MAX-MIN Problem

A Divide and Conquer Approach

Finding the Maximum and Minimum

- The problem is to find the maximum and minimum items in a set of n elements
- In analyzing the time complexity we concentrate on the no of element comparisons
- The frequency count for other operations is of the same order as that for element comparisons

Straight Forward Max-Min

```
Algorithm  straightmaxmin(a,n,max,min)
{
max=min=a[1];
For i=2 to n do
{
If  (a[i]>max) then max=a[i];
If (a[i]<min) then min=a[i];
}
}
```

2(n-1) Comparison

OR

An Improvement…

```
If  (a[i]>max) then max=a[i];
 else If (a[i]<min) then min=a[i];
```

Best Case :(n-1) Comparison
Worst Case: 2(n-1) Comparison

Complexity Analysis

- The best case occurs when the elements are in increasing order – no of comparisons –n-1
- The worst case occurs when the elements are in decreasing order – no of comparisons – 2(n-1)
- The average case and the average no of comparisons is

  3n/2 - 1

10	20	30	40	50

Min=Max=10

```
If  (a[i]>max) then max=a[i];
 else If (a[i]<min) then min=a[i];
```

50	40	30	20	10

Min=Max=50

Divide And Conquer Algorithm For max-min

- If the list has more than 2 elements , P has to be divided into smaller instances
- We might divide P into two instances
  - P1=([n/2],a[1],…………a[n/2])
  - P2=([n-[n/2],a[n/2+1],………a[n])
- We can solve the sub problems by recursively invoking the same divide and conquer algorithm
- If MAX(P) and MIN(P) are the maximum and minimum of the elements in P, then MAX(P) is the larger of MAX(P1) and MAX(P2) and MIN(P) is the smaller of MIN(P1) and MIN(P2)

Recursive Algorithm For Max-Min

Algorithm

```
MaxMin(i,j,max,min)
{
If(i=j) then max=min=a[i]
   else if (i=j-1) then
   {
    if  (a[i]< a[j])  then
          {
             max=a[j];min=a[i];
          }
          else
          {
                max=a[i];
                min=a[j];
          }
else
   {
   mid=(i+j)/2;
   maxmin(i,mid,max,min);
   maxmin(mid+1,j,max1,min1)
   ;
   if (max< max1) then
   max=max1;
   if(min > min1)  then
   min=min1;
   }
}
```

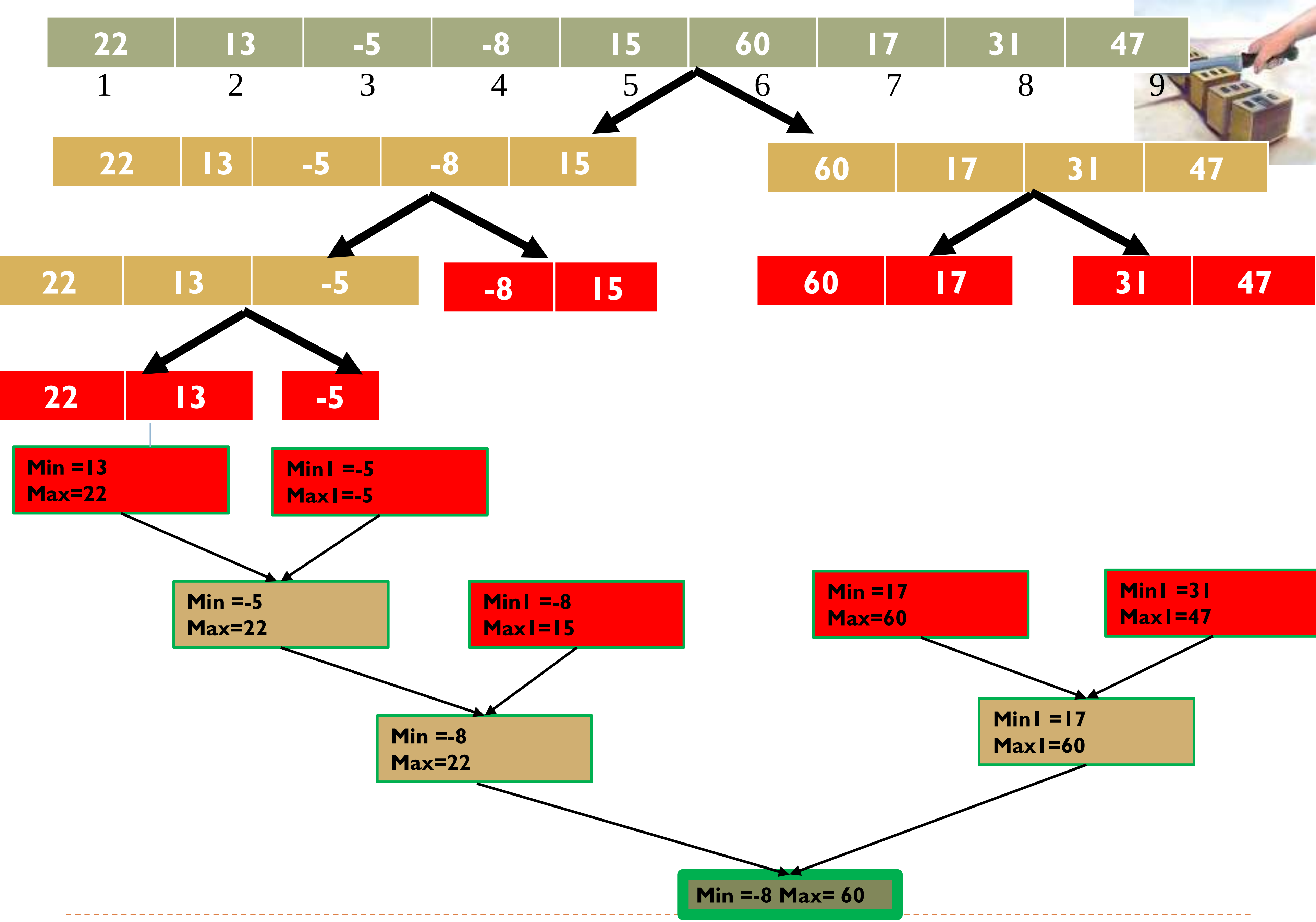
Ahis procedure is initially invoked by the statement Maxmin(1,n,x,y)

Example

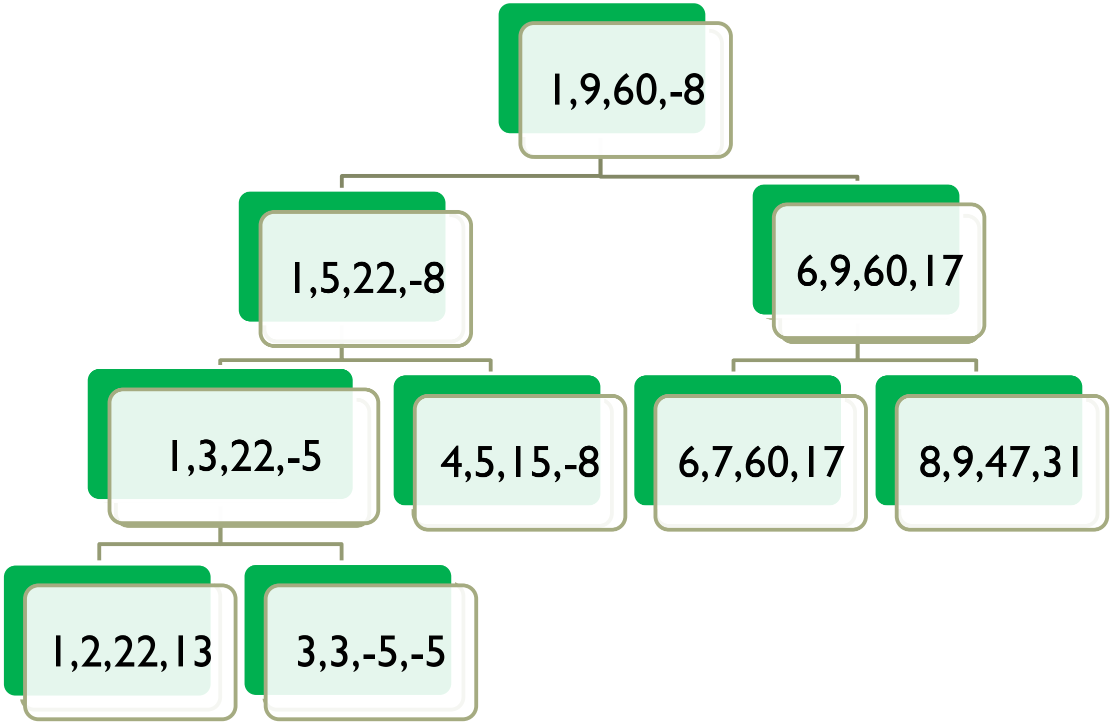
Consider the array of numbers

22 13 -5 -8 15 60 17 31 47

For this algorithm each node has four items of information i, j, max and min


22 13 -5 -8 15 60 17 31 47
1 2 3 4 5 6 7 8 9
22 13 -5 -8 15
60 17 31 47
22 13 -5
-8 15
60 17
31 47
22 13
-5
Min =13
Max=22
Min1 =-5
Max1=-5
Min =-5
Max=22
Min1 =-8
Max1=15
Min =17
Max=60
Min1 =31
Max1=47
Min =-8
Max=22
Min1 =17
Max1=60
Min =-8 Max= 60

22	13	-5	-8	15	60	17	31	47
1	2	3	4	5	6	7	8	9




```
MaxMin(i,j,max,min)
{
        If(i=j) then max=min=a[i]
        else if (i=j-1) then
        {
         if  (a[i]< a[j])  then
        {
                        max=a[j];min=a[i];
        }
        else
        {
                      max=a[i];
                      min=a[j];
        }

         else
         {
                      mid=(i+j)/2;
                      maxmin(i,mid,max,min);
                      maxmin(mid+1,j,max1,min1);
                      if (max< max1) then max=max1;
                      if(min > min1)  then  min=min1;
         }
}
```

Complexity Analysis

- **T(n)=T(n/2)+T(n/2)+2,   n>2**
  1 ,   n=2
  0 ,   n=1

$T(n) = 2T(n/2) + 2$

$= 2[2T(n/4)+2] + 2$

$= 4T(n/4)+4 + 2$

$= 4[2T(n/8)+2]+4 + 2$

$= 8T(n/8)+8+4 +2$

$= 2^3T(n/2^3)+2^3+2^2 + 2$

$= 2^kT(n/2^k)+ 2^k+ 2^{k-1}+\ldots+2$

for T(2), $n/2^k=2$ **=>** $n= 2.2^k=2^{k+1}$

$= 2^kT(2)+ 2^k+ 2^{k-1}+\ldots+2$

$= 2^k+ 2^k+ 2^{k-1}+\ldots+2= 2^{k+1}+ 2^{k-1}+\ldots+2$ $2^k+2^{k-1}+\ldots+2 + 1= 2^{k+1}-1$

$= n + 2^{k-1+1}-1 -1= n + n/2 -2$ $\mathbf{=3n/2-2=O(n)}$

Example Recurrence Solutions

- Examples
  - $T(n) = T(n-1) + k \quad \Rightarrow \quad O(n)$
  - $T(n) = T(n-1) + n \quad \Rightarrow \quad O(n^2)$
  - $T(n) = T(n/2) + k \quad \Rightarrow \quad O(\log(n))$
  - $T(n) = 2 \times T(n/2) + k \quad \Rightarrow \quad O(n)$
  - $T(n) = 2 \times T(n/2) + n \quad \Rightarrow \quad O(n \log(n))$
  - $T(n) = 2 \times T(n-1) + k \quad \Rightarrow \quad O(2^n)$


Thank you!