Lecture Notes

on

Merge Sort

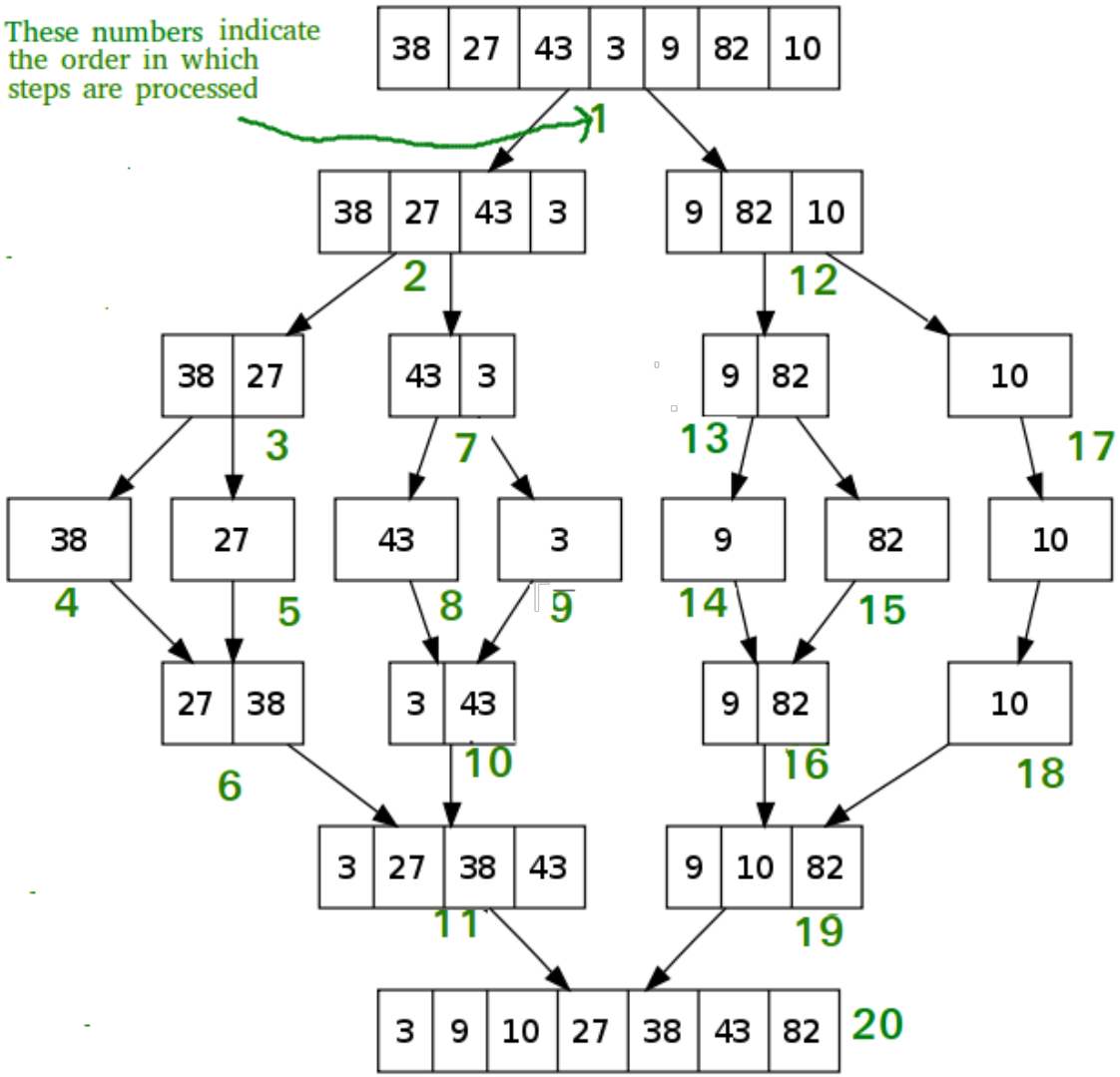
July 2020

(Be safe and stay at home)

Merge Sort

- It is based on divide and conquer approach.
- It contain nice property that in the worst case it's time complexity is $O(nlogn)$.
- If there is n elements array $a[1], ..., a[n]$, then merge sort split the array into two sets of size $a[1]....a[\lfloor \frac{n}{2} \rfloor]$ & $a[\lfloor \frac{n}{2} \rfloor + 1]...a[n]$.
- Each set is individually sorted & the resulting sorted sequences are merged to produce a single sorted sequence of n elements.

Running example:



Drawbacks:

- Recursive Calls Result In Additional Overhead Making It Unsuitable For Small Number of input.
- Sorting Is Done In Place Requiring The Client To Keep A Copy Of The Original Elements. Requires Additional Memory To Sort The Elements.

Algorithm:

```
Merge-sort(a[],low,high)
{
    if(low==high)
     return(a);
    else
    {
      mid=(low+high/2);
     Merge-sort(a,low,mid);
     Merge-sort(a,mid+1,high);
     Merge(a,low,mid,high);
     return(a);
    }
}
```

```
 merge(a,low,mid,high)
{
       h=low; i=low; j=mid+1;
      while((h<=mid) && (j<=high)) do
      //until 1st half is not over & 2nd half also not
over, comparing first element of 1st half to the 1st
element of the second half.
      {
         if(a[h]<=a[j])then
           {
           b[i]=a[h];
           h=h+1;
           }
        else
          {
           b[i]=a[j];
           j=j+1;
          }
        i=i+1;
      }
if(h>mid) then
      for(k=j to high) do
       {
          b[i]=a[k];
// copy 2nd half remaining elements to B, b[i].
          i=i+1;
       }
else
      for(k=h to mid) do
       {
          b[i]=a[k];
// copy 1st half remaining elements to B, b[i].
          i=i+1;
       }
      for(k=low to high) do
       a[k]=b[k];
}
```

Analysis

Recurrence relation for divide & conquer:

$$\text{T(n)}= \begin{cases} O(1); if n = 1 \\ 2T(\frac{n}{2}) + n; otherwise \end{cases}$$

- after applying master's method
- $T(n) = \theta(nlog_2n)$
- if any array is given then two sorted sub-array is merged. so same procedure in all cases.
- **best-case=worst-case**
- $T(n) = \theta(nlog_2n)$

Q & A?

Queries are welcome on slack channel for discussion