COMP 250
Lecture 24
heaps 2
Nov. 3, 2017
RECALL: min Heap (definition)

Complete binary tree with (unique) comparable elements, such that each node’s element is less than its children’s element(s).
parent = child / 2
left = 2*parent
right = 2*parent + 1

Heap index relations

Not used
parent = child / 2
left    = 2*parent
right   = 2*parent + 1
Heap index relations

parent = child / 2
left = 2*parent
right = 2*parent + 1
parent = \text{child} / 2
left = 2 \times \text{parent}
right = 2 \times \text{parent} + 1
ASIDE: an array data structure can be used for *any* binary tree. But this is uncommon and often inefficient.
add(element)

"upHeap"

removeMin()

"downHeap"
add(element)
{
    size = size + 1  // number of elements in heap
    heap[size] = element  // assuming array
        // has room for another element
    i = size

    // the following is sometimes called "upHeap"

    while (i > 1 and heap[i] < heap[i/2]){
        swapElements(i, i/2)
        i = i/2
    }
}
e.g. add(c)
e.g. \texttt{add(c)}
e.g. \(\text{add}(c)\)
e.g. $\text{add}(c)$
Given a list with size elements:

```java
buildHeap(list){
    create new heap array // length > list.size
    for (k = 0; k < list.size; k++)
        add( list[k] ) // add to heap[ ]
}
```
You could write the buildHeap algorithm slightly differently by putting all the list elements into the array at the beginning, and then `upheaping` each one.
Best case of buildHeap is ... ?

Suppose we want to add some elements to an empty heap:

```
  a e b c l u k m f
```

How many swaps do we need to add each element?

In the best case, ...
Best case of buildHeap is $\Theta(n)$

How many swaps do we need to add each element?

In the best case, the order of elements that we add is already a heap, and no swaps are necessary.
Worse case of buildHeap?

How many swaps do we need to add the $i$-th element?
Worse case of buildHeap?

How many swaps do we need to add the $i$-th element? Element $i$ gets added to some level, such that:

$$2^{level} \leq i < 2^{level} + 1$$
Worse case of buildHeap?

Thus, \( \text{level} = \text{floor}(\log_2 i) \)
Worse case of buildHeap

Suppose there are $i$ elements to add.

Worst case number of swaps needed to add node $i$.

$$t(n) = \sum_{i=1}^{n} \text{floor}(\log_2 i)$$
$\log_2 i$

$\text{floor}(\log_2 i)$
\[ t(n) = \sum_{i=1}^{n} \text{floor}(\log_2 i) \]

Area under the dashed curve is the total number of swaps (worst case) of buildHeap.
\[ t(n) \leq n \log_2 n \]
\[
\frac{1}{2} n \log_2 n \leq t(n) \leq n \log_2 n
\]
Thus, worst case: \texttt{buildHeap} is $\Theta(n \log_2 n)$

Next lecture I will show you a $\Theta(n)$ algorithm.
add(element)

"upHeap"

removeMin()

"downHeap"
e.g. `removeMin()`
Not used
removeMin()

Let heap[] be the array.
Let size be the number of elements in the heap.

removeMin()
{
    heap[1] = heap[size]
    heap[size] = null
    size = size - 1
    downHeap(1, size)
}

return tmpElement
removeMin()

Let heap[ ] be the array.
Let size be the number of elements in the heap.

removeMin()
{
    heap[1] = heap[size]
    heap[size] = null
    size = size - 1
    downHeap(1, size)                     // next slide
    return tmpElement
}
downHeap( startIndex , maxIndex ){

i = startIndex
while (2*i <= maxIndex){  // if there is a left child
    child = 2*i
    Find the smaller child (left or right?)
}
}

downHeap(startIndex, maxIndex) {

  i = startIndex
  while (2*i <= maxIndex) { // if there is a left child
    child = 2*i
    if (child < size) { // if there is a right sibling
      if (heap[child + 1] < heap[child]) // if rightchild < leftchild ?
        child = child + 1
    }
  }
}
downHeap(startIndex, maxIndex) {

  i = startIndex
  while (2*i <= maxIndex) {
    child = 2*i
    if child < size {
      if (heap[child + 1] < heap[child])
        child = child + 1
    }
    if (heap[child] < heap[i]){
      swapElements(i, child)
      i = child
    }
  }
}
Announcements

• Mycourses survey about MATH 240/235 and COMP 251

• Update on final exam grading policy
Final Exam grading policy

• Multiple Choice with 50 questions
• Four choices on each question
• *No penalty for incorrect answers*
  (so don’t leave any question blank)

• Grade out of 50
  = \( \max(0, -10 + \frac{6}{5} \times \text{raw number correct}) \)

• No crib sheet
Raw number correct for pure guessing?
(binomial distribution, \( n=50, \ p=0.25 \))

Hey, me and all my buddies averaged 25% raw scores on the final.