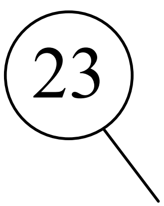
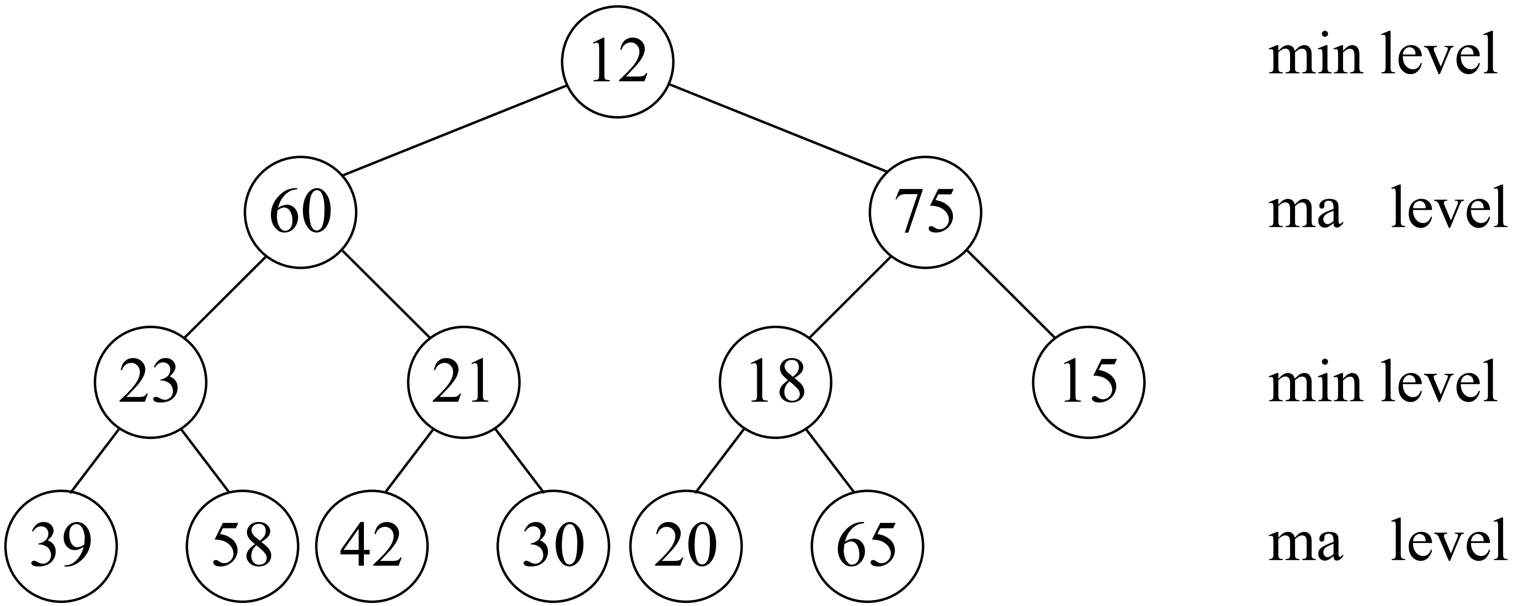
Advanced Data Structures

Min-Max Heap

Min-Max Heap

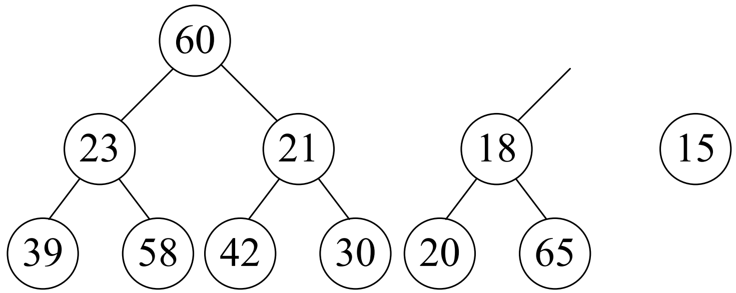


- I e 10 80



12
60
75
23
21
18
15
39
58
42
30
20
65
min level
ma level
min level
ma level

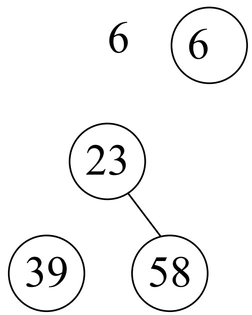
1.



2.

3.

4.



5. Ne de i i 9 na le el a d i la ge ha i a e 9 e cha ge

6.

```
template <class T  pe>
void MinMa  Heap<T  pe>::In  er (con    Elemen <T  pe>&    )
   if (n == Ma  Si  e )    MinMa  F  ll( ); return;
   n++;
   int p = n/2;
   if (!p)   h[1] =    ; return;
   s   itch (le  el(p))
     case MIN:
       if (  .ke    < h[p].ke  )
           h[n]=h[p];
          Verif  Min(p,   );
```

```
    else Verif  Ma  (n,   );
     break;
  case MAX:
    if (  .ke   > h[p].ke  )
       h[n]=h[p];
      Verif  Ma  (p,   );

    else Verif  Min(n,   );
```

- Level:

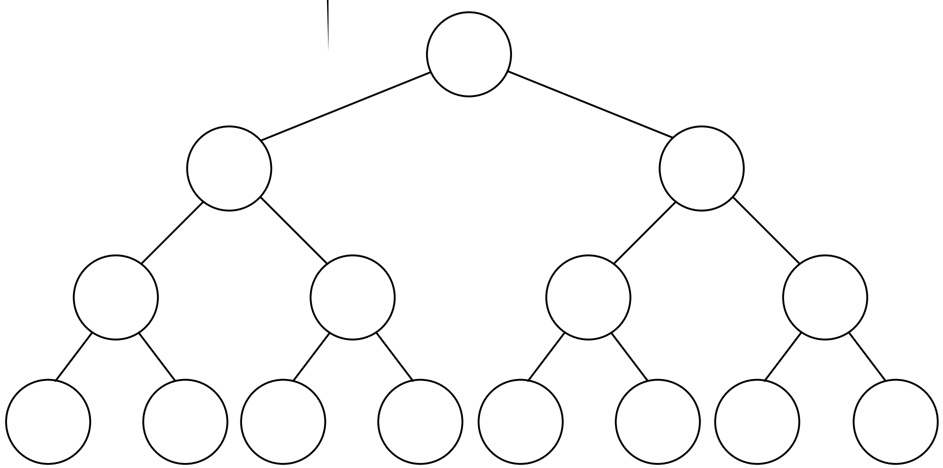
  Ma Level:

  $\lceil \log_2(j + 1) \rceil$ is even

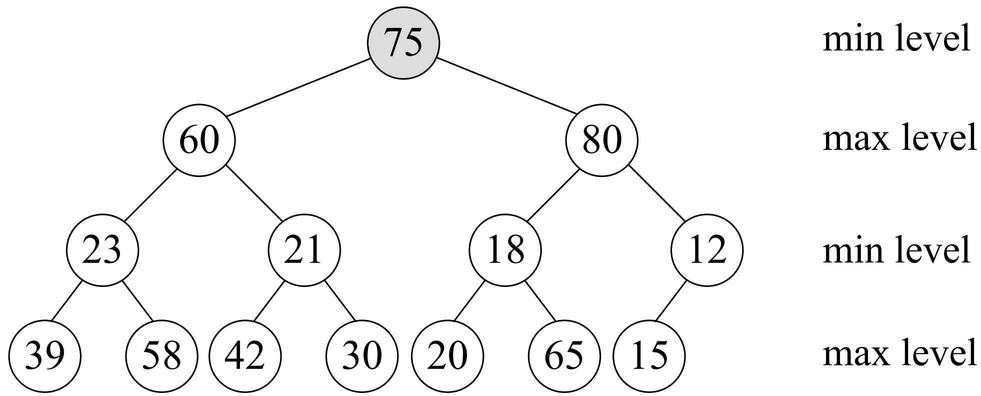
  Min Level:

  $\lceil \log_2(j + 1) \rceil$ is odd
- Verif Ma
- Verif Min

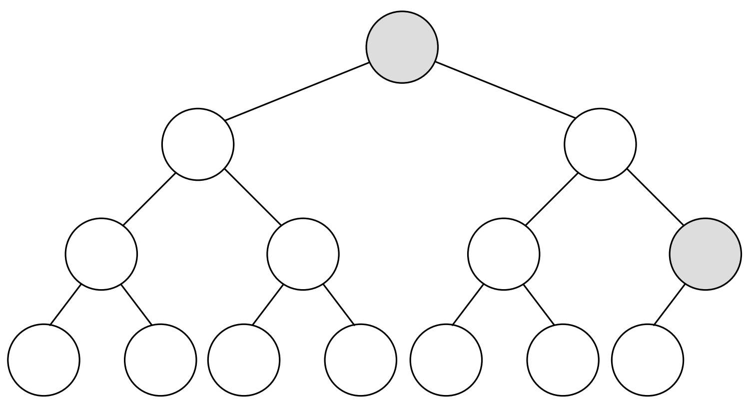
- Dele e Mi f m Mi 9-Ma Hea



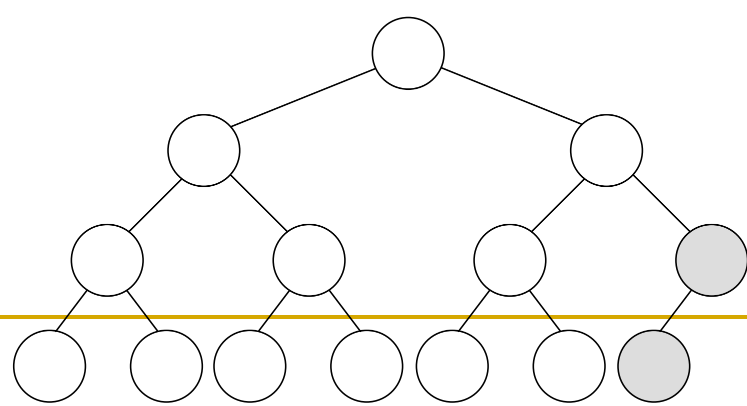
1.


75
60
80
23
21
18
12
39
58
42
30
20
65
15
min level
max level
min level
max level

2.



3.



I e a e de i a i – a hea i h

1 e hea : i he e ;

2 a lea e child f :

fi d de k i h he alle da a;

(a) .ke h[k].ke

i he e ;

(b) .ke > h[k].ke a d k i a child f
:
e lace k i h ;
k i he e .

(c) .ke > h[k].ke a d k i a g a dchild f :

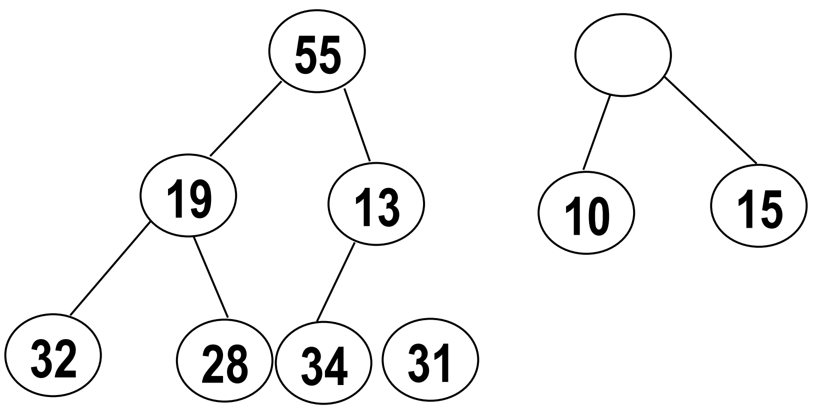
k i he e

if .ke > h[ ].ke e cha ge a d h[ ]

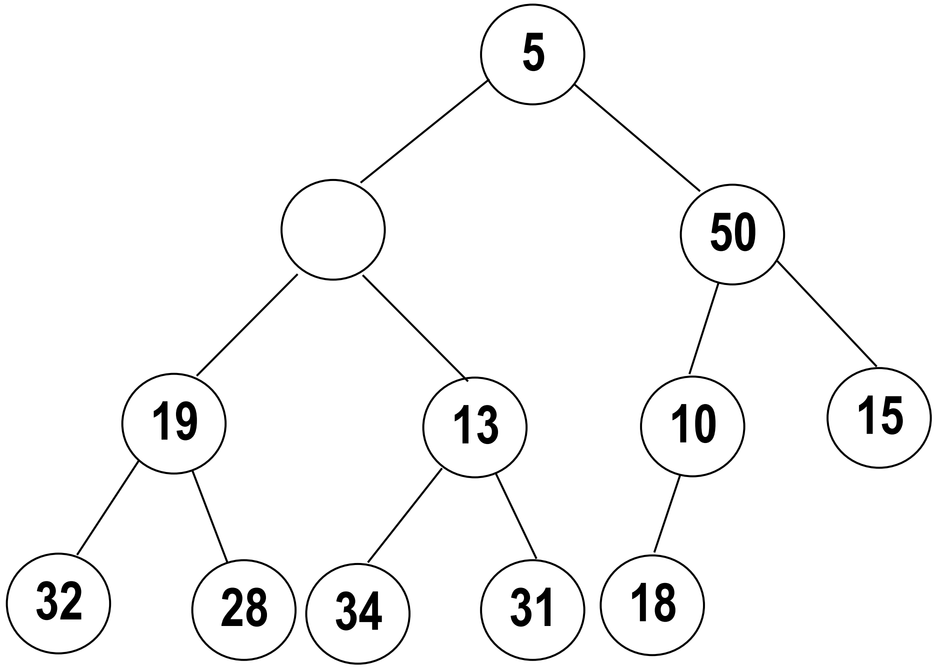
i e i (b) i _ a hea ed i h k

```
template <class T pe>
Elemen <T pe>* MinMa  Heap<T pe>::
                                    Dele eMin(Elemen <T pe>&  )
    if (!n)   MinMa  Emp   ( ); return 0;
       = h[1];
     Elemen <T pe>    = h[n--];
    int i = 1, j = n/2;
       hile (i <= j)        //  ha   children
       int k = MinChildGrandChild(i);
       if (  .ke   <= h[k].ke  )
          break;            //  2(a)
```

Min-max heap



Min-max heap (con.t)



- Dele e 55

Advanced Data Structures

Deap

A deap is a complete binar tree：

（1） no data in root;

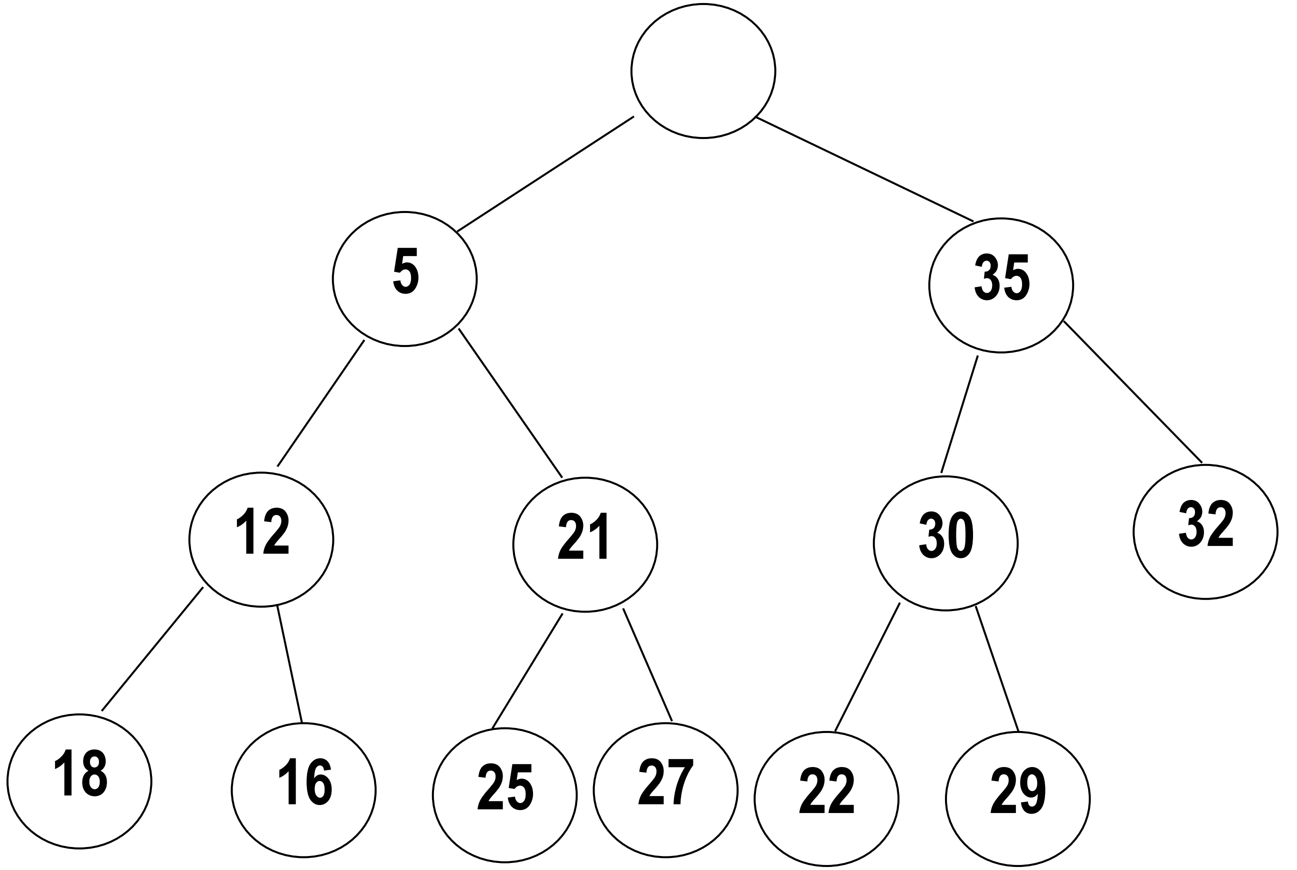
（2） left subtree is a min heap

（3） right subtree is a ma heap

（4） ever node in left subtree is no bigger than its corresponding node in right subtree

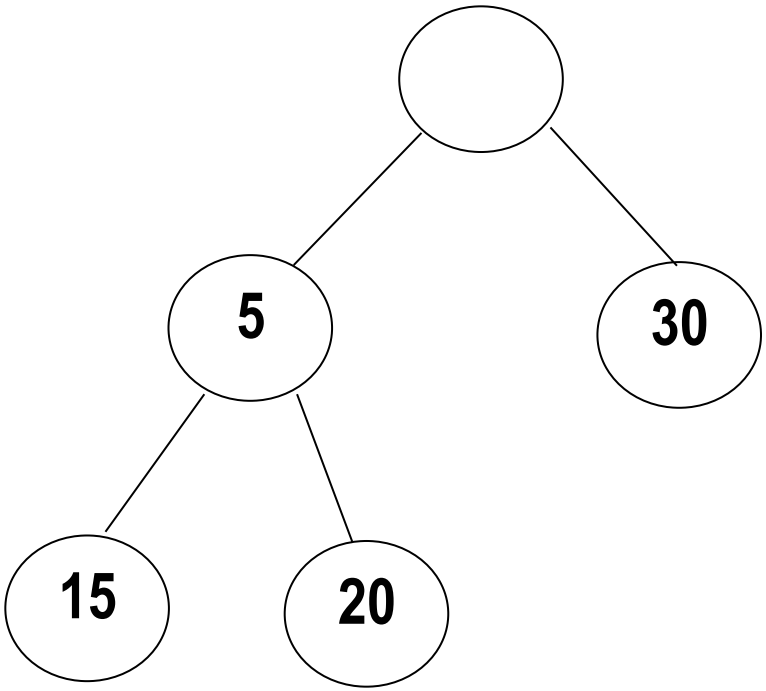
corresponding nodes

Deap



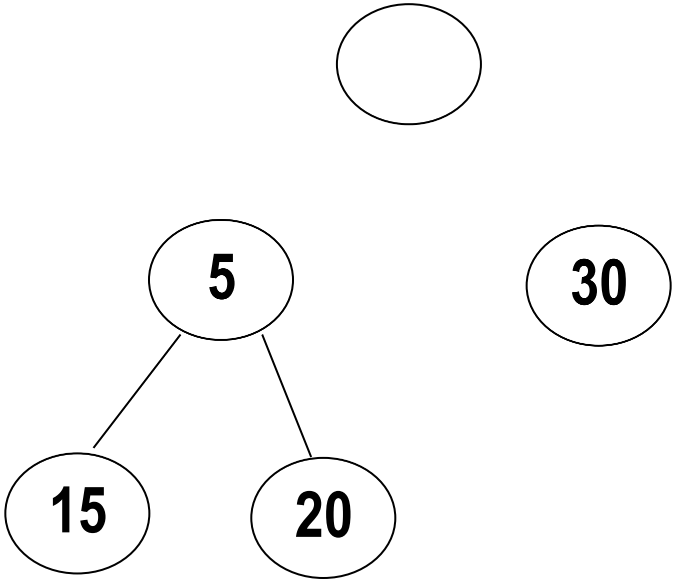
Deap (con.t)

- I e i
  - Add e de he e d f ee, Heapif

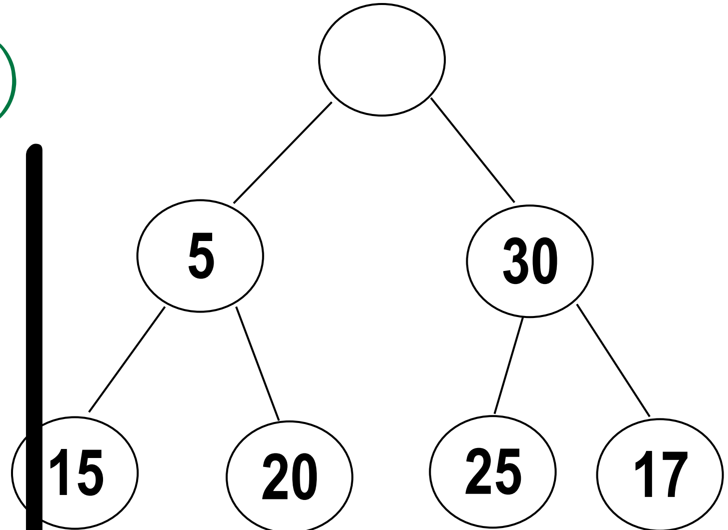


  - I e 25

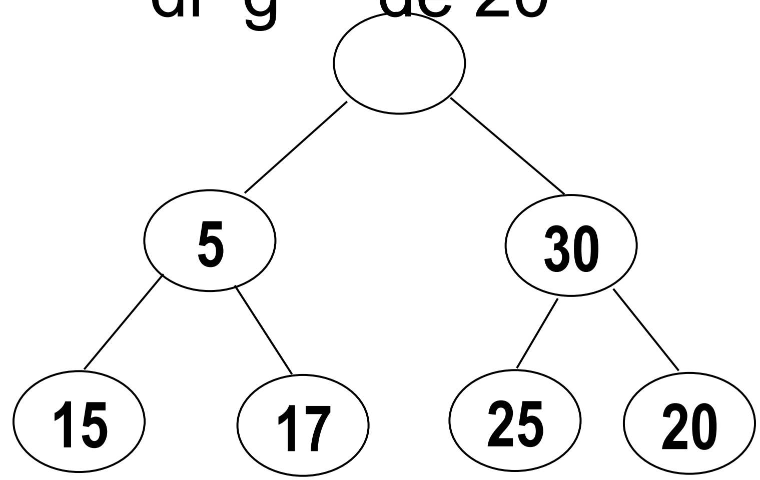
Deap (con.t)



Deap (con.t)

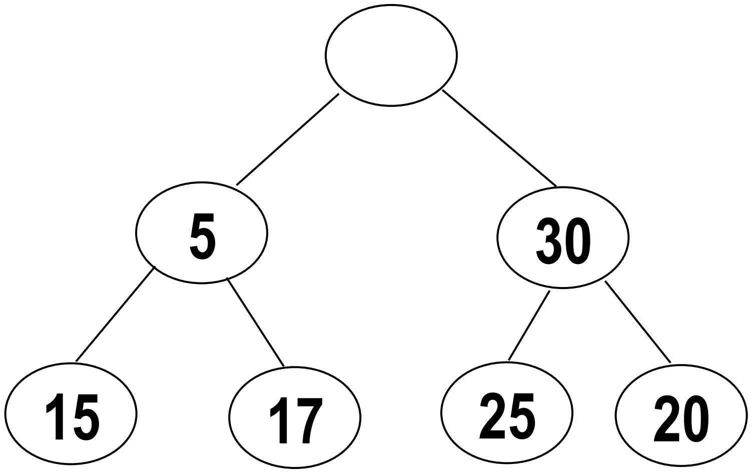


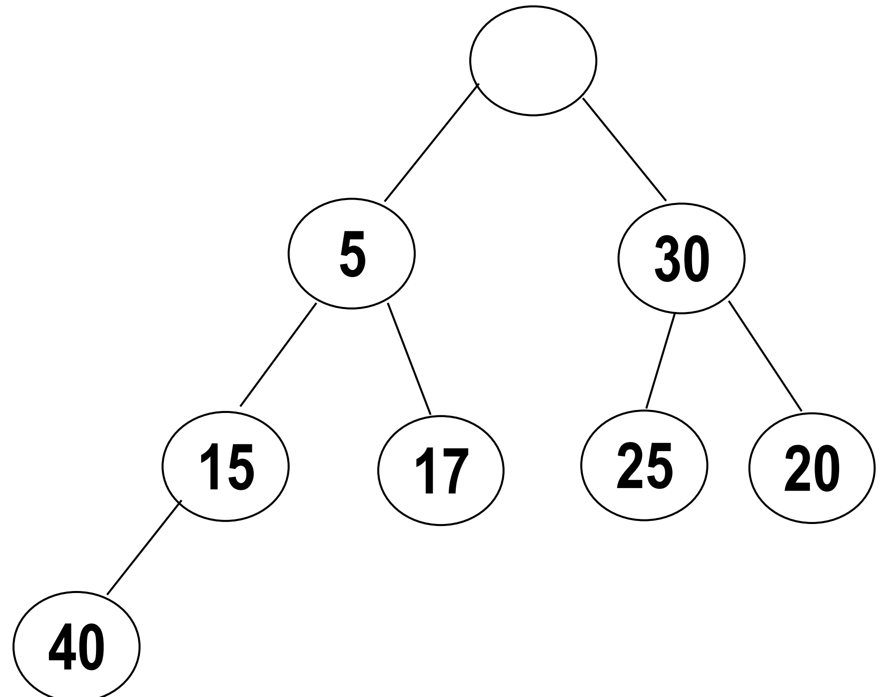
- 17 i malle ha i c e di g de 20
  - E cha ge 17 a d 20
- Check he mi -hea



- N eed check he ma -hea , WHY

Deap (con.t)

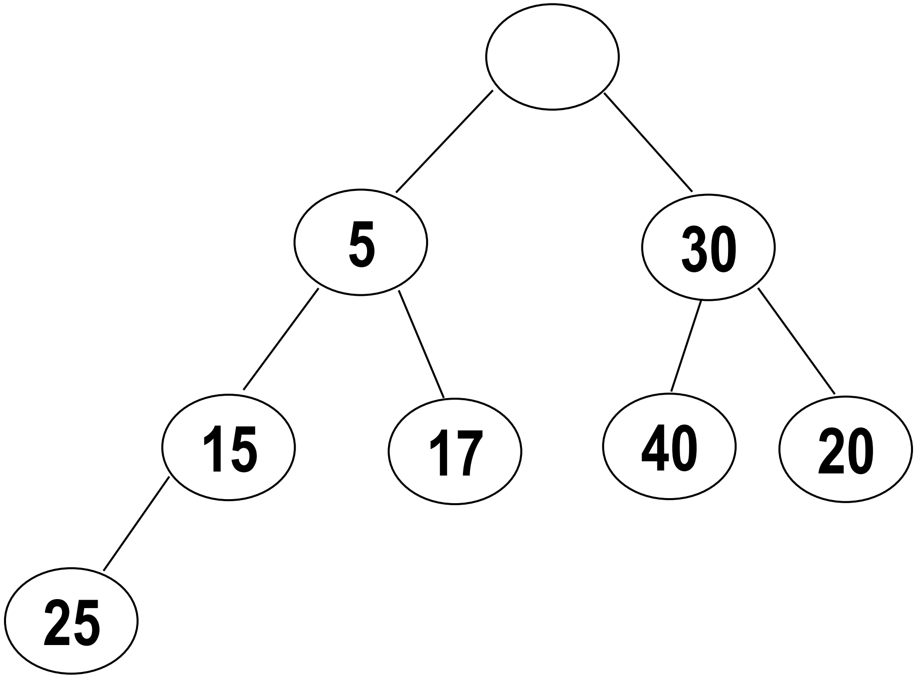


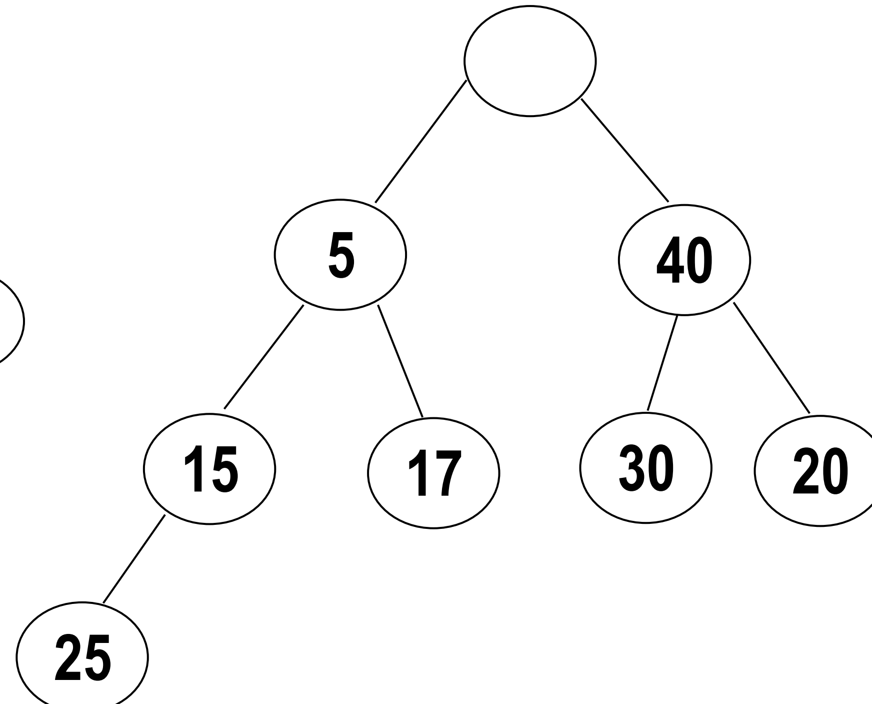


- I e 40

- 40 i bigge ha 25( )
  - 9 E cha ge 40 a d 25

Deap (con.t)





- The righ b ree i no a ma -heap
  - Heapif
- No need o adj he lef b ree, WHY ?

```
template <class T  pe>
void Deap<T  pe>::In  er  (const Elemen <T  pe>&   )

   int i;
   if (n == Ma  Si  e )    DeapF  ll( ); return;
   n++;
   if (n == 1)    d[2] =    ; return;
   int p = n + 1;
   s   itch (Ma  Heap(p))
     case TRUE:            // p in ma  -heap
            i = MinPar ner(p);
            if (  .ke   < d[i].ke  )
               d[p] = d[i];
               MinIn  er (i,   );
```

（1） Deap::Ma Heap(int p)

for $p > 1$， if $2^{\lfloor \log_2 p \rfloor} + 2^{\lfloor \log_2 p \rfloor - 1} \le p < 2^{\lceil \log_2 p \rceil}$， p is in the ma -heap

（2） Deap::MinPartner(int p)

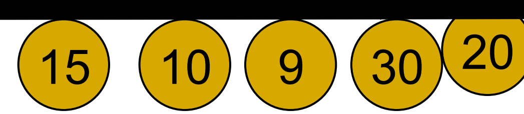
P is in ma -heap and P is the last one in Deap:

$$p \quad 2^{\lfloor \log_2 p \rfloor - 1}$$

（3） Deap::Ma Partner(int p)

P in min-heap and P is the last one in Deap:

$$(p + 2^{\lfloor \log_2 p \rfloor - 1}) / 2$$


15
10
9
30
20