Recursion

Presentation Subtitle

Brad Miller David Ranum[1]

[1]Department of Computer Science
Luther College

12/19/2005

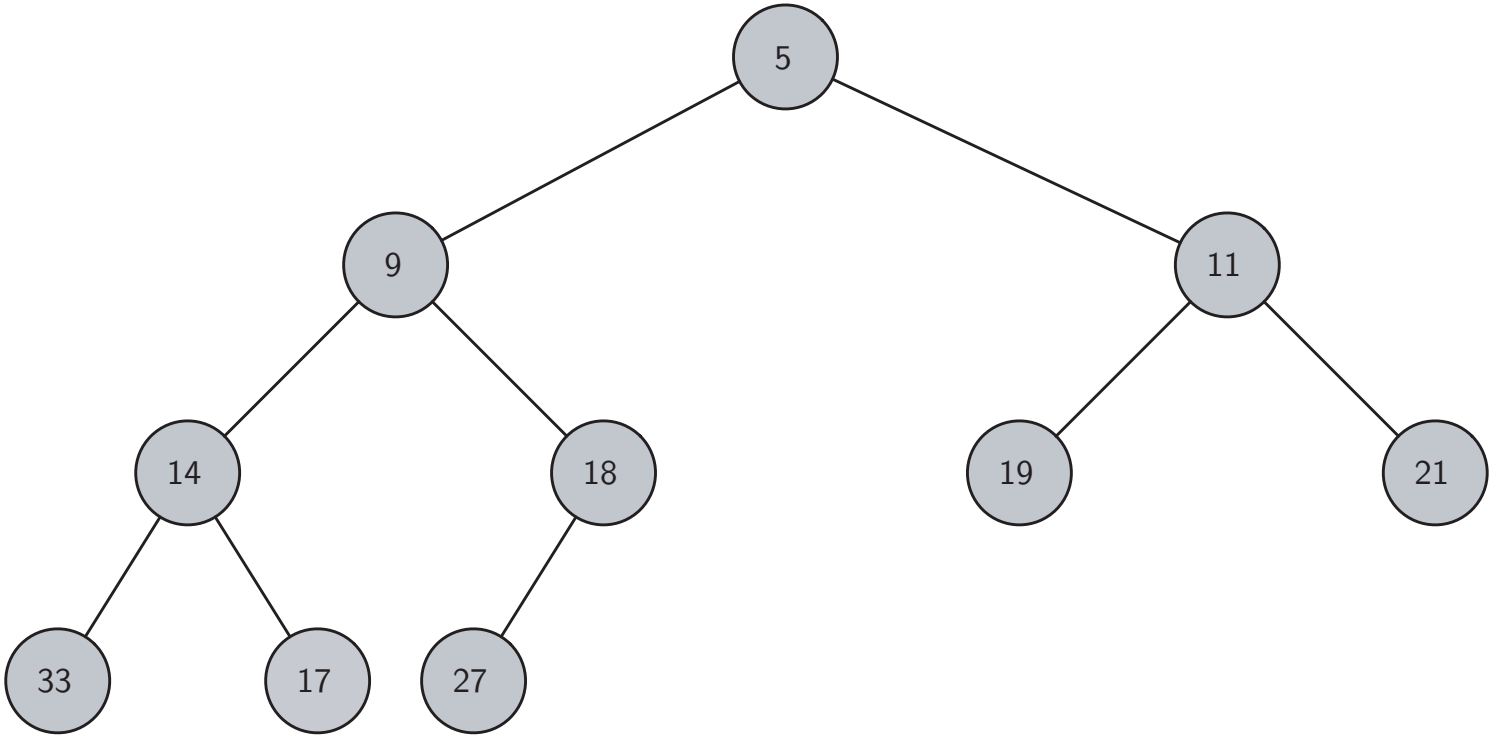
Outline

Outline

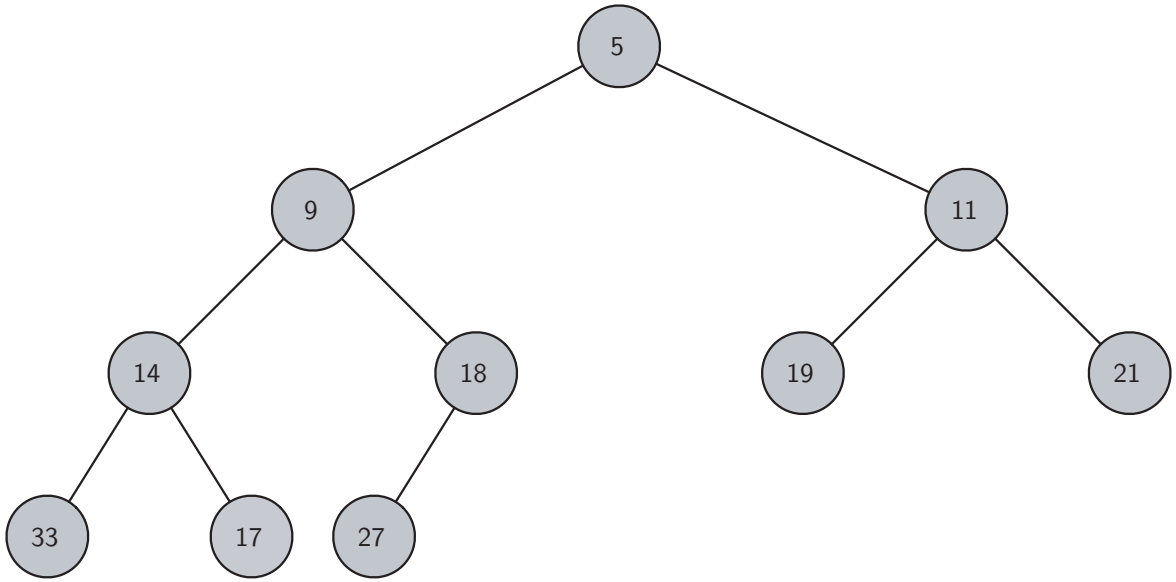
- `BinaryHeap()` creates a new binary heap.
- `insert(k)` adds a new item to the heap.
- `findMin()` returns the item with the minimum key value, leaving item in the heap.
- `delMin()` returns the item with the minimum key value, removing the item from the heap.
- `isEmpty()` returns true if the heap is empty, false otherwise.
- `size()` returns the number of items in the heap.
- `buidHeap(list)` builds a new heap from a list of keys.
- `decreaseKey(k)` finds a key in the heap and updates its key value to a new lower value.

Outline

A Complete Binary Tree



A Complete Binary Tree, Along with its List Representation

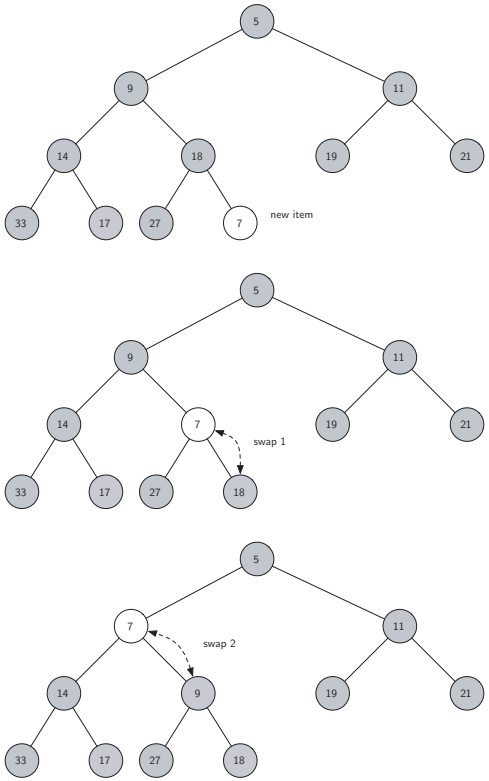


0	5	9	11	14	18	19	21	33	17	27	
0	1	2	3	4	5	6	7	8	9	10	11

Create a New Binary Heap

```
1       def __init__(self):
2           self.heapList = [0]
3           self.currentSize = 0
```

Percolate the New Node up to its Proper Position



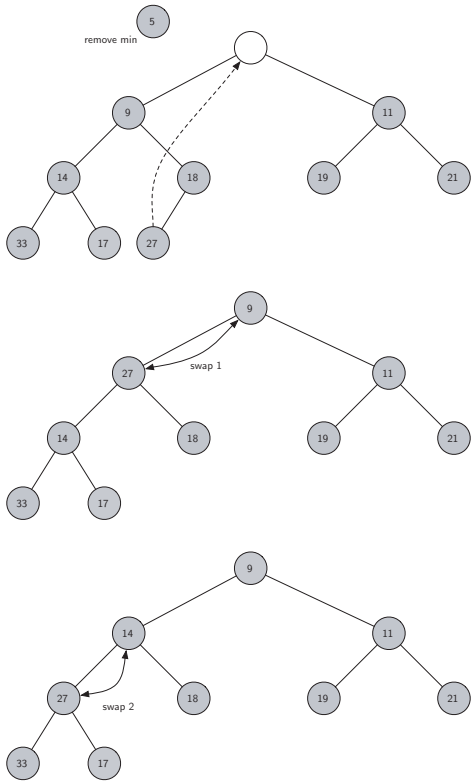
The `percUp` Method

```
1       def percUp(self,i):
2           while i > 0:
3               if self.heapList[i] < self.heapList[i/2]:
4                   tmp = self.heapList[i/2]
5                   self.heapList[i/2] = self.heapList[i]
6                   self.heapList[i] = tmp
7               i = i/2
```

Adding a New Item to the Binary Heap

```
1       def insert(self,k):
2           self.heapList.append(k)
3           self.currentSize = self.currentSize + 1
4           self.percUp(self.currentSize)
```

Percolating the Root Node down the Tree



The `percDown` Method

```
1  def percDown(self,i):
2      while (i * 2) <= self.currentSize:
3          child = i * 2
4          mc = self.minChild(i)
5          if self.heapList[i] > self.heapList[mc]:
6              tmp = self.heapList[i]
7              self.heapList[i] = self.heapList[mc]
8              self.heapList[mc] = tmp
9          i = mc
```

The `minChild` Method

```
1  def minChild(self,i):
2      if i*2 > self.currentSize:
3          return -1
4      else:
5          if i*2 + 1 > self.currentSize:
6              return i*2
7          else:
8              if self.heapList[i*2] < self.heapList[i*2+1]:
9                  return i*2
10             else:
11                 return i*2+1
```

Deleting the Minimum Item from the Binary Heap

```
1  def delMin(self):
2      retval = self.heapList[1]
3      self.heapList[1] = self.heapList[self.currentSize]
4      self.currentSize = self.currentSize - 1
5      self.percDown(1)
6      return retval
```

Building a New Heap from a List of Items

```
1  def buildHeap(self,alist):
2      i = len(alist) / 2
3      self.currentSize = len(alist)
4      self.heapList = [0] + alist[:]
5      while (i > 0):
6          self.percDown(i)
7          i = i - 1
```

Building a Heap from the List [9, 5, 6, 2, 3]

```
i = 2   [0, 9, 5, 6, 2, 3]
i = 1   [0, 9, 2, 6, 5, 3]
i = 0   [0, 2, 3, 6, 5, 9]
```