp;

         i++;

         j--;

      }

   }

   if(kiri<j){

      quicksort(A,0,8);

   }

   if(i<kanan){

      quicksort(A,i,kanan);

   }

}
```

g. Heap Sort

Heap sort merupakan metode sorting yang menggunakan struktur data heap, dengan nilai parent selalu lebih besar dari pada nilai childnya.
adapun langkah algoritma nya sebagai berikut :

- Buat suatu heap

- Ambil isi dari root, lalu masukkan kedalam sebuah array.

- Hapus element root dengan mempertahankan properti heap.
- Ulangi sampai tree menjadi kosong

Berikut contoh Heap Sort :

Algoritma

```
Procedure Heapify(A:int,i:int){
   le <- left(i)

   ri <- right(i)

   if (le<=heapsize)and(A[le]>A[i])

      largest <- le

   else

      largest <- i

   if
(ri<=heapsize)and(A[ri]>A[largest])

      largest <- ri

   if (largest != i) {

      exchange A[i] <-> A[largest]

      Heapify(A, largest)

   }

}
```

```
Procedure BuildHeap(A) {
   heapsize <- length(A)
   for i <- floor( length/2
) downto 1
      Heapify(A, i)
}
```

```
Procedure Heapsort(A) {
   BuildHeap(A)
   for i <- length(A) downto
2 {
      exchange A[1] <-> A[i]
      heapsize <- heapsize -
      Heapify(A, 1)
}
```

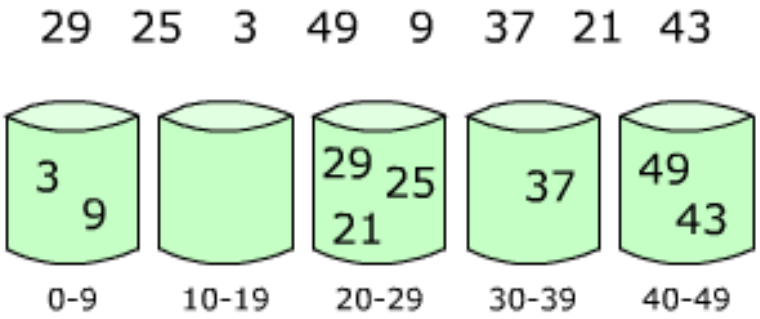
h. Bucket Sort

Bucket Sort merupakan algoritma sorting yang mempartisi deret angka menjadi beberapa deret yang kemudian dianalogikan menjadi ember.
Algoritma nya sebagai berikut :

Cari nilai maksimum dan minimum di dalam array.
Inisialisasi array bucket Daftar <> unsur (ukuran maxValue – minValue + 1)

Pindahkan elemen dalam array untuk bucket

Write bucket keluar (dalam rangka) ke array yang asli
berikut contoh bucket sort :




```
function bucketSort(array, n) is
  buckets ← new array of n empty lists
  for i = 0 to (length(array)-1) do
    insert array[i] into buckets[msbits(array[i], k)]
  for i = 0 to n - 1 do
    nextSort(buckets[i]);
  return the concatenation of buckets[0], ...., buckets[n-1]
```

i. Radix Sort

Radix Sort adalah metode sorting yang ajaib yang mana mengatur pengurutan nilainya tanpa melakukan beberapa perbandingan pada data yang dimasukkan. Secara umum yang proses yang dilakukan dalam metode ini adalah mengklasifikasikan data sesuai dengan kategori terurut yang tertentu dan dalam tiap kategorinya dilakukan pengklasifikasian lagi dan seterusnya sesuai dengan kebutuhan.
Secara kompleksitas waktu, radix sort termasuk ke dalam Divide and Conquer.Namun dari segi algoritma untuk melakukan proses pengurutan, radix sort tidak termasuk dalam Divide and Conquer.

```
PROC radixsort = (REF []INT array) VOID:
(
    [UPB array]INT zero;
```

```
    [UPB array]INT one;
    BITS mask := 16r01;
    INT zero_index  := 0,
        one_index   := 0,
        array_index := 1;

    WHILE ABS(mask) > 0 DO
        WHILE array_index <= UPB array DO
            IF (BIN(array[array_index]) AND mask) = 16r0 THEN
                zero_index +:= 1;
                zero[zero_index] := array[array_index]
            ELSE
                one_index +:= 1;
                one[one_index] := array[array_index]
            FI;
            array_index +:= 1
        OD;

        array_index := 1;
        FOR i FROM 1 TO zero_index DO
            array[array_index] := zero[i];
            array_index +:= 1
        OD;

        FOR i FROM 1 TO one_index DO
            array[array_index] := one[i];
            array_index +:=1
        OD;

        array_index := 1;
        zero_index := one_index := 0;
        mask := mask SHL 1
    OD
);

main:
(
    [10]INT a;
    FOR i FROM 1 TO UPB a DO
        a[i] := ROUND(random*1000)
    OD;

    print(("Before:", a));
    print((newline, newline));
    radixsort(a);
    print(("After: ", a))
)
```