Heap Sort Notes

David Casperson

[1]Department of Computer Science
University of Northern BC

2007-11-12 / CPSC 200 Lecture

Outline

The Notion of a Heap

A heap is a binary tree

- where the left and right subtrees are heaps, and
- the root of the tree is larger than everything below it.

The Notion of a Heap
Array Based Heaps

- often implement heaps using trees and pointers
- can use arrays when the shape doesn't change often
- for 0-indexed arrays:
  - we store the left subnode of node i in node $2i + 1$
  - we store the right subnode of node i in node $2i + 2$

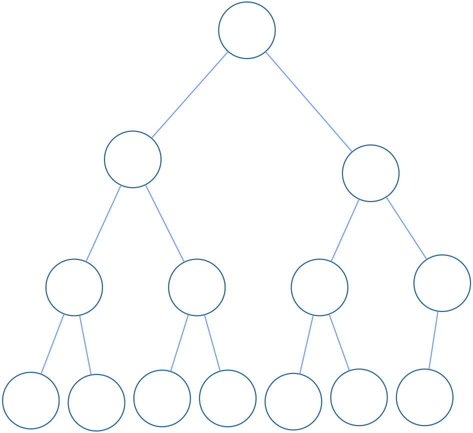
The Notion of a Heap
Array Based Heaps

- often implement heaps using trees and pointers
- can use arrays when the shape doesn't change often
- for 0-indexed arrays:
  - we store the left subnode of node i in node $2i + 1$
  - we store the right subnode of node i in node $2i + 2$
  - the parent of node i is in node $(i - 1)/2$.

Building a Heap

How to form a heap

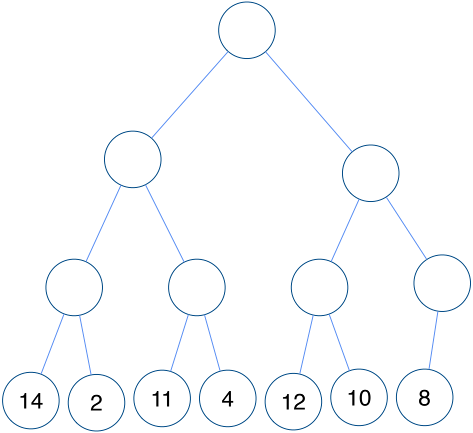
1
6
9
13
3
5
7
14
2
11
4
12
10
8



Building a Heap

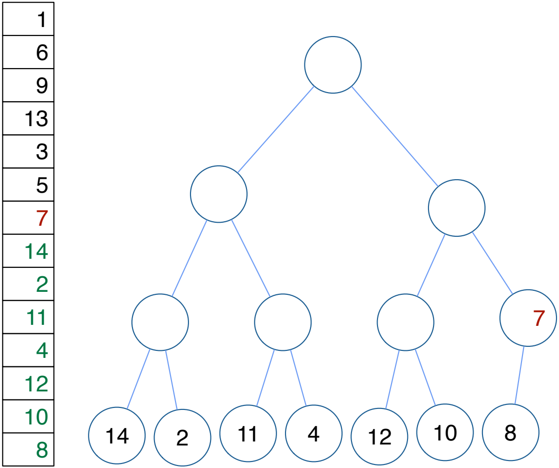
How to form a heap

1
6
9
13
3
5
7
14
2
11
4
12
10
8



Building a Heap

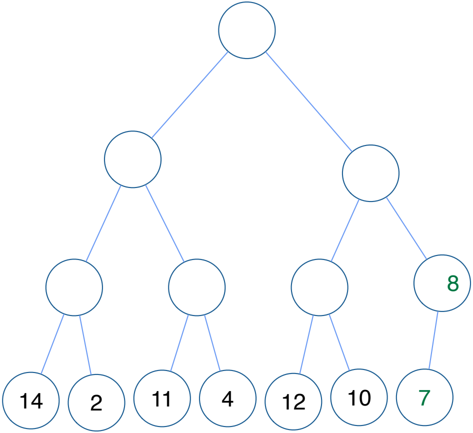
How to form a heap



Building a Heap

How to form a heap

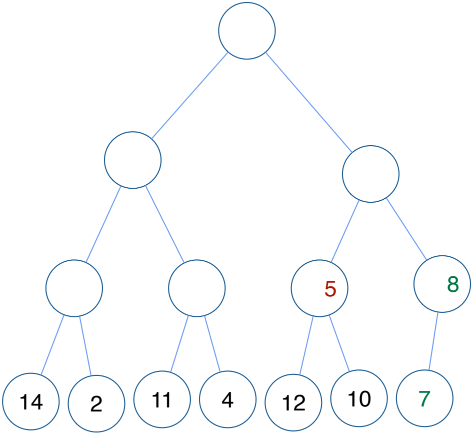
1
6
9
13
3
5
8
14
2
11
4
12
10
7



Building a Heap

How to form a heap

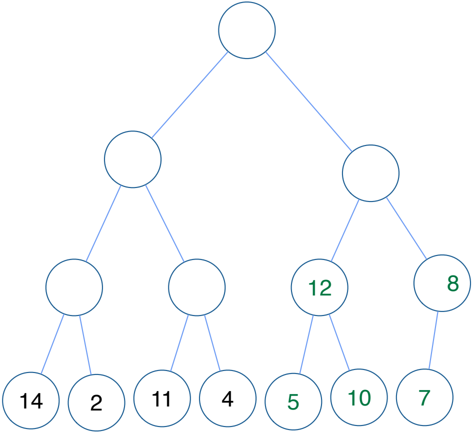
1
6
9
13
3
5
8
14
2
11
4
12
10
7



Building a Heap

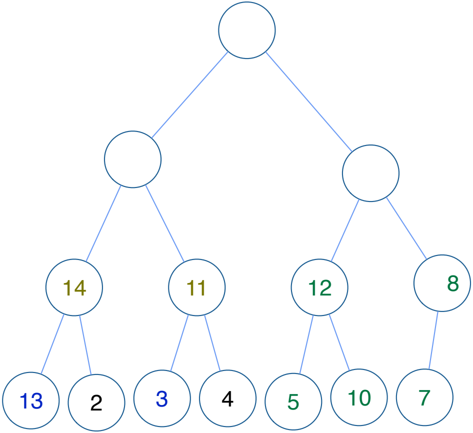
How to form a heap

1
6
9
13
3
12
8
14
2
11
4
5
10
7



Building a Heap
How to form a heap

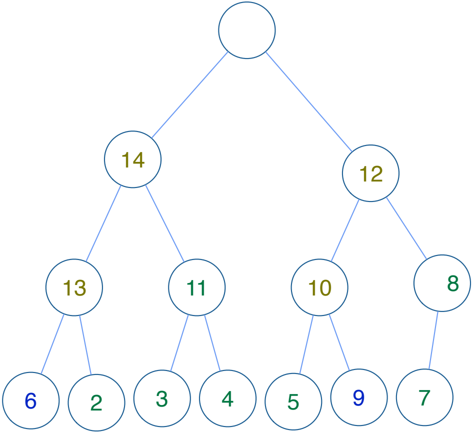
1
6
9
14
11
5
8
13
2
3
4
12
10
7



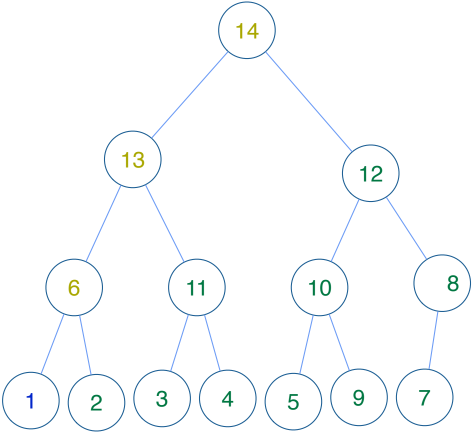
Building a Heap

How to form a heap

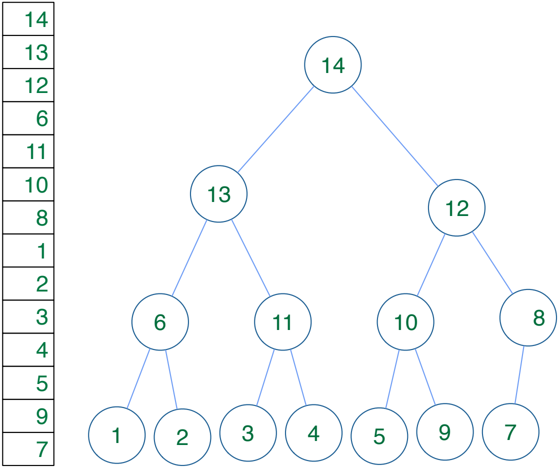
1
14
12
13
11
10
8
6
2
3
4
5
9
7



14
13
12
6
11
10
8
1
2
3
4
5
9
7



Building a Heap
How to form a heap



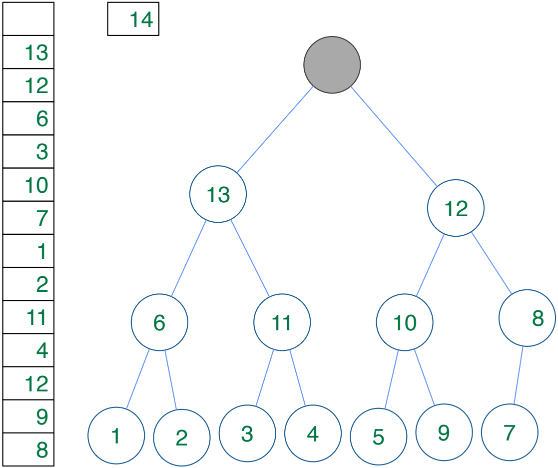
Building a Heap

How to form a heap

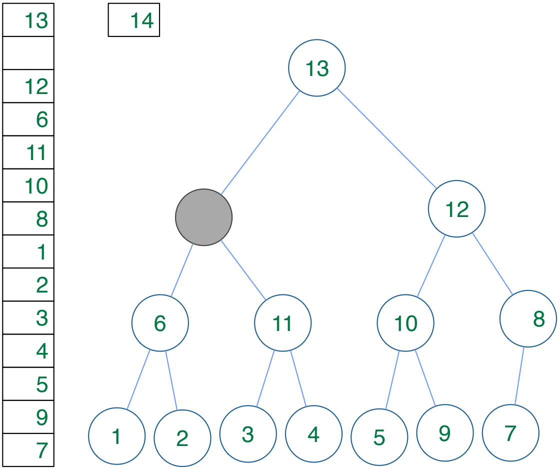
- Building a heap from the bottom up costs $\Theta(n)$-time.
- Building a heap from the bottom up costs $\Theta(1)$-space.
- Building a heap is not stable.

Unbuilding the Heap

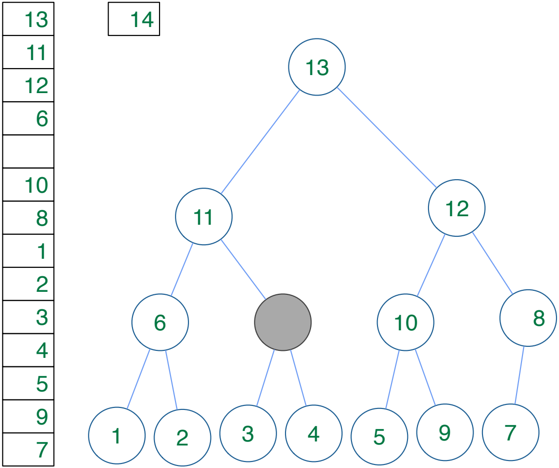
Making a hole



Unbuilding the Heap
Moving the Hole Down

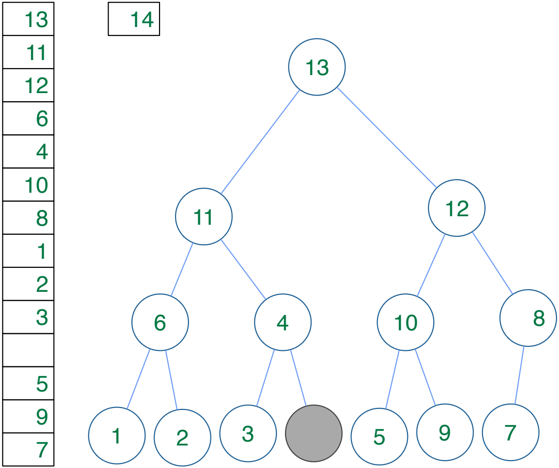


Unbuilding the Heap
Moving the Hole Down



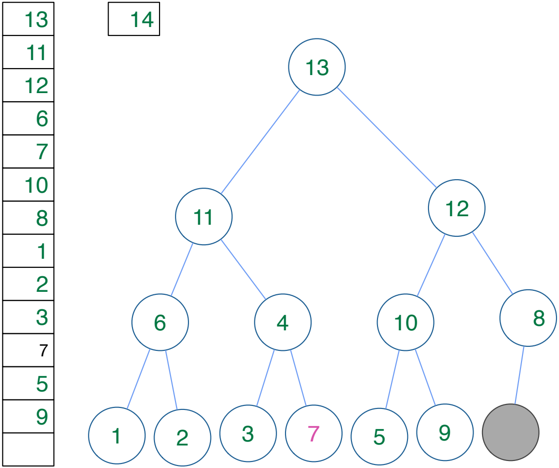
Unbuilding the Heap

Moving the Hole Down



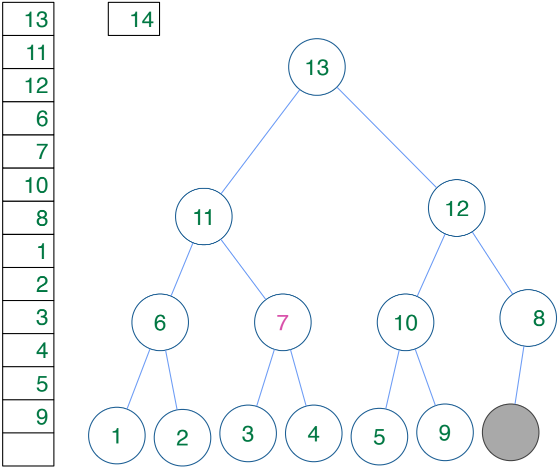
Unbuilding the Heap

Moving an Element Up

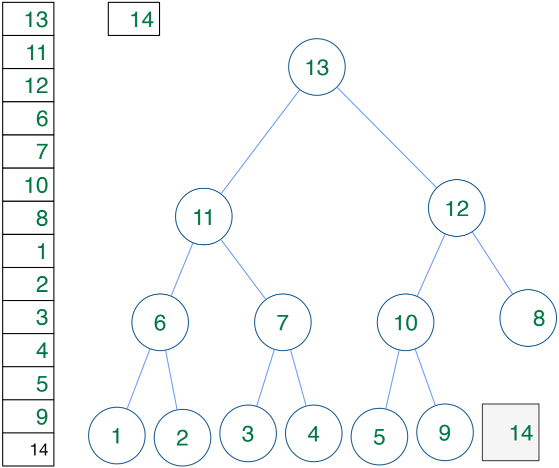


Unbuilding the Heap

Moving an Element Up

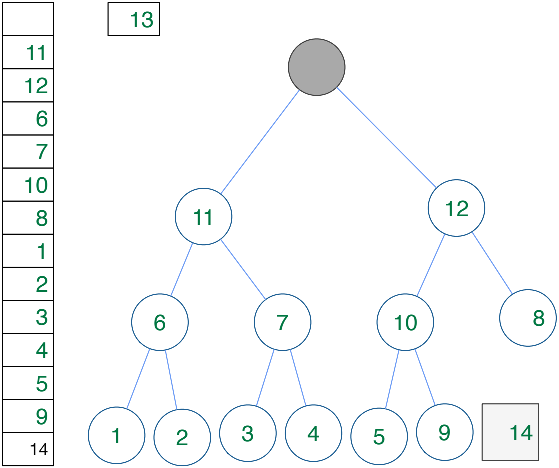


Unbuilding the Heap
Moving an Element Up



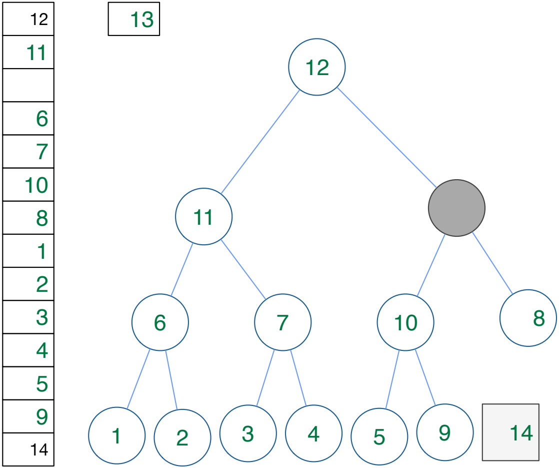
Unbuilding the Heap

The second element



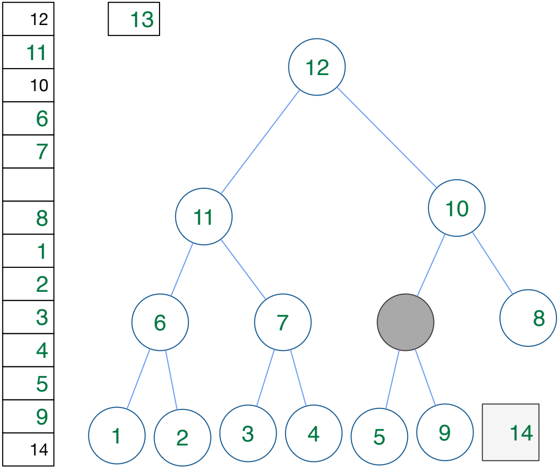
Unbuilding the Heap

The second element

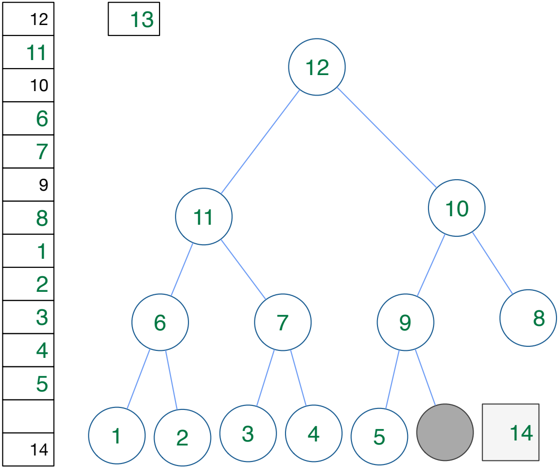


Unbuilding the Heap

The second element

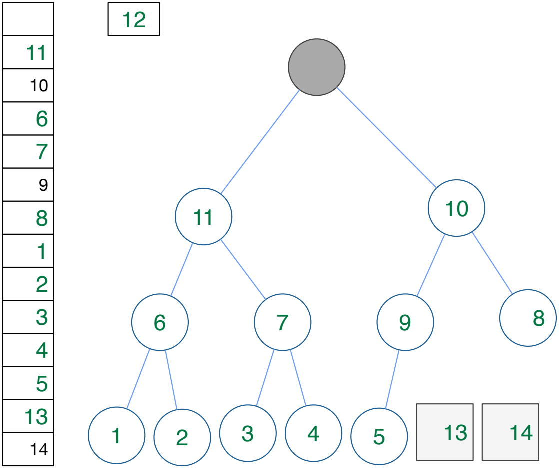


Unbuilding the Heap
The second element



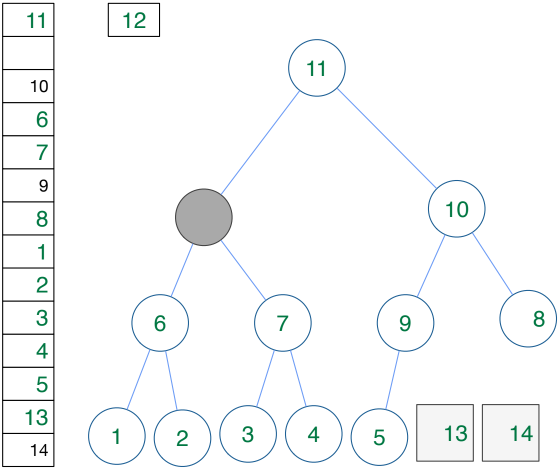
Unbuilding the Heap

The second element

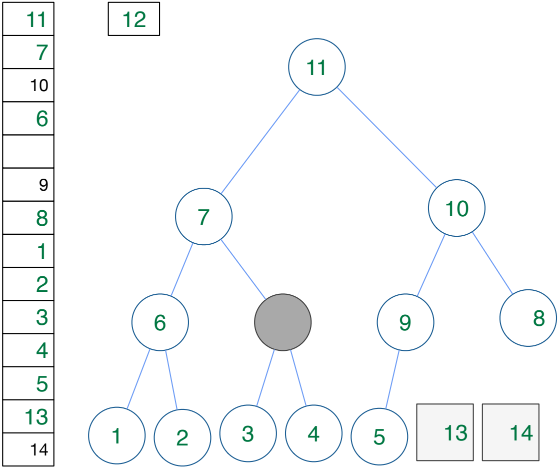


Unbuilding the Heap

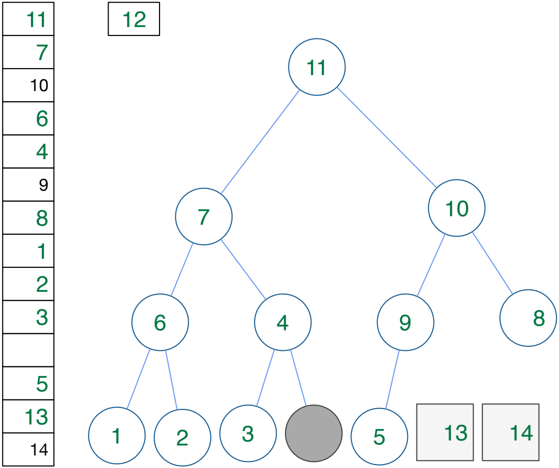
The third element



Unbuilding the Heap
The third element

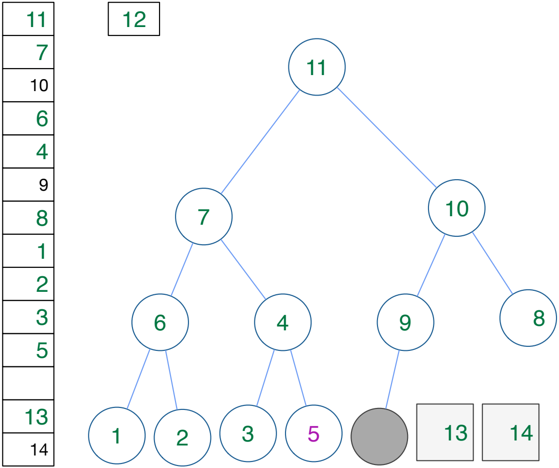


Unbuilding the Heap
The third element

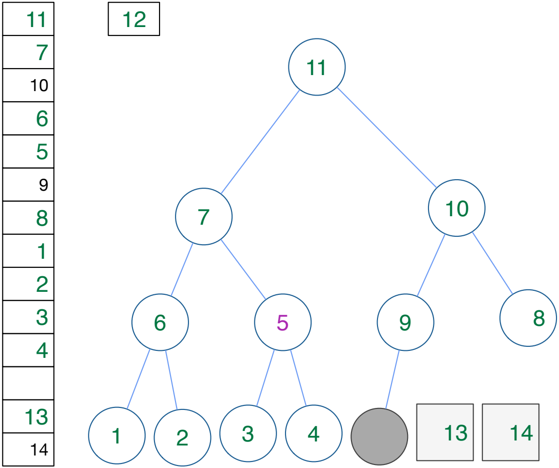


Unbuilding the Heap

The third element

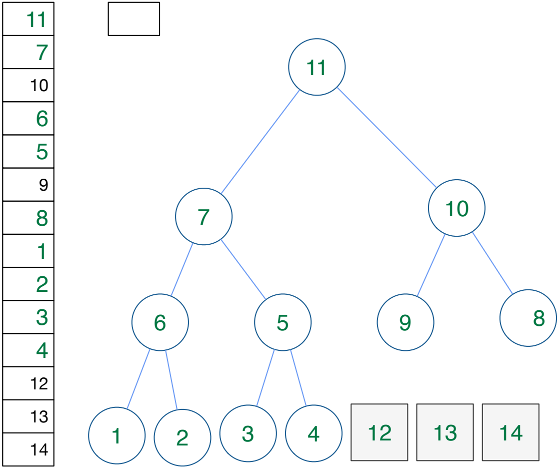


Unbuilding the Heap
The third element



Unbuilding the Heap

The third element



- It is faster to move a hole down, then an element up.
- Moving a hole down costs $\Theta(\log n)$-time.
- Moving a element up costs worst-case $\Theta(\log n)$-time.
- Moving a element up costs average-case $\Theta(1)$-time?
- Unbuilding the whole heap costs $\Theta(\log(n!))$-time.
- Unbuilding the whole heap costs $\Theta(1)$-space.
- Unbuilding the heap is not stable.

Heap Sort:

- uses a *heap*
- represents a binary tree in an array
- has worst-case running time of $\Theta(n \log n)$
- has constant extra space usage.

Quick Sort Summary

Report Card

- $T_{worst}(n) = \Theta(n \log n)$

Quick Sort Summary

Report Card

- $T_{worst}(n) = \Theta(n \log n)$
- $T_{ave}(n) = \Theta(n \log n)$
  almost all the time

Quick Sort Summary
Report Card

- $T_{worst}(n) = \Theta(n \log n)$
- $T_{ave}(n) = \Theta(n \log n)$
  almost all the time
- $O(1)$ extra storage for recursion

Quick Sort Summary
Report Card

- $T_{worst}(n) = \Theta(n \log n)$
- $T_{ave}(n) = \Theta(n \log n)$
  almost all the time
- $\mathrm{O}(1)$ extra storage for recursion
- not stable.

The End