An Efficient Algorithm for Concurrent Priority Queue Heaps

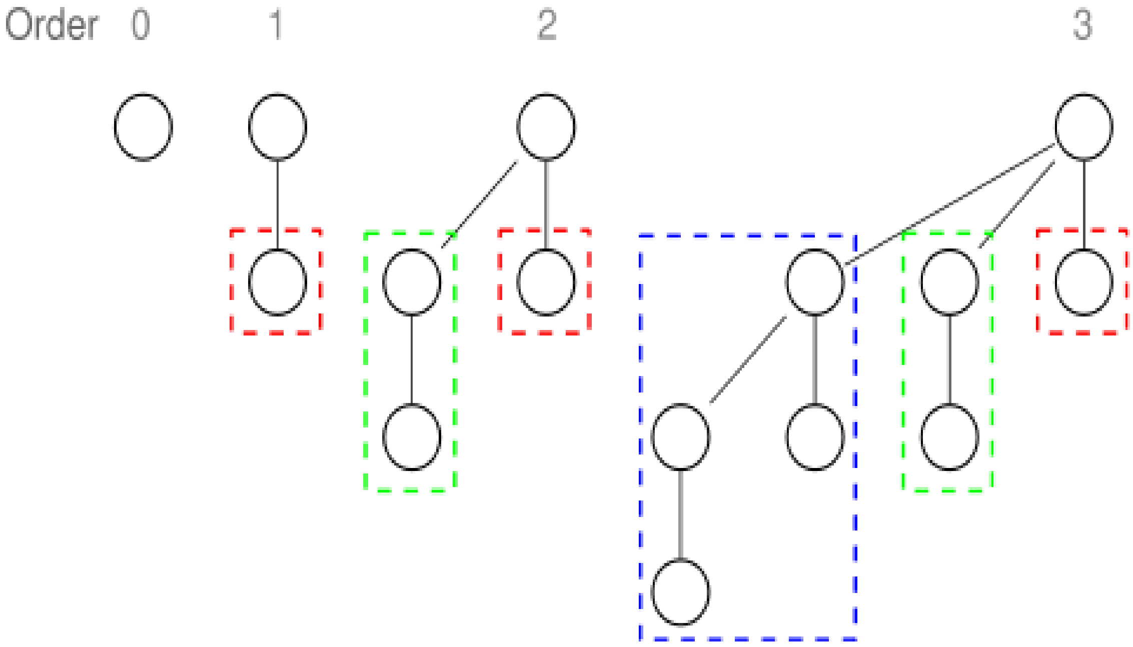
Shouzheng Yang
CSE6490A
Class Presentation 1
Feb 9, 2011

Outline

- Introduction
  - Priority Queue
  - Array based queue
- The algorithm
  - Technique
  - Problems
  - Solutions
- Following tasks

Priority Queue

- Priority Queue
  - Binary heap
  - Binomial heap
  - Fabonacci heap
  - Fusion tree



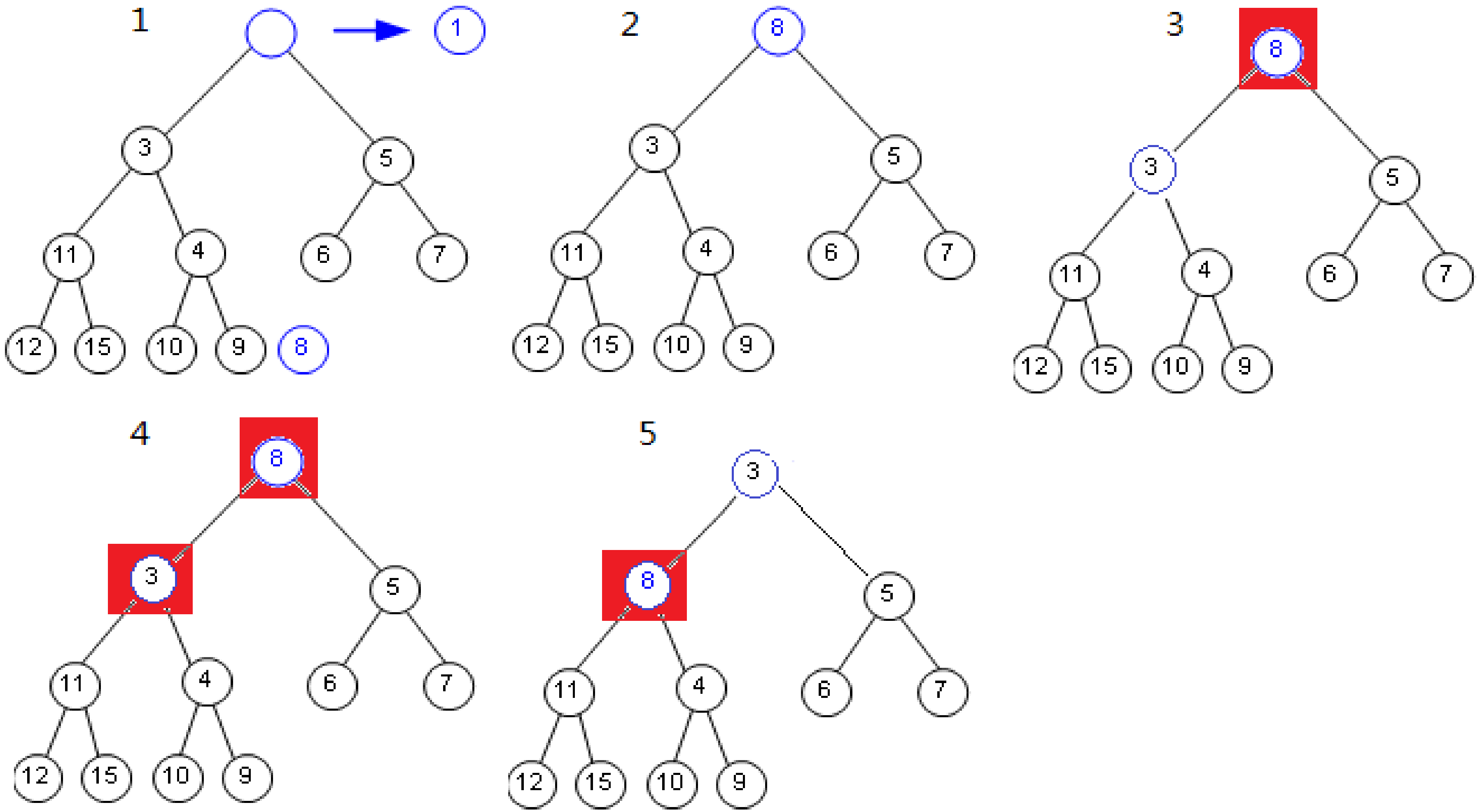
Array Based Heap

- Root occupies location 1.
- For the node at location i, the left and right child of that node will be at 2i and 2i+1, respectively.
- Space efficient since no item exists in level L unless level L-1 is full.

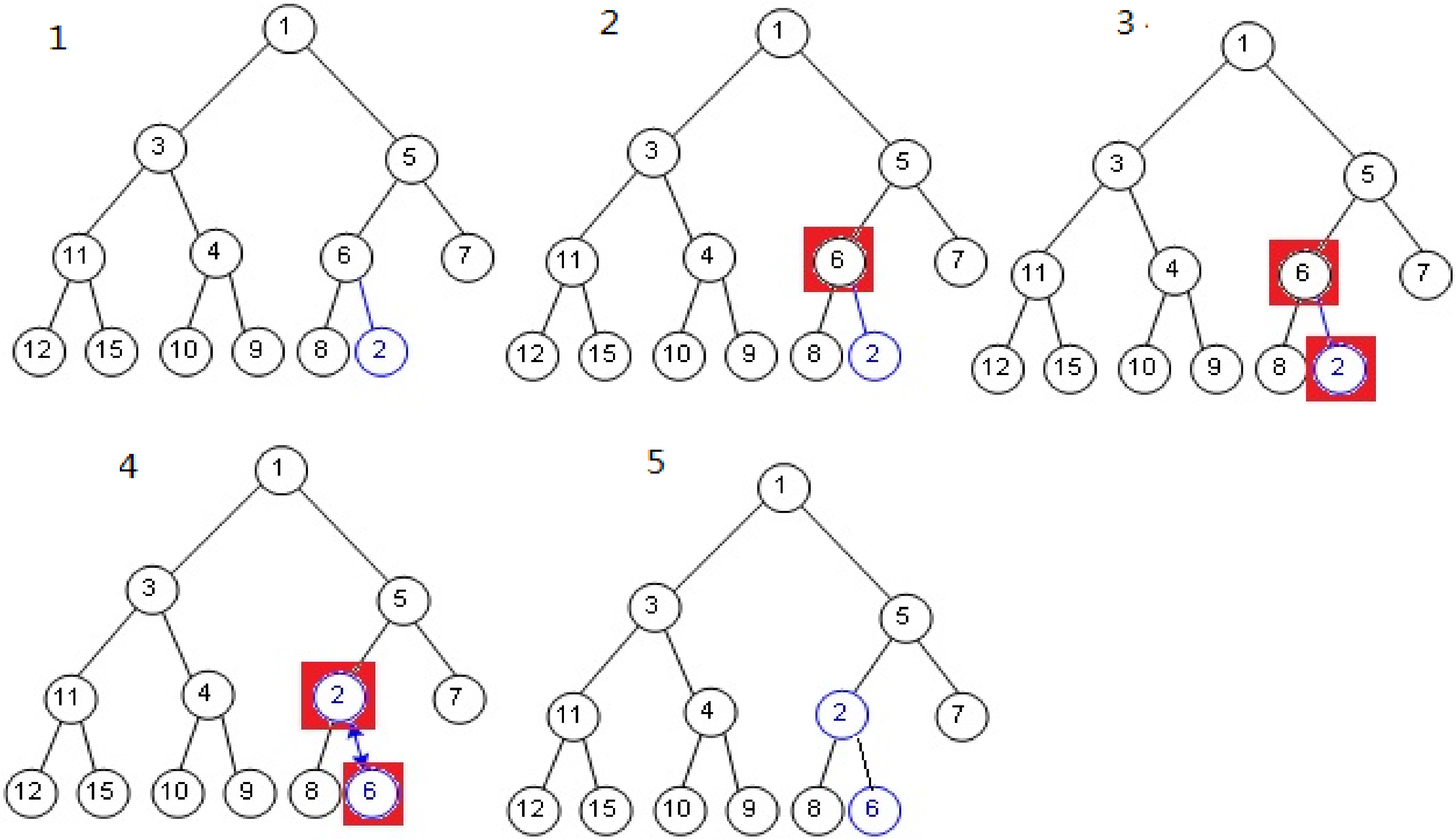
The Algorithm

- Insertion and deletion in opposite directions.
- Lock mechanism
  - A lock on the heap’s size
  - Locks on each node in the heap.
- Tags
- Bit-reversal technique

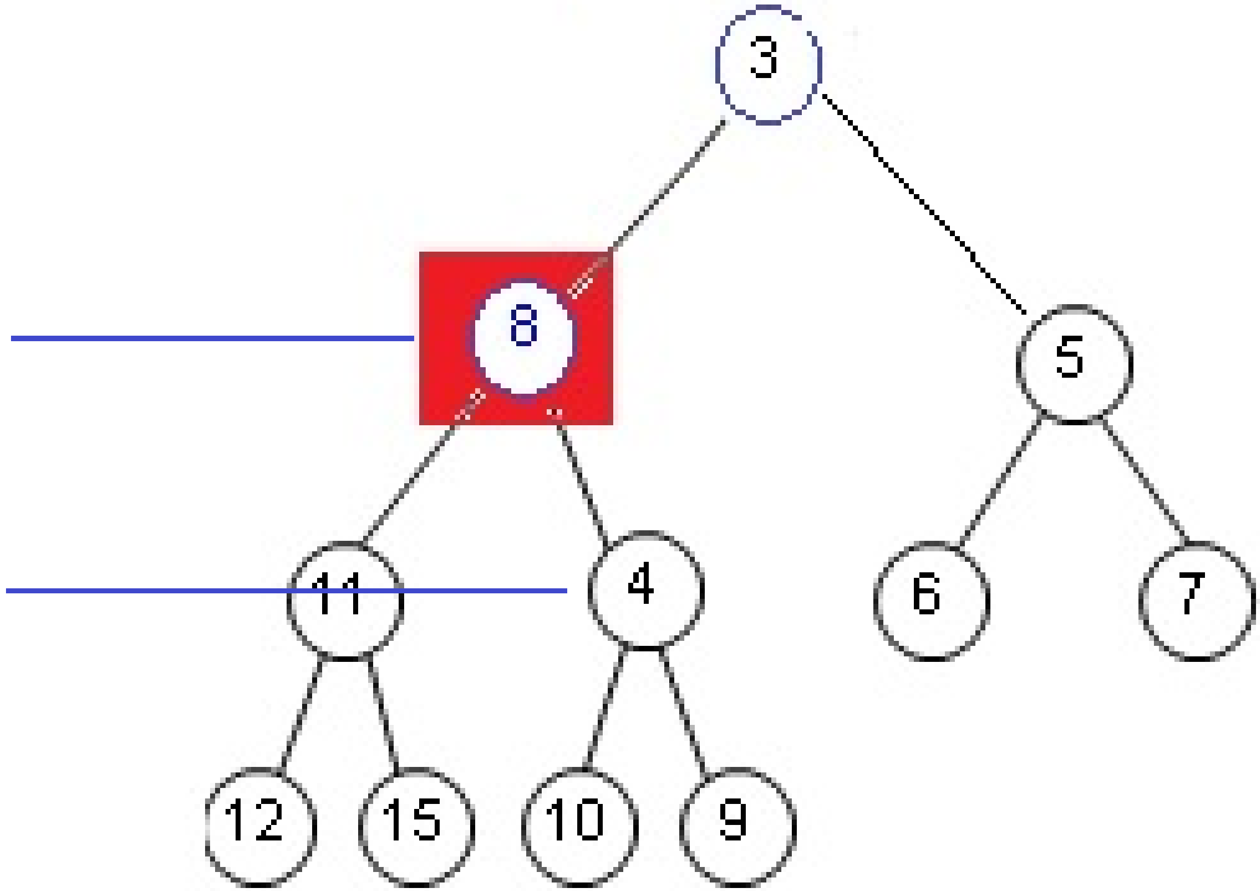
Deletion


1
1
3
5
11
4
6
7
12
15
10
9
8
2
8
3
5
11
4
6
7
12
15
10
9
3
8
3
5
11
4
6
7
12
15
10
9
4
8
3
5
11
4
6
7
12
15
10
9
5
3
8
5
11
4
6
7
12
15
10
9

Insertion


1
1
3
5
11
4
6
7
12
15
10
9
8
2
2
1
3
5
11
4
6
7
12
15
10
9
8
2
3
1
3
5
11
4
6
7
12
15
10
9
8
2
4
1
3
5
11
4
2
7
12
15
10
9
8
6
5
1
3
5
11
4
2
7
12
15
10
9
8
6

Problems



Deleting

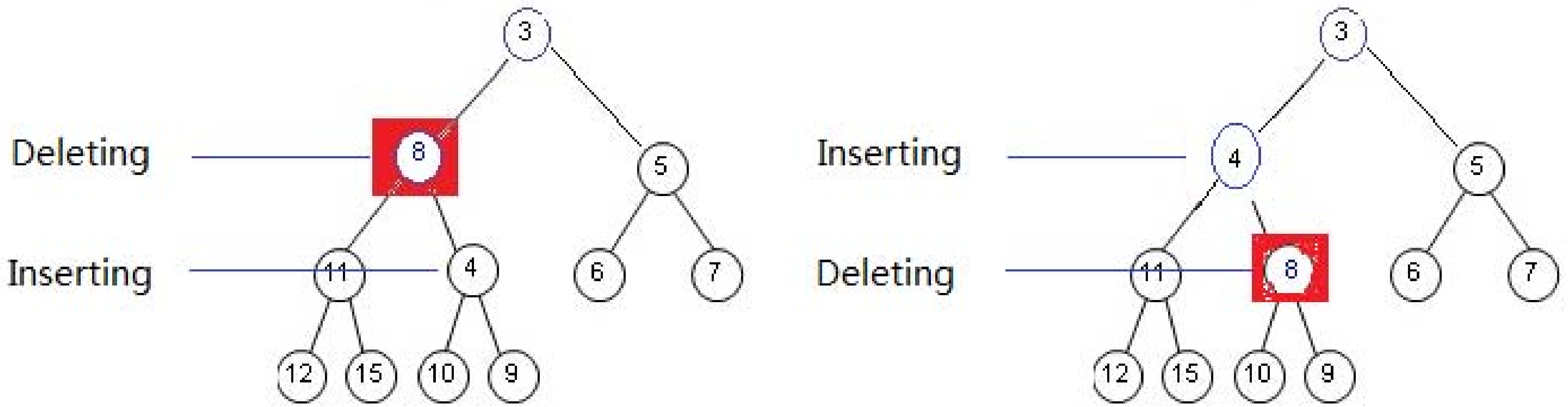
Inserting

Tags

- Available:
  - Valid node.
- Empty
  - Empty node, currently contains no data.
- PID
  - Processor identifier. Tagged on a node that has started insertion, and is being moved into place.

Problems

The inserted item has been moved upwards by a delete operation. The insert operation moves upward in pursuit of the inserted item.



Bit-reversal Technique

- Purpose
    - Improve concurrency
- All nodes in the last level of the heap to the left of the last item must be non-empty?
- Bit-reversal technique
    - Consecutive insertions traverse different sub-trees, avoiding early encounter.
    - Eg., it the third level of a heap (nodes 8 -15), eight consecutive insertions would start from nodes 8, 12, 10, 14, 9, 13, 11 and 15.

Results

- Significantly superior performance on large heaps with mixed insertion/deletion workloads compared to previous work at that time.

- Reasonable performance on small heaps.

Looking Ahead

- Implementation
  - Sequential
  - Concurrent
    - With / without bit-reversal technique
    - Modify the code for issues not mentioned in paper
      - Eg. The way of choosing a waited thread to be notified.
  - Compare the performance

Reference

- Hunt, G., Michael, M., Parthasarathy, S., Scott, M.: An efficient algorithm for concurrent priority queue heaps. Information Processing Letters 60(3) (1996) 151-157
- Rao, N., Kumar, V.: Concurrent Access of Priority Queues. In: IEEE Transactions on Computers, Citeseer (1988)
- Cormen, T.: Introduction to algorithms. The MIT press (2001)

Thank you for your attention!