

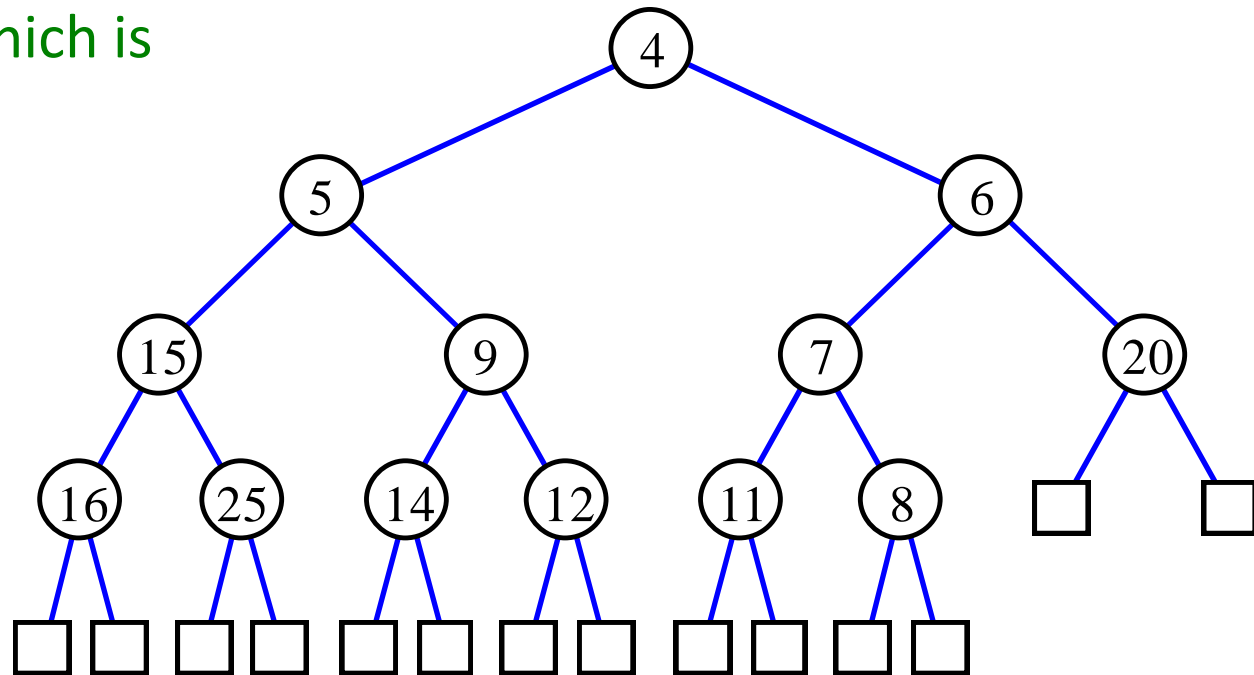
Heaps in C

CHAN Hou Pong, Ken

CSCI2100 Data Structures Tutorial 7

Review on Heaps

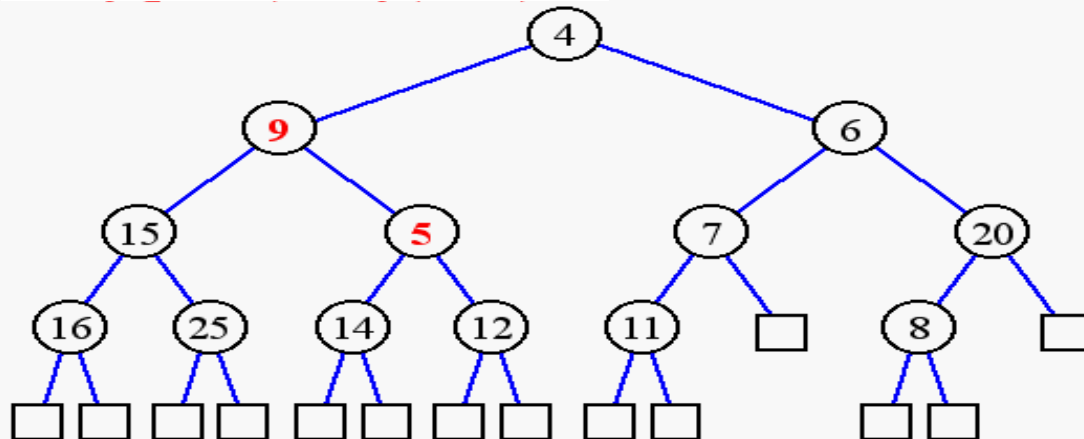
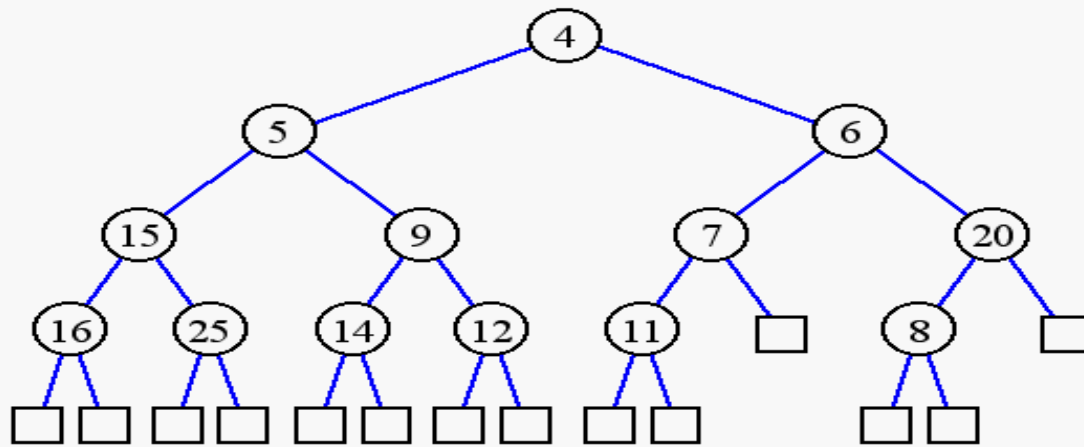
- A *heap* is implemented as a binary tree
- It satisfies two properties:
 - **MinHeap: parent \leq child**
 - **[OR MaxHeap: parent \geq child]**
 - all levels are full, except the last one, which is left-filled



What are Heaps Useful for?

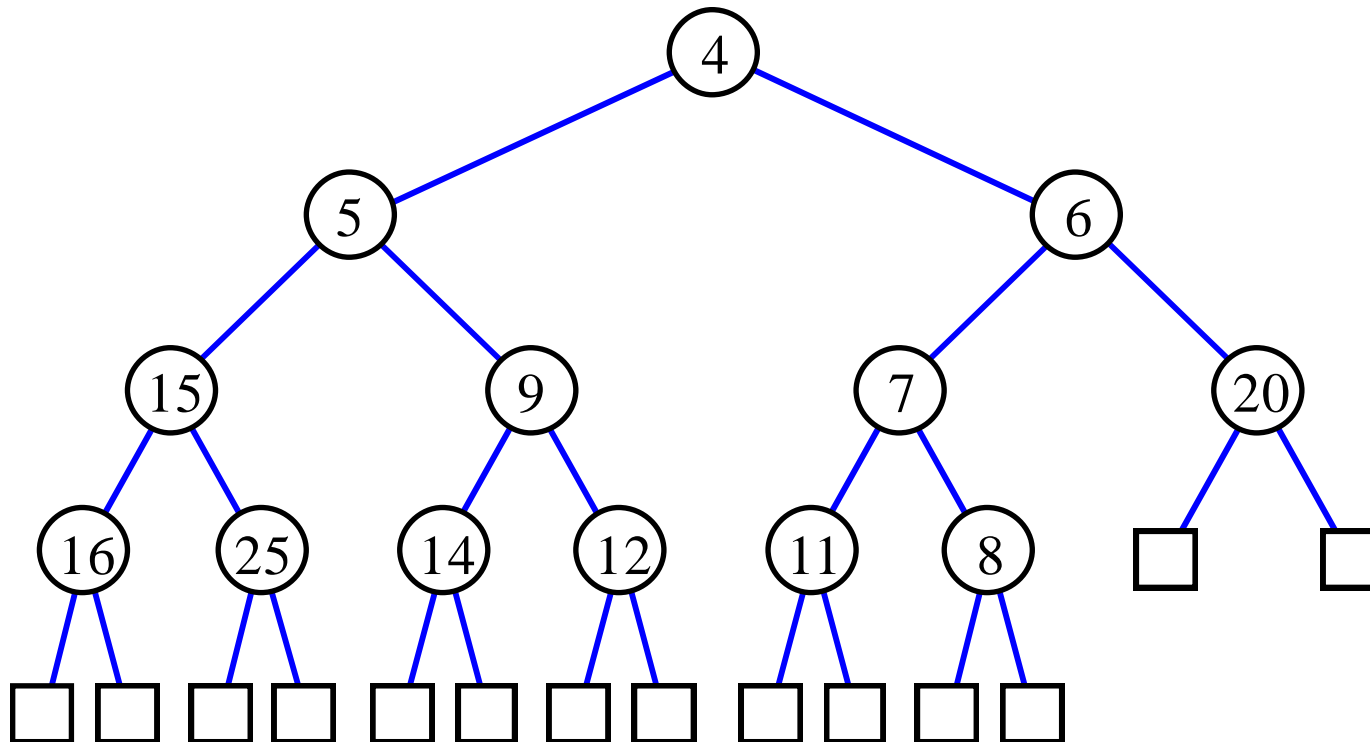
- To implement priority queues
- Priority queue = a queue where all elements have a “priority” associated with them
- **Remove** in a priority queue removes the element with the smallest priority
- Basic operations:
 - insert
 - removeMin

Heap or Not a Heap?



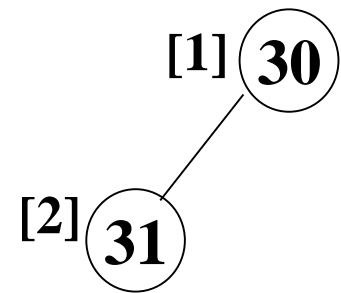
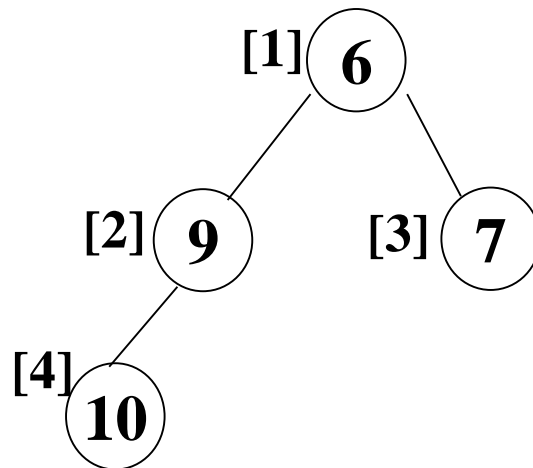
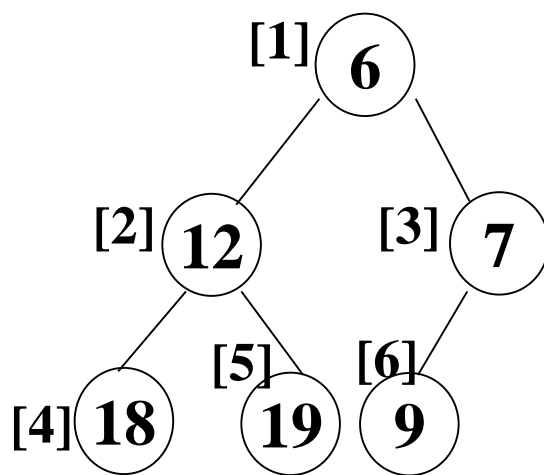
Heap Properties

- A heap T storing n keys has height $h = \lfloor \log_2 n \rfloor$,
- e.g. 13 keys, height = 3



Heap Implementation

- Using arrays
- Parent = k ; Children = $2k$, $2k+1$



Heap Structure in C

```
struct HeapStruct {  
    int capacity;  
    int size;  
    ElementType *Elements;  
};  
typedef struct HeapStruct Heap;
```

ADT for Min Heap

objects: $n \geq 0$ elements organized in a binary tree so that the value in each node is at least as large as those in its children

method:

Heap Create(MAX_SIZE)::= create an empty heap that can
hold a maximum of max_size elements

Boolean HeapFull(heap)::= if (heap->size == heap->capacity) return TRUE
else return FALSE

ADT for Min Heap (cont')

method:

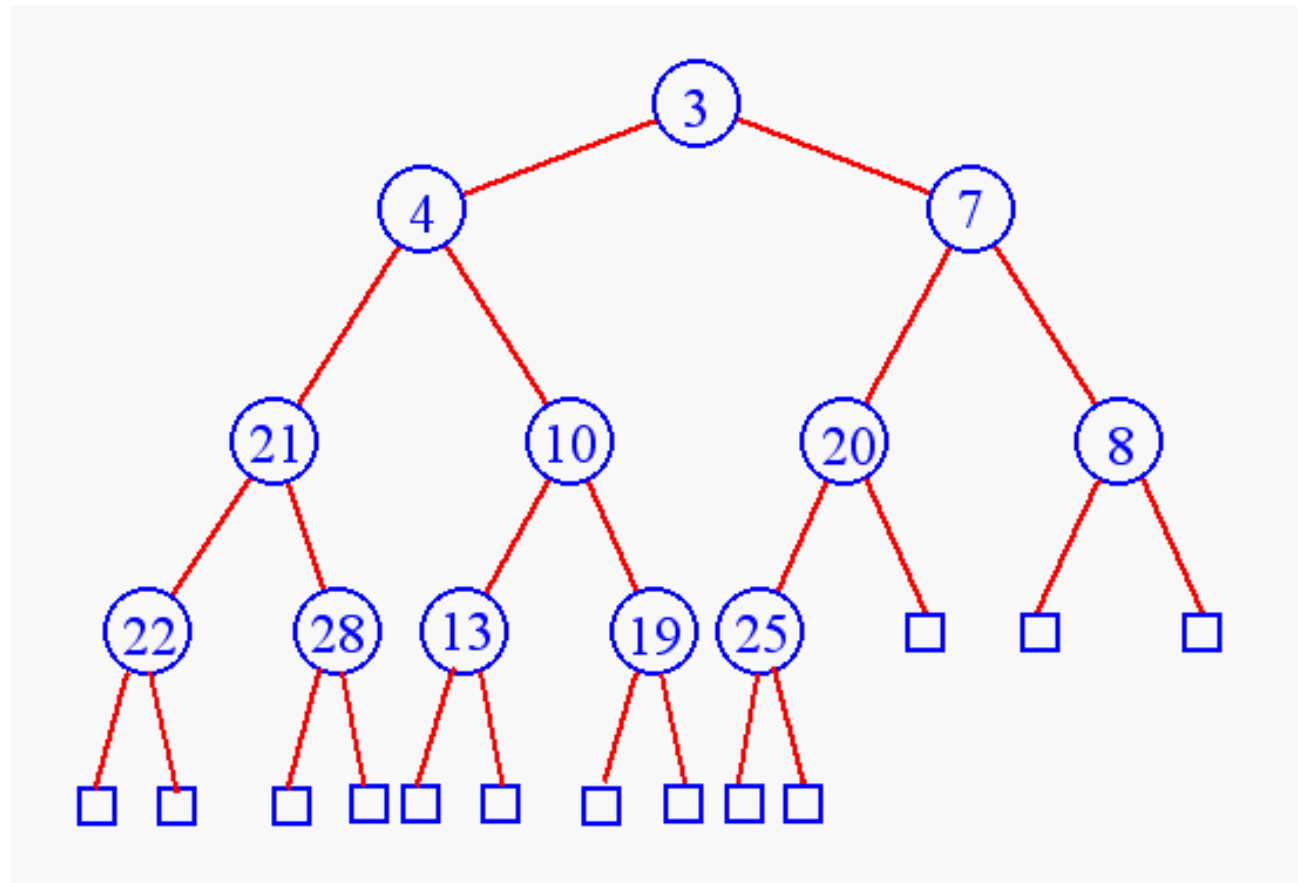
Heap Insert(heap, item)::= if (!HeapFull(heap)) insert
item into heap and return the resulting heap
else return error

Boolean HeapEmpty(heap)::= if (heap->size>0) return FALSE
else return TRUE

Element Delete(heap)::= if (!HeapEmpty(heap)) return one
instance of the **smallest** element in the heap
and remove it from the heap
else return error

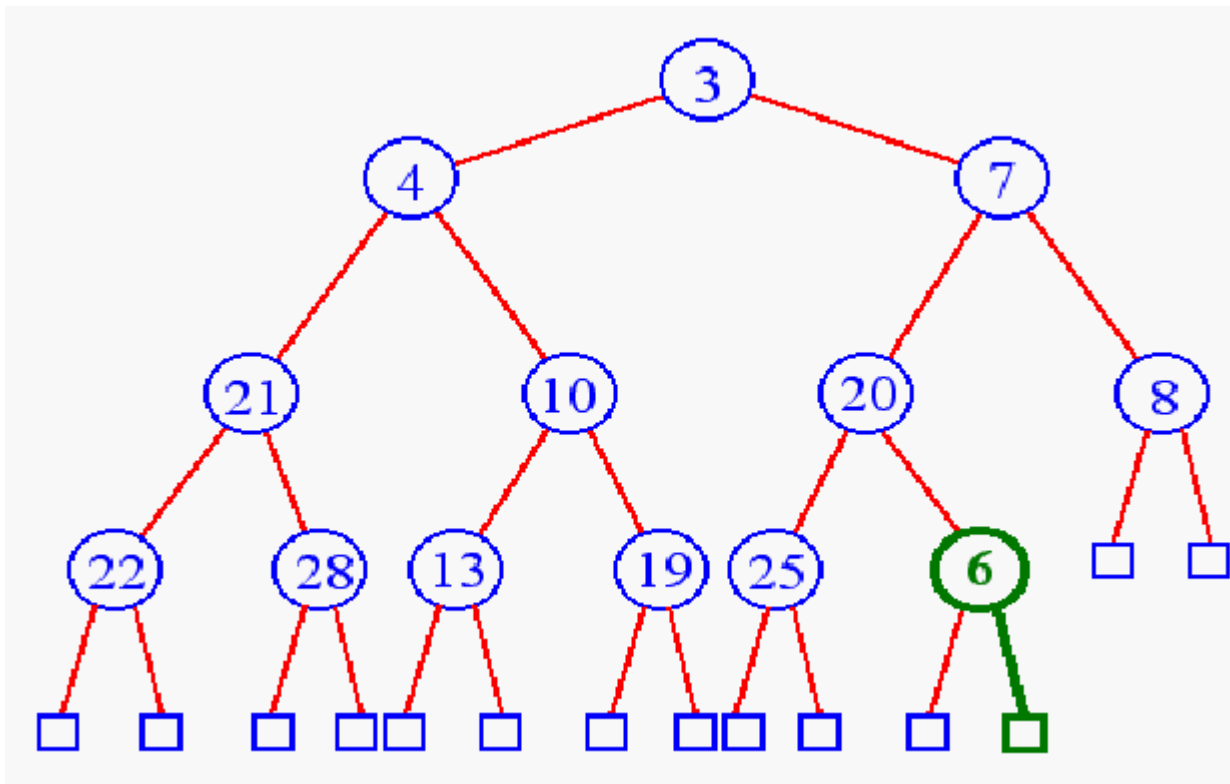
Review on Heap Insertion

- Insert 6



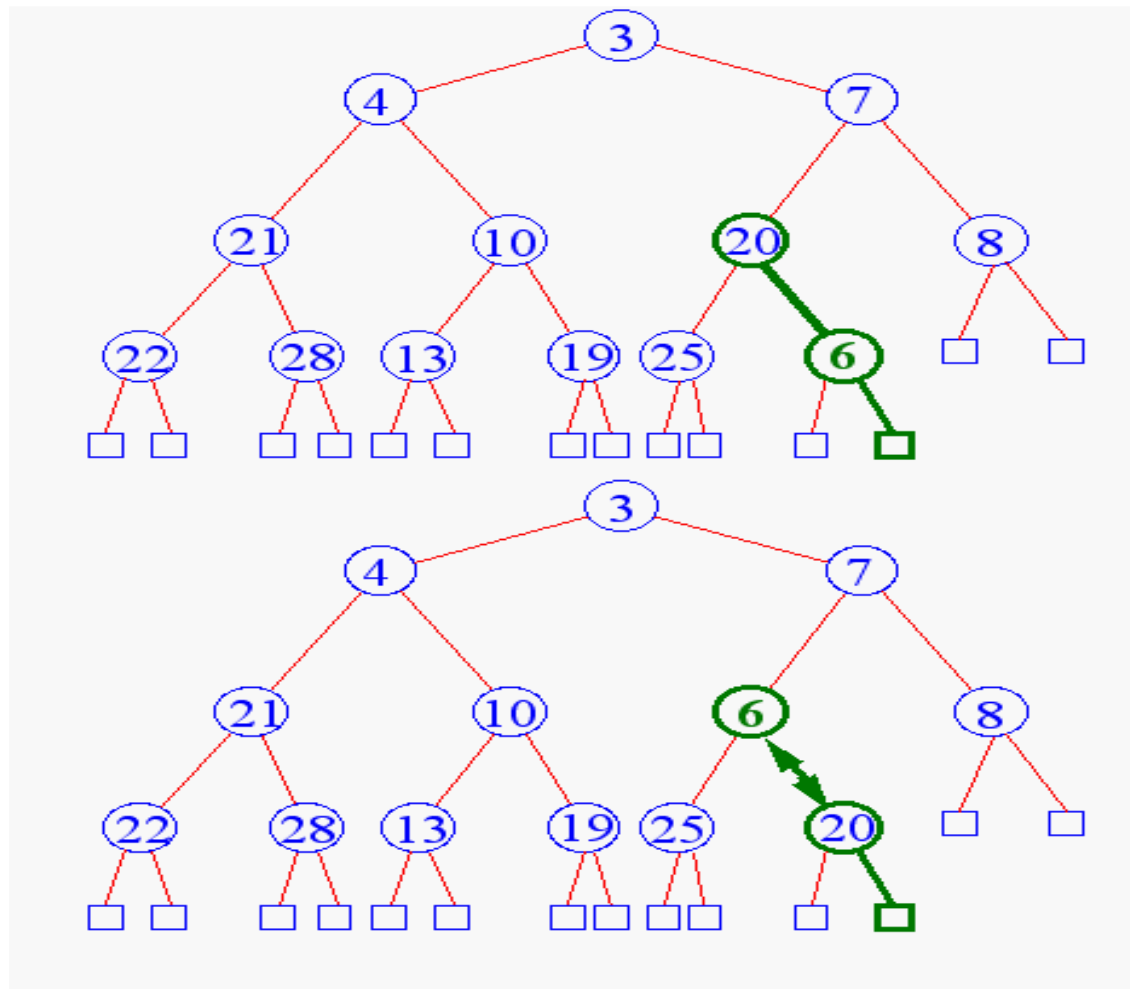
Heap Insertion

- Add key in next available position
- Violate Heap properties

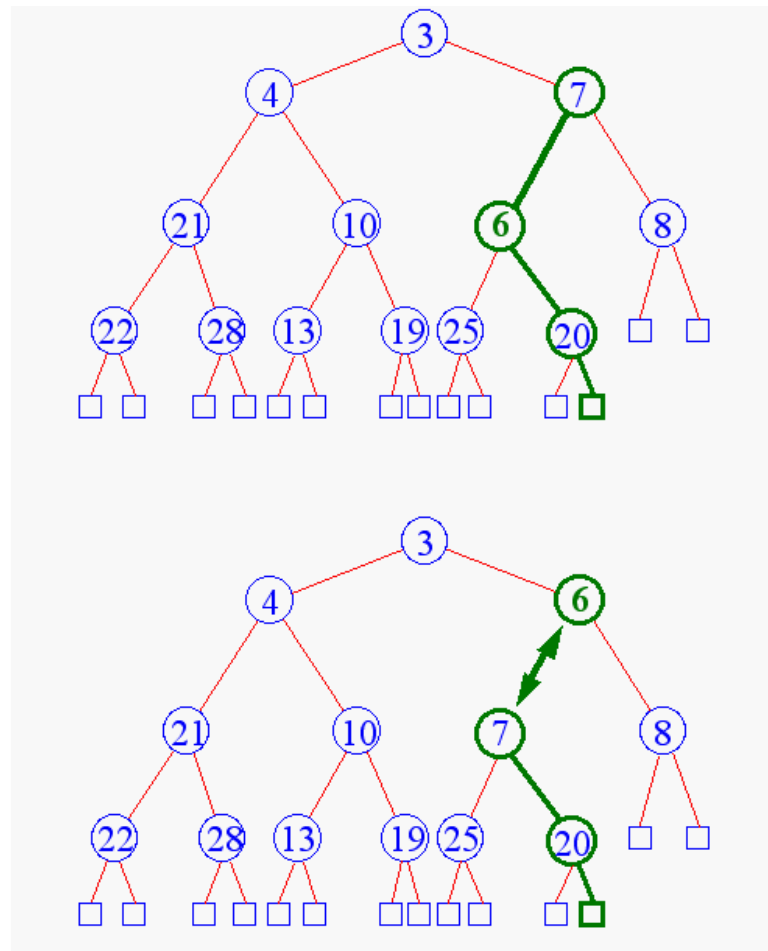


Heap Insertion

- Begin percolate up

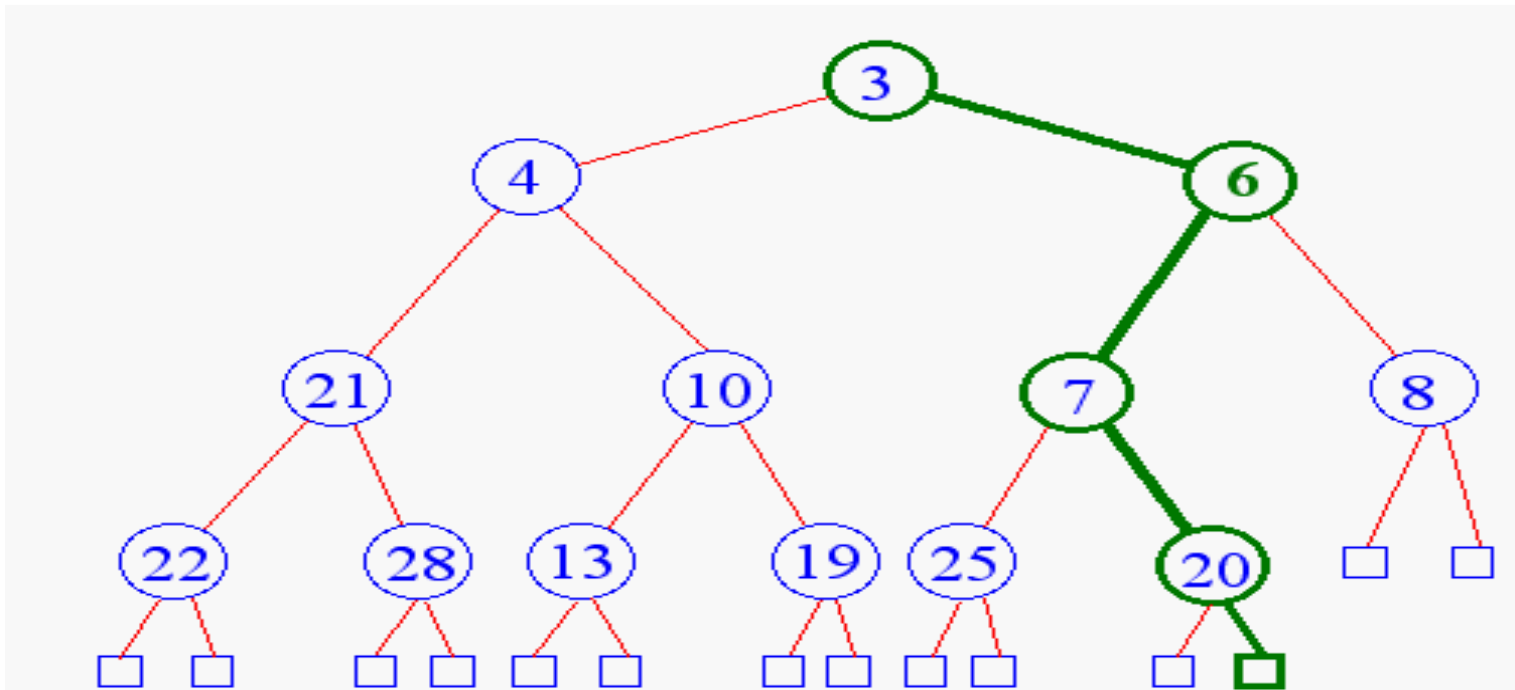


Heap Insertion



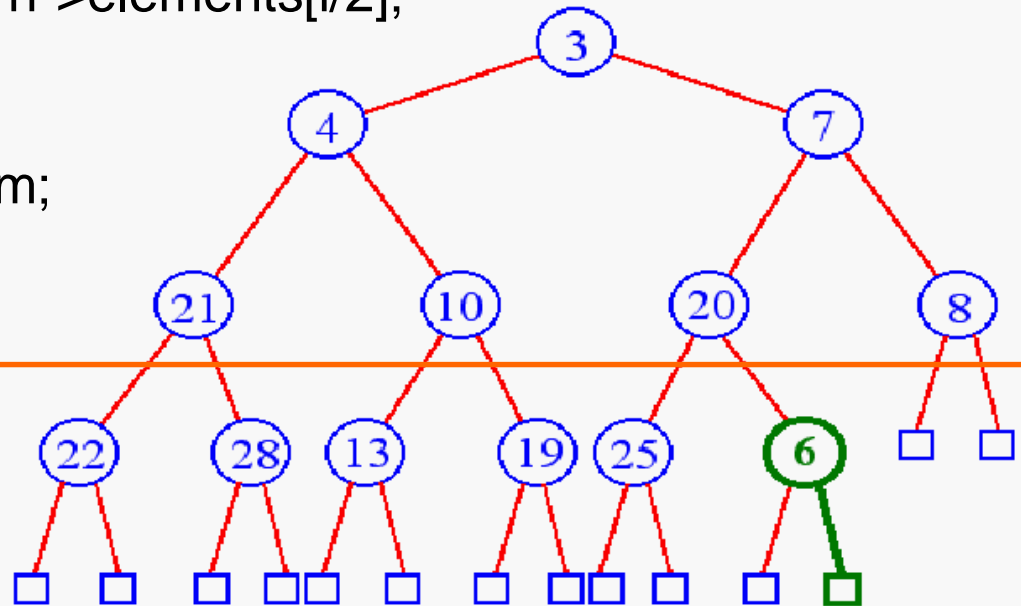
Heap Insertion

- Terminate percolate-up when
 - reach root
 - key child is greater than key parent



Insertion into a Heap $O(\log_2 n)$

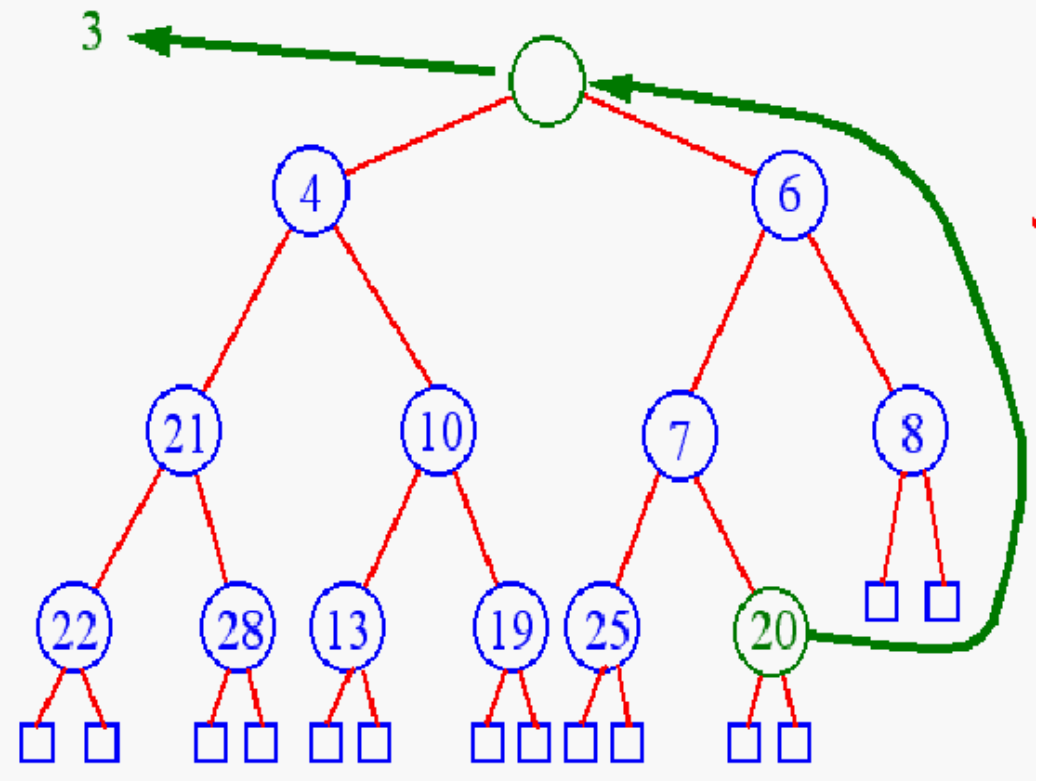
```
void insertHeap(Heap *h, ElementType item){
    int i;
    if (HEAP_FULL(h)) {
        printf("The heap is full.\n");
        exit(1);
    }
    i = ++h->size;
    while ( (i!=1) && (item < h->elements[i/2]) ){
        h->elements[i] = h->elements[i/2];
        i /= 2;
    }
    h->elements[i]=item;
}
```



Heap Removal

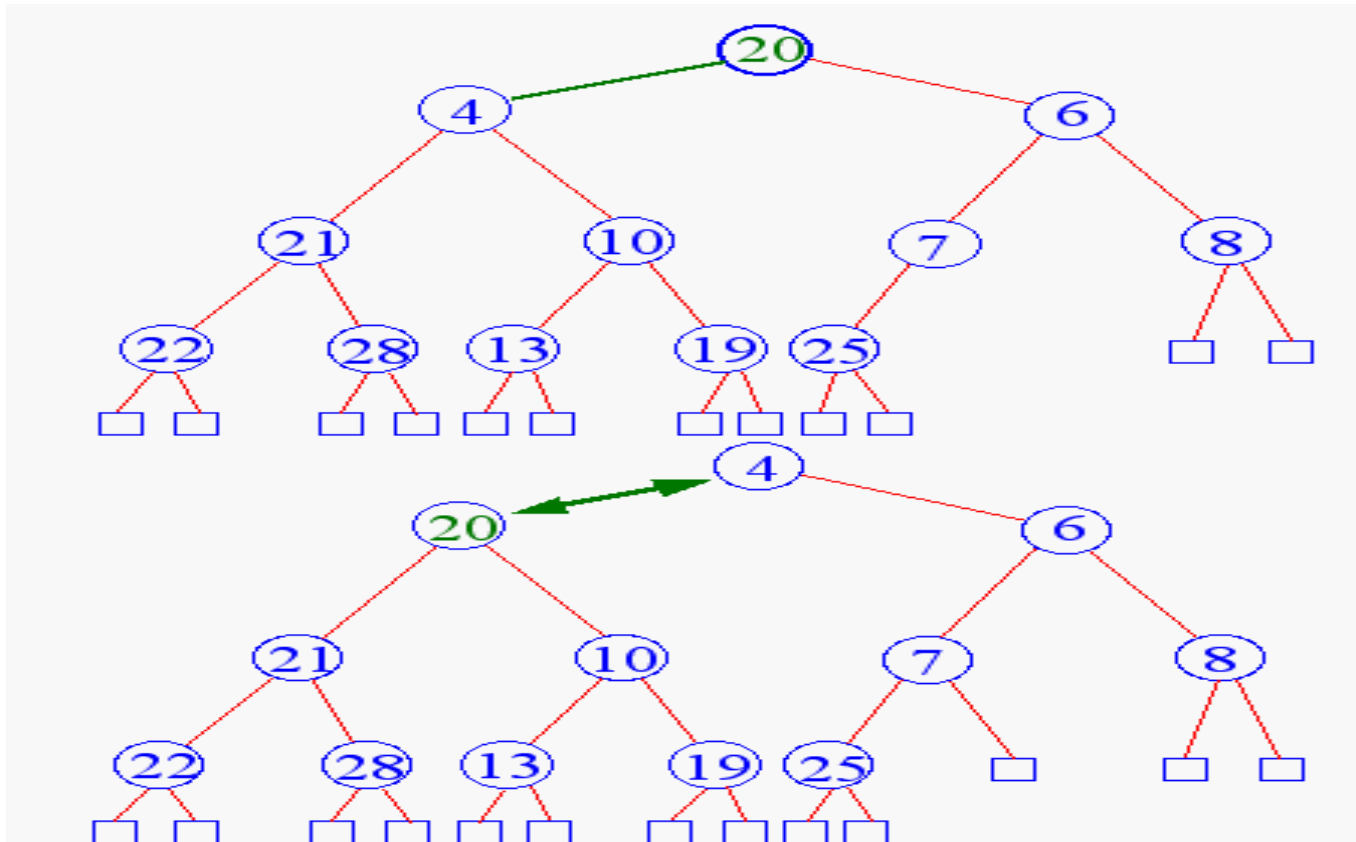
- Remove element from priority queues?

removeMin()

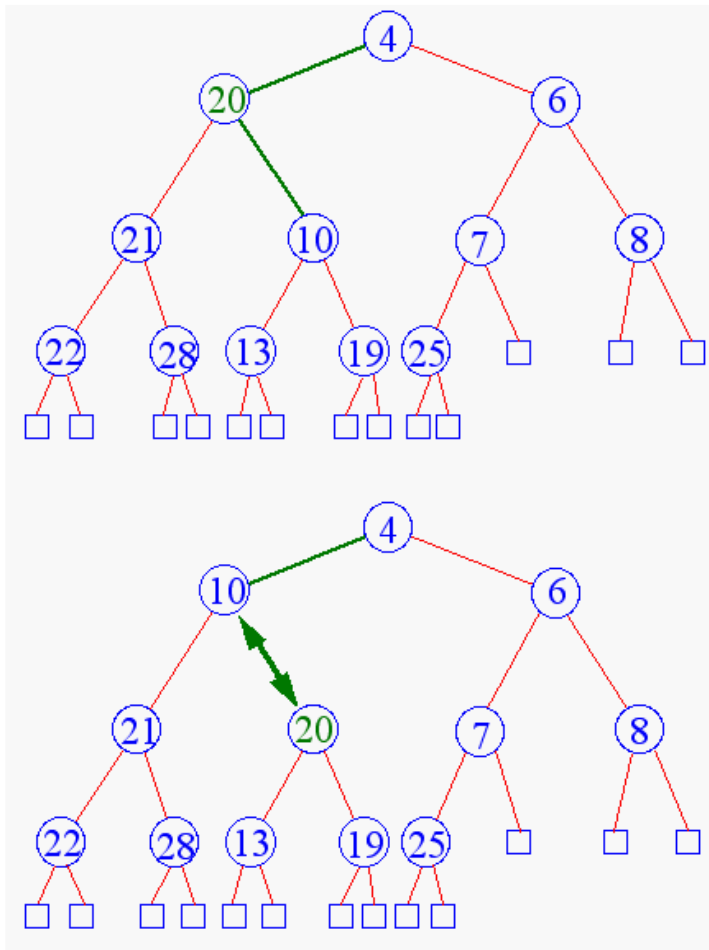


Heap Removal

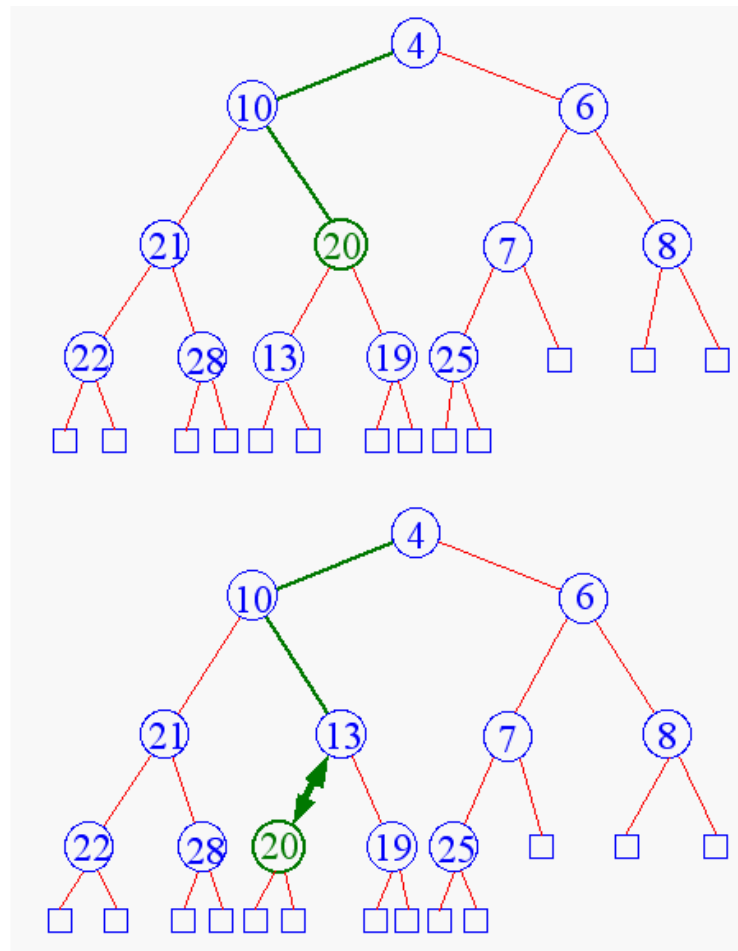
- Begin percolate down



Heap Removal

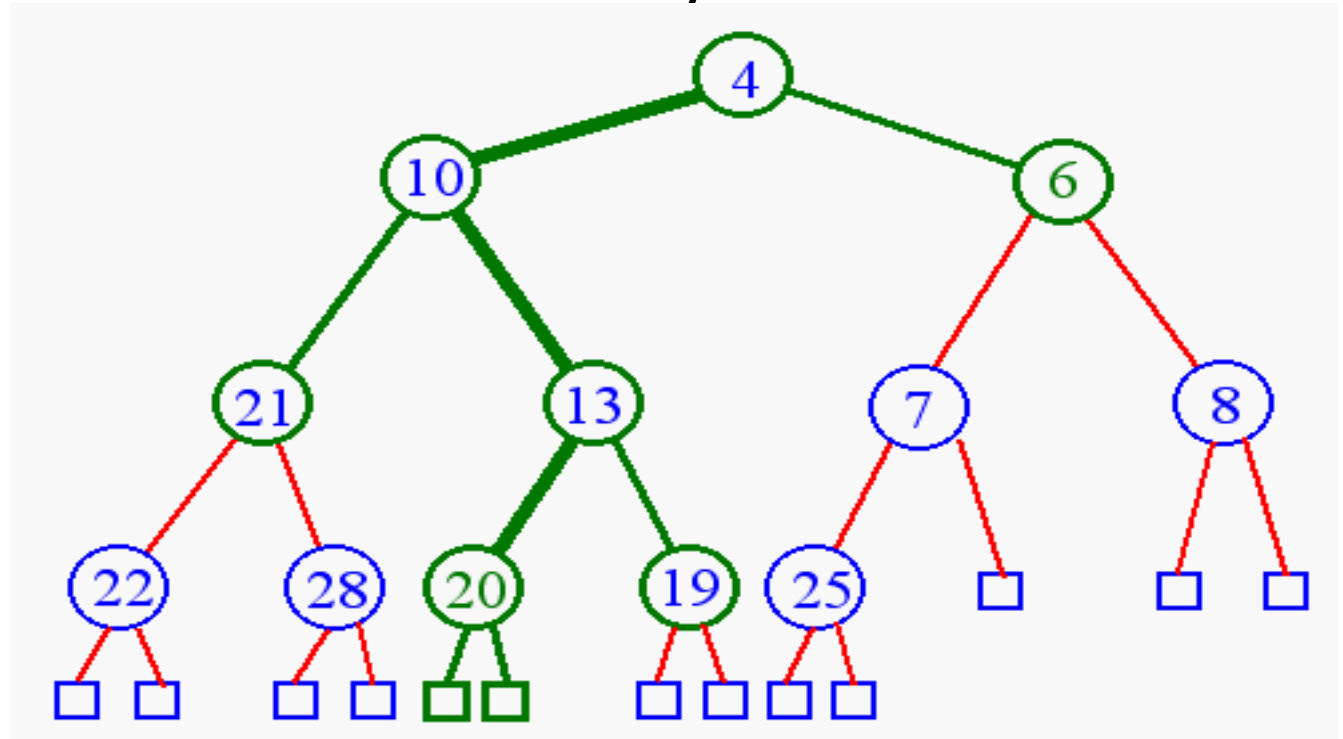


Heap Removal



Heap Removal

- Terminate percolate-down when
 - reach leaf level
 - key parent is smaller than key child



Deletion from a Heap

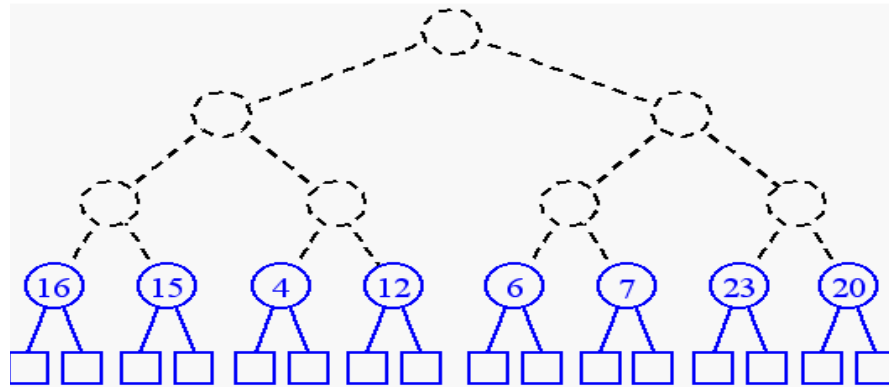
```
ElementType deleteHeap(Heap *h){
    int parent, child;
    ElementType item, temp;
    if (HEAP_EMPTY(h)){
        printf("The heap is empty\n");
        exit(1);
    }
    // save value of the minimum element
    item = h->elements[1];
    //use last element in heap to adjust heap
    temp = h->elements[h->size--];
    parent = 1;
    child = 2;
```

Deletion from a Heap (cont'd)

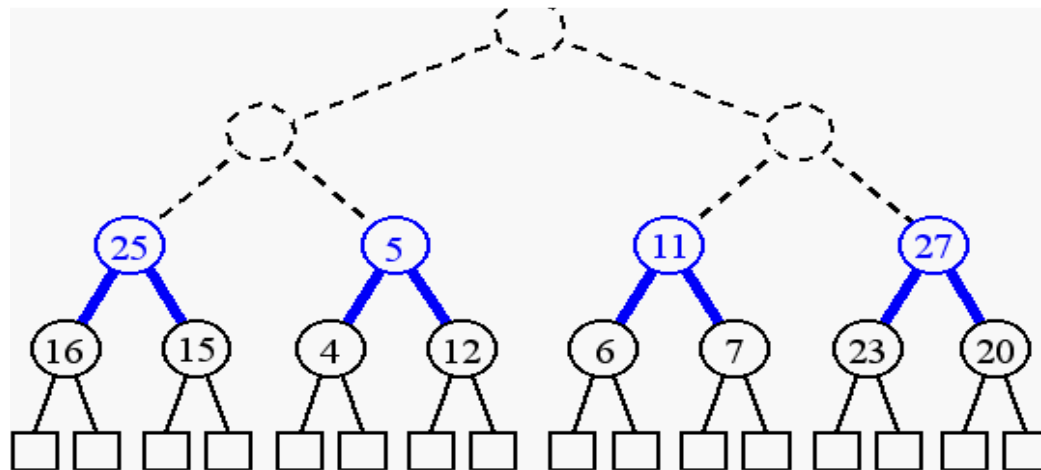
```
while (child <= h->size){  
    // find the smaller child of the current parent  
    if ( (child < h->size) &&  
        (h->elements[child] > h->elements[child+1] ) )  
        child++;  
    if (temp <= h->elements[child]) break;  
    // move to the next lower level  
    h->elements[parent] = h->elements[child];  
    parent = child;  
    child *= 2;  
}  
h->elements[parent] = temp;  
return item;  
}
```

Building a Heap

- build $(n + 1)/2$ trivial one-element heaps

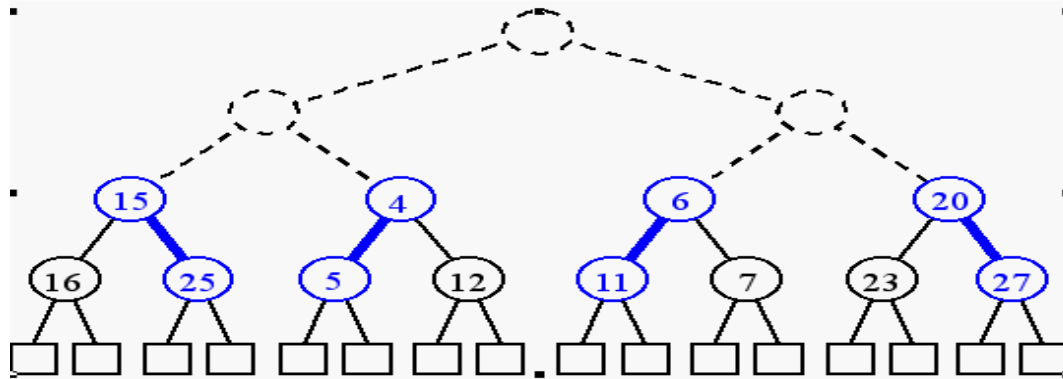


- build three-element heaps on top of them

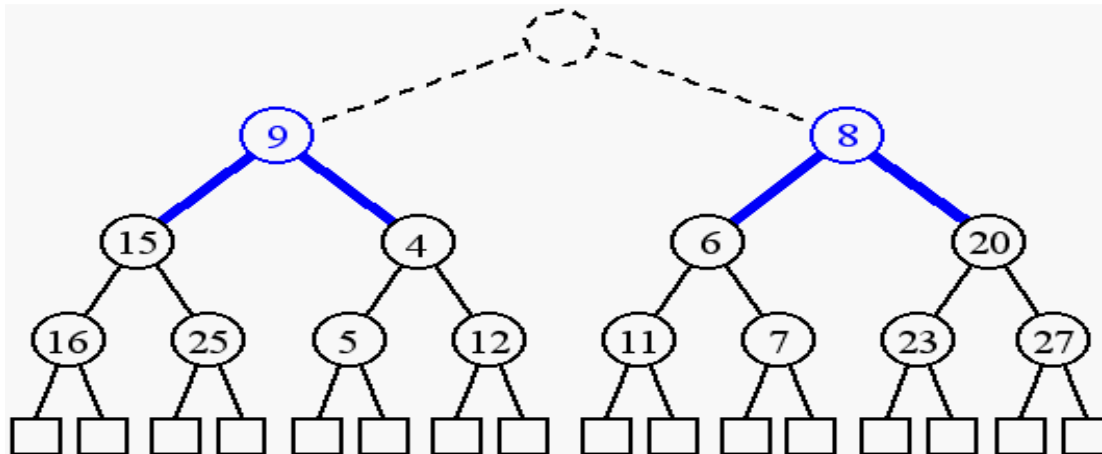


Building a Heap

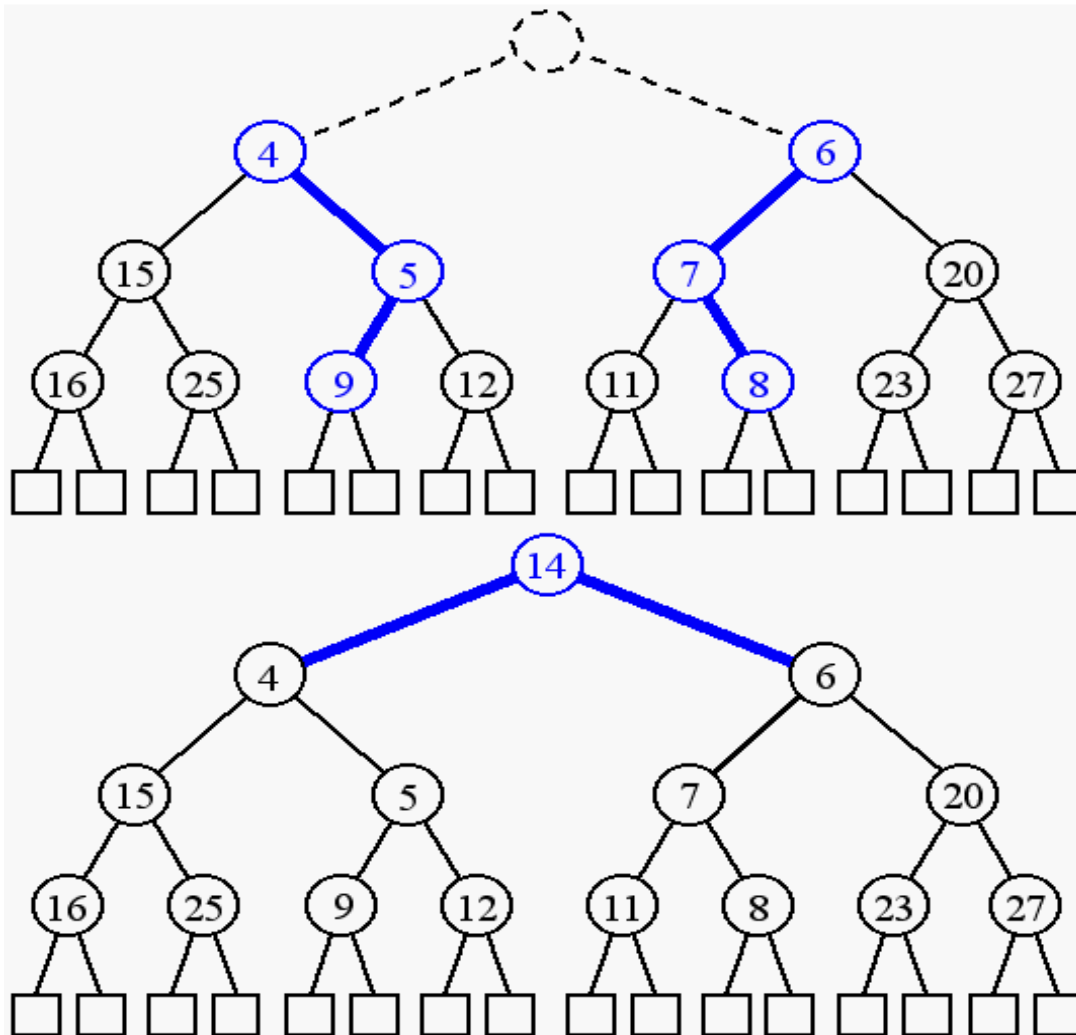
- Percolate-down to preserve the order property



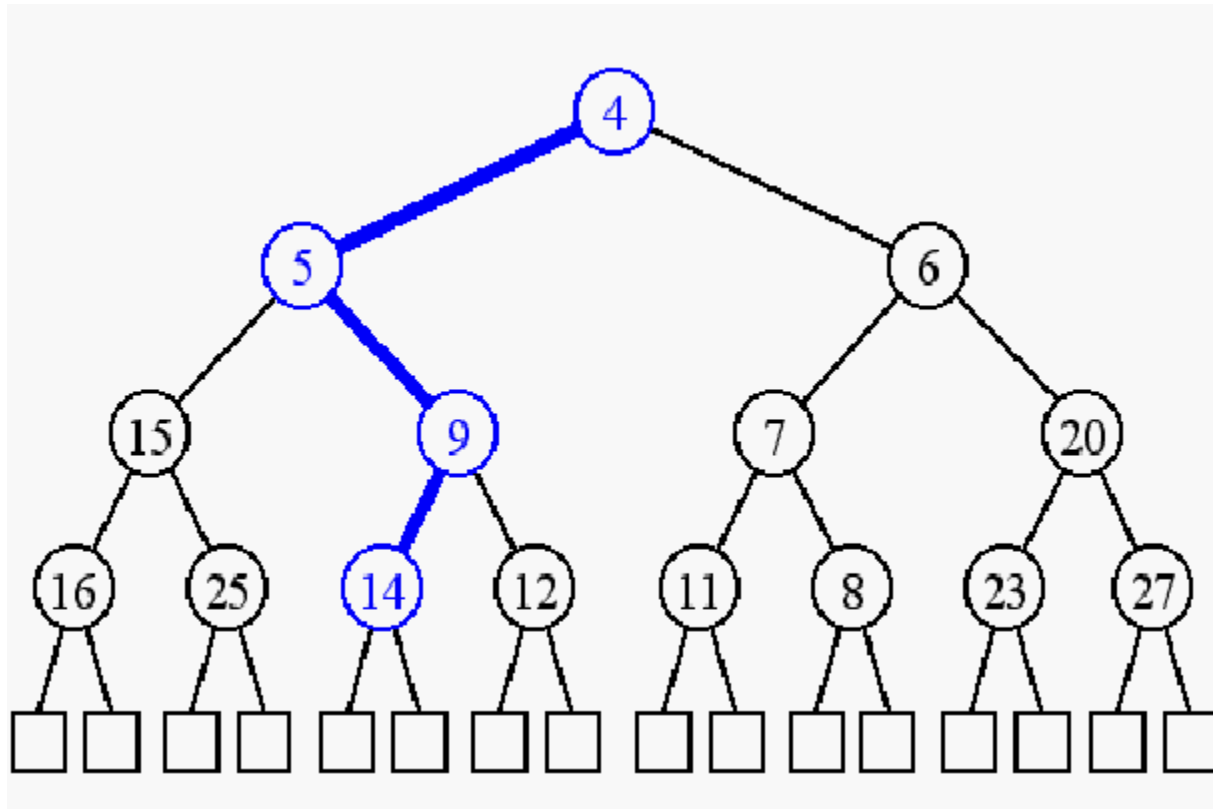
- Now form seven-element heaps



Building a Heap



Building a Heap



Running time for build a heap

- For $i = \lfloor n/2 \rfloor$ down to 1:
 - PercolateDown(i)
- Time complexity: $O(n)$
- Complete proof on the supplementary file

Heap Sorting

- Step 1: Build a heap
- Step 2: removeMin()

Appendix: A quick start tutorial for GDB

```
1  /* test.c */
2  /* Sample program to debug. */
3
4  #include <stdio.h>
5  #include <stdlib.h>
6
7  int main (int argc, char **argv)
8  {
9      if (argc != 3)
10         return 1;
11     int a = atoi (argv[1]);
12     int b = atoi (argv[2]);
13     int c = a + b;
14     printf ("%d\n", c);
15     return 0;
16 }
```

A quick start tutorial for GDB

- Compile with the -g option:
 - gcc -g -o test test.c
- Load the executable, which now contain the debugging symbols, into gdb:
 - gdb test

A quick start tutorial for GDB

- Now you should find yourself at the gdb prompt. There you can issue commands to gdb.
- Say you like to place a breakpoint at line 11 and step through the execution, printing the values of the local variables - the following commands sequences will help you do this:

A quick start tutorial for GDB

```
(gdb) break test.c:11
Breakpoint 1 at 0x401329: file test.c, line 11.
(gdb) set args 10 20
(gdb) run
Starting program: c:\Documents and Settings\VMatthew\Desktop/test.exe 10 20
[New thread 3824.0x8e8]

Breakpoint 1, main (argc=3, argv=0x3d5a90) at test.c:11
(gdb) n
(gdb) print a
$1 = 10
(gdb) n
(gdb) print b
$2 = 20
(gdb) n
(gdb) print c
$3 = 30
(gdb) c
Continuing.
30

Program exited normally.
(gdb)
```


Commands all you need to start:

```
break file:lineno - sets a breakpoint in the file at lineno.  
set args - sets the command line arguments.  
run - executes the debugged program with the given command line arguments.  
next (n) and step (s) - step program and step program until it  
                      reaches a different source line, respectively.  
print - prints a local variable  
bt - print backtrace of all stack frames  
c - continue execution.
```

- Type help at the (gdb) prompt to get a list and description of all valid commands.

Further GDB guides

- Peter's GDB tutorial <http://dirac.org/linux/gdb/>
- Tutorial on using the GDB debugger (Video)
<http://www.youtube.com/watch?v=k-zAgbDq5pk>