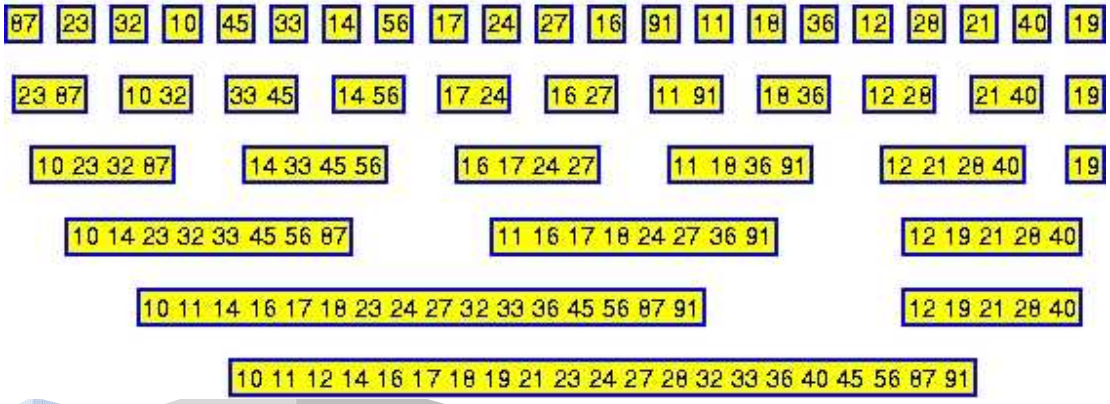
Struktur Data dan Algoritma

Metode Pengurutan Merge Sort

Dr. Taufik Fuadi Abidin, M.Tech
Irvanizam Zamanhuri, M.Sc



Merge Sort

Metode pengurutan *merge sort* adalah metode pengurutan lanjut, sama dengan metode *Quick Sort*. Metode ini juga menggunakan konsep *devide and conquer* yang membagi data *S* dalam dua kelompok yaitu S_1 dan S_2 yang tidak beririsan (*disjoint*). Proses pembagian data dilakukan secara rekursif sampai data tidak dapat dibagi lagi atau dengan kata lain data dalam sub bagian menjadi tunggal.

Setelah data tidak dapat dibagi lagi, proses penggabungan (*merging*) dilakukan antara sub-sub bagian dengan memperhatikan urutan data yang diinginkan (*ascending/kecil ke besar* atau *descending/besar ke kecil*). Proses penggabungan ini dilakukan sampai semua data tergabung dan terurut sesuai urutan yang diiginkan. Kompleksitas algoritma *merge sort* adalah O(n log n).

Secara umum, algoritma *merge sort* dapat diimplementasikan secara rekursif. Fungsi rekursif adalah sebuah fungsi yang didalam implementasinya memanggil dirinya sendiri. Pemanggilan diri sendiri ini berakhir jika kondisi tertentu terpenuhi (*terminated condition is true*). Pada contoh berikut ini, *terminated condition* dari proses rekursif *mergesort* akan berakhir jika data tidak dapat dibagi lagi (data tunggal telah diperoleh). Dengan kata lain, proses pembagian data dilakukan terus selama `S.size > 1` (belum tunggal).

Ilustrasi dari algoritma *merge sort* adalah sebagai berikut:

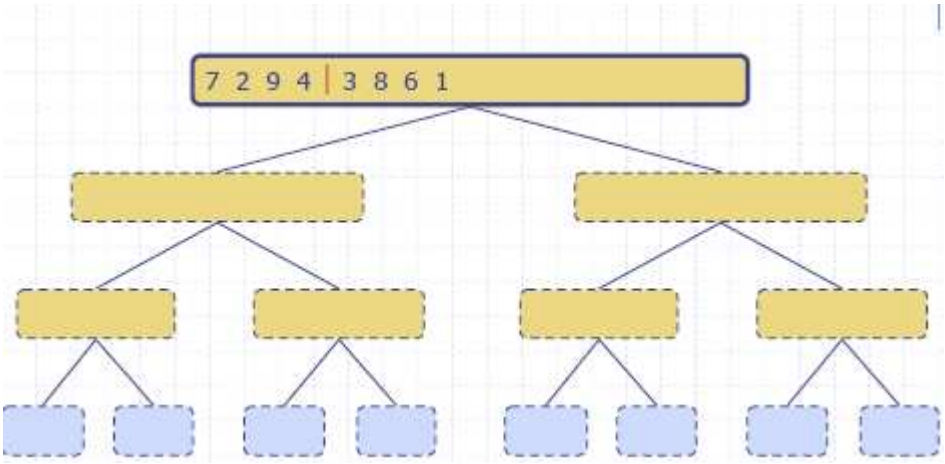
```
Algoritma mergesort(S,C)
Input: n elemen dari data S dan comparator C
Output: data S terurut berdasarkan C

if S.size()>1
   (S1,S2) ← partisi(S,n/2)
    mergesort(S1,C)
    mergesort(S2,C)
    S ← merge(S1,S2)
```

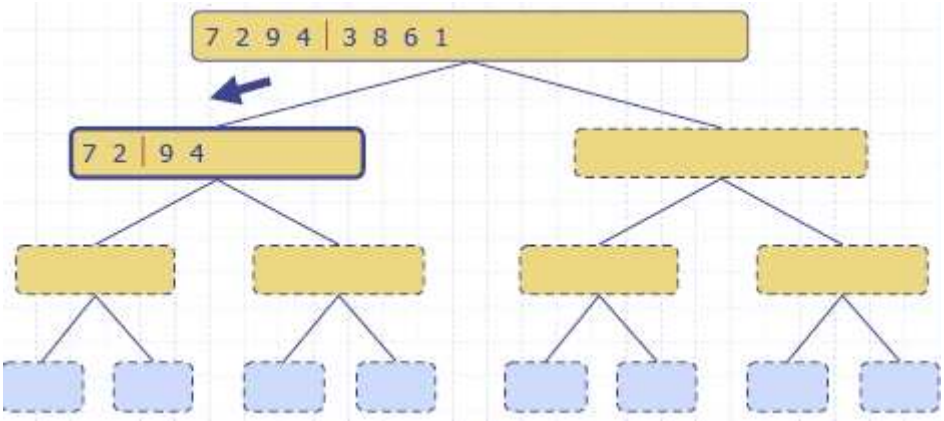
Fungsi *merge* berfungsi untuk menggabungkan hasil pengurutan dari sub bagian S_1 dan S_2 berdasarkan urutan tertentu (*ascending* atau *descending order*). Kompleksitas proses penggabungan ini (*merging*) adalah O(n).

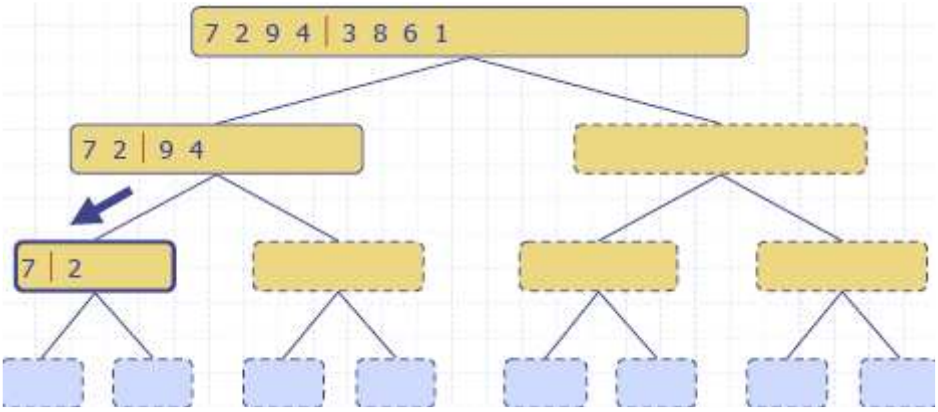
Algoritma dari proses penggabungan adalah:

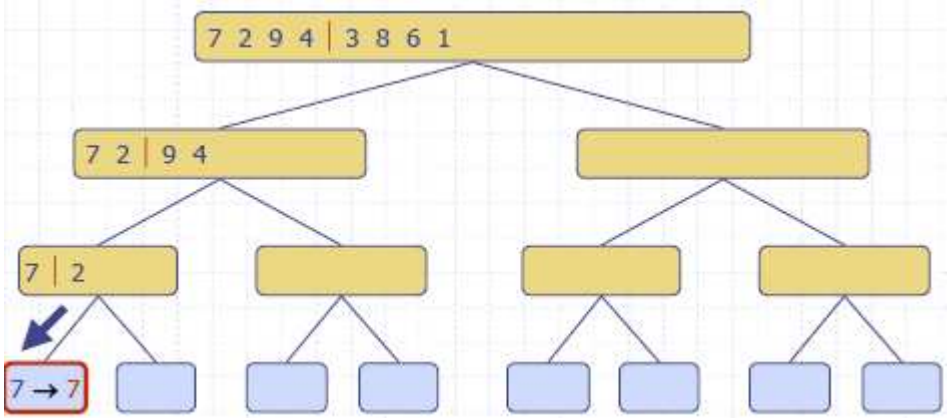
```
Algoritma merge(S1,S2)
Input: data S1 dan S2 dengan n/2 elemen per data
Output: data terurut dari penggabungan S1 ∪ S2

S ← data kosong
while not S1.empty() and not S2.empty()
   if S1.first().elemen() < S2.first().elemen()
       S.insert(S1.remove(S1.first())
   else
       S.insert(S2.remove(S2.first())
while not S1.empty()
   S.insert(S1.remove(S1.first())
while not S2.empty()
   S.insert(S2.remove(S2.first())
return S
```

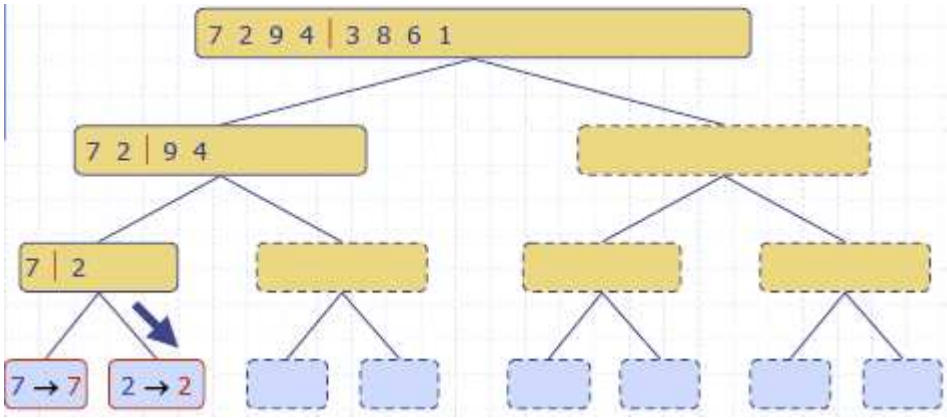
Ilustrasi Algoritma Merge Sort

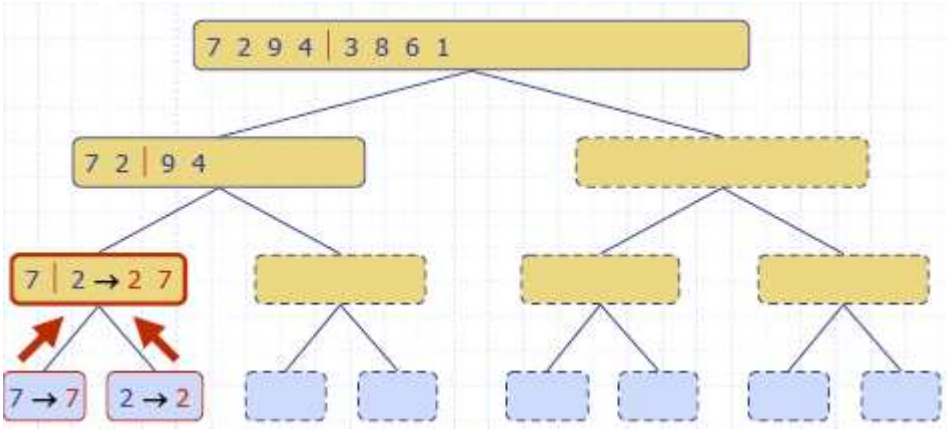


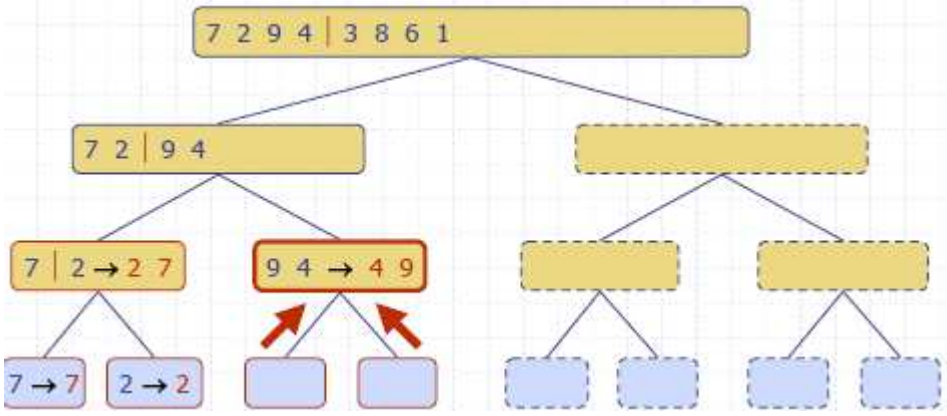

7 2 9 4 | 3 8 6 1
7 2 | 9 4

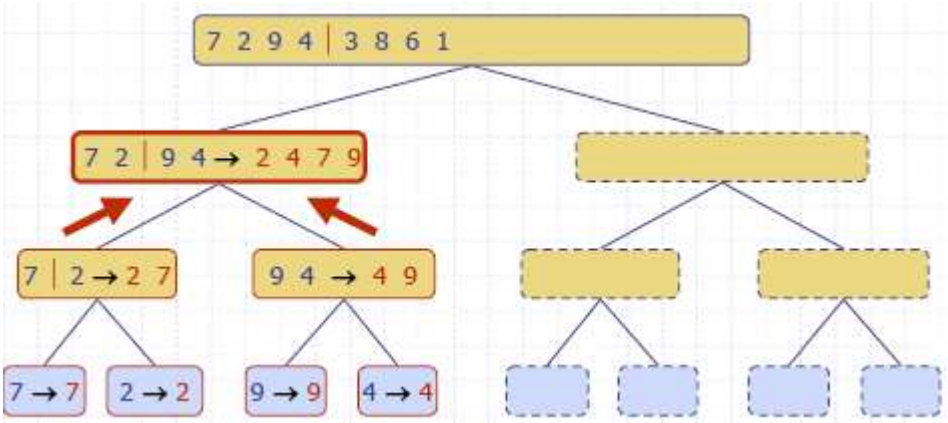

7 2 9 4 | 3 8 6 1
7 2 | 9 4
7 | 2

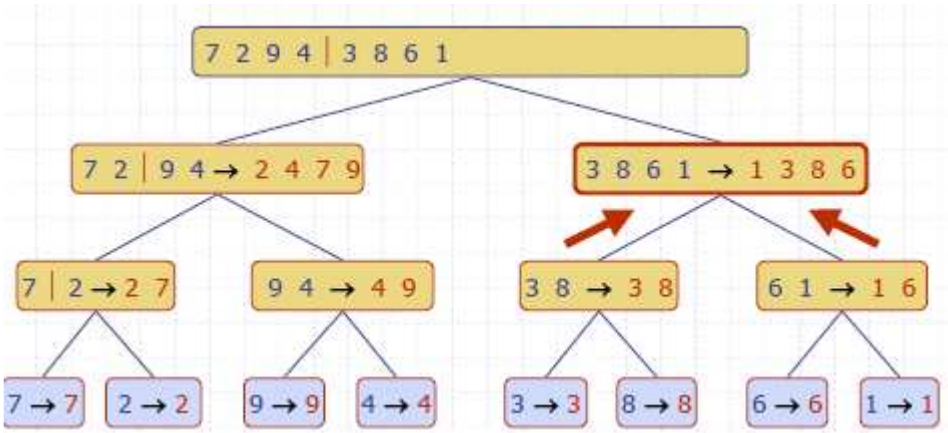

7 2 9 4 | 3 8 6 1
7 2 | 9 4
7 | 2
7 → 7

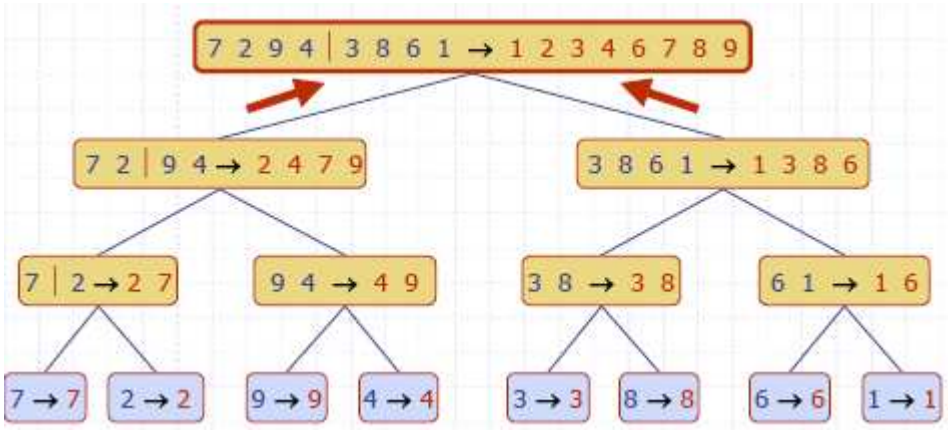

7 2 9 4 | 3 8 6 1
7 2 | 9 4
7 | 2
7 → 7
2 → 2


7 2 9 4 | 3 8 6 1
7 2 | 9 4
7 | 2 → 2 7
7 → 7
2 → 2


7 2 9 4 | 3 8 6 1
7 2 | 9 4
7 | 2 → 2 7
9 4 → 4 9
7 → 7
2 → 2


7 2 9 4 | 3 8 6 1
7 2 | 9 4 → 2 4 7 9
7 | 2 → 2 7
9 4 → 4 9
7 → 7
2 → 2
9 → 9
4 → 4


7 2 9 4 | 3 8 6 1
7 2 | 9 4 → 2 4 7 9
3 8 6 1 → 1 3 8 6
7 | 2 → 2 7
9 4 → 4 9
3 8 → 3 8
6 1 → 1 6
7 → 7
2 → 2
9 → 9
4 → 4
3 → 3
8 → 8
6 → 6
1 → 1


7 2 9 4 | 3 8 6 1 → 1 2 3 4 6 7 8 9
7 2 | 9 4 → 2 4 7 9
3 8 6 1 → 1 3 8 6
7 | 2 → 2 7
9 4 → 4 9
3 8 → 3 8
6 1 → 1 6
7 → 7
2 → 2
9 → 9
4 → 4
3 → 3
8 → 8
6 → 6
1 → 1

Latihan

Urutkan data berikut menggunakan metode *merge sort* dan tunjukkan langkah per langkah proses pengurutannya.

40 75 12 68 72 9 56 18 60 5 20 65 2

Implementasi Merge Sort

dikutip dari buku "A Book in C"

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

void mergesort(int key[],int n);
void merge(int a[],int b[],int c[],int m, int n);

int main(void){
  int i, key[8]={12,10,45,11,9,23,4,17};
  mergesort(key,8);
  for(i=0;i<8;++i)
    printf("%d\t",key[i]);
  return 1;
}

void mergesort(int key[], int n){
  int j,k,m,*w;
  for (m=1;m<n;m*=2);
  if (m!=n){
    printf("jumlah arraynya salah ");
    exit(1);
  }
  w = calloc(n,sizeof(int));
  assert(w!=NULL);
  for (k=1;k<n;k*=2){
    for(j=0;j<n-k;j+=2*k)
      merge(key+j,key+j+k,w+j,k,k);
    for(j=0; j<n; ++j)
      key[j]=w[j];
  }
  free(w);
}
```

```
void merge(int a[],int b[],int c[],int m,int n){
  int i=0,j=0,k=0;
  while (i < m && j < n)
    if (a[i] < b[j])
      c[k++] =a[i++];
    else
      c[k++] =b[j++];
  while (i<m)
    c[k++] =a[i++];
  while (j<n)
    c[k++] =b[j++];
}
```