NPTEL MOOC,JAN-FEB 2015
Week 5, Module 5

DESIGN AND ANALYSIS OF ALGORITHMS

Heaps: Updating values, heap sort

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

Heaps

- Heaps are a tree implementation of priority queues
  - insert( ) and delete_max( ) are both O(log N)
  - heapify( ) builds a heap in O(N)
  - Tree can be manipulated easily using an array

Recall Dijkstra's algorithm

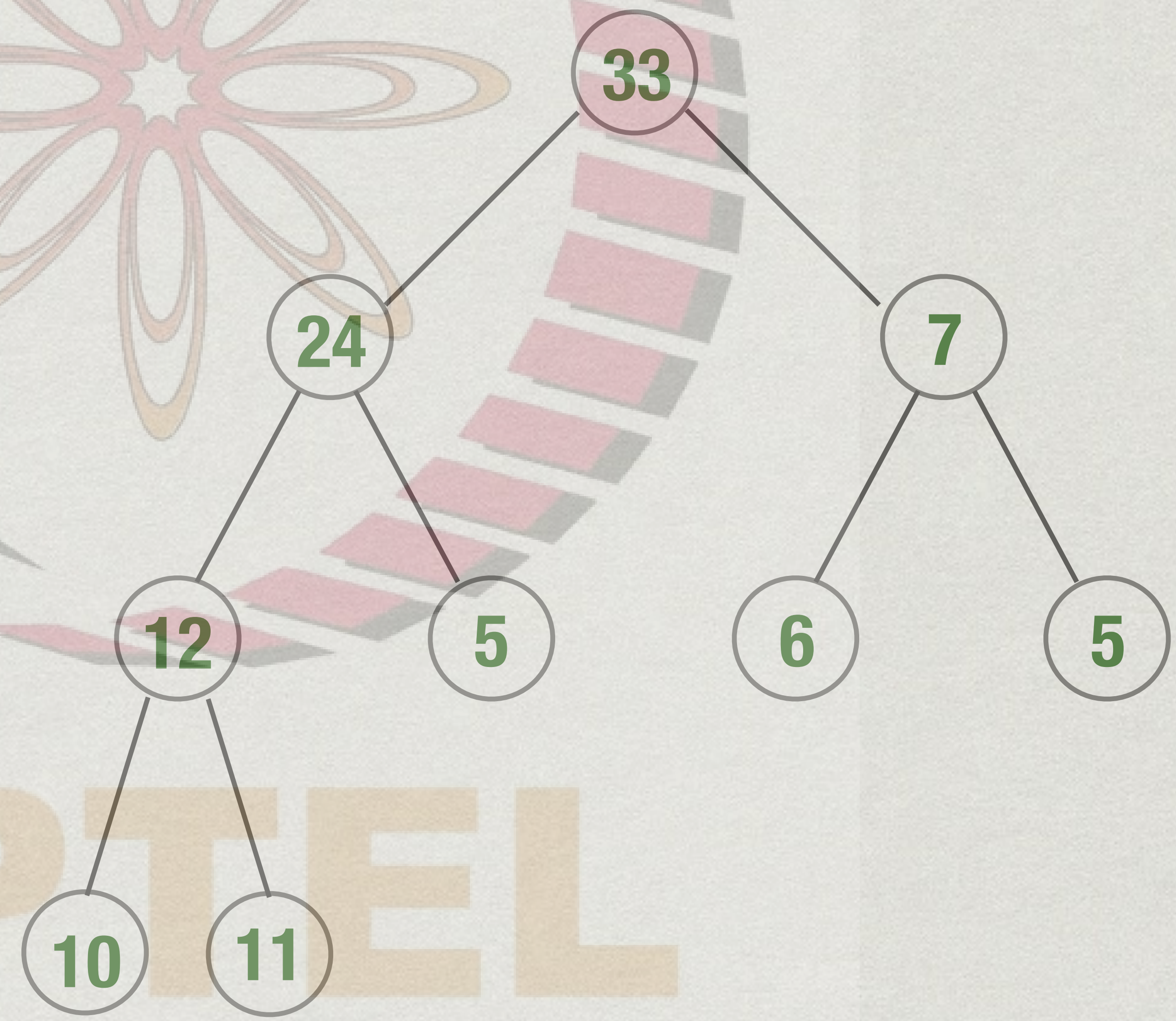
- Maintain two arrays
  - `Visited[ ]`, initially `False` for all i
  - `Distance[ ]`, initially ∞ for all i
    - For ∞, use sum of all edge weights + 1
- Set `Distance[1] = 0`
- Repeat, until all vertices are burnt
  - Find `j` with minimum `Distance`
  - Set `Visited[j] = True`
  - Recompute `Distance[k]` for each neighbour `k` of `j`

Bottlenecks

- Find `j` with minimum `Distance`
  - Naive implementation takes O(n) time
  - Maintain `Distance[]` as min-heap, delete_min( ) is O(log n)
- Recompute `Distance[k]` for each neighbour `k` of `j`
  - Use adjacency lists to look up neighbours efficiently
  - To recompute `Distance[k]`, need to update heap values
    - Not a basic operation on heaps, as defined

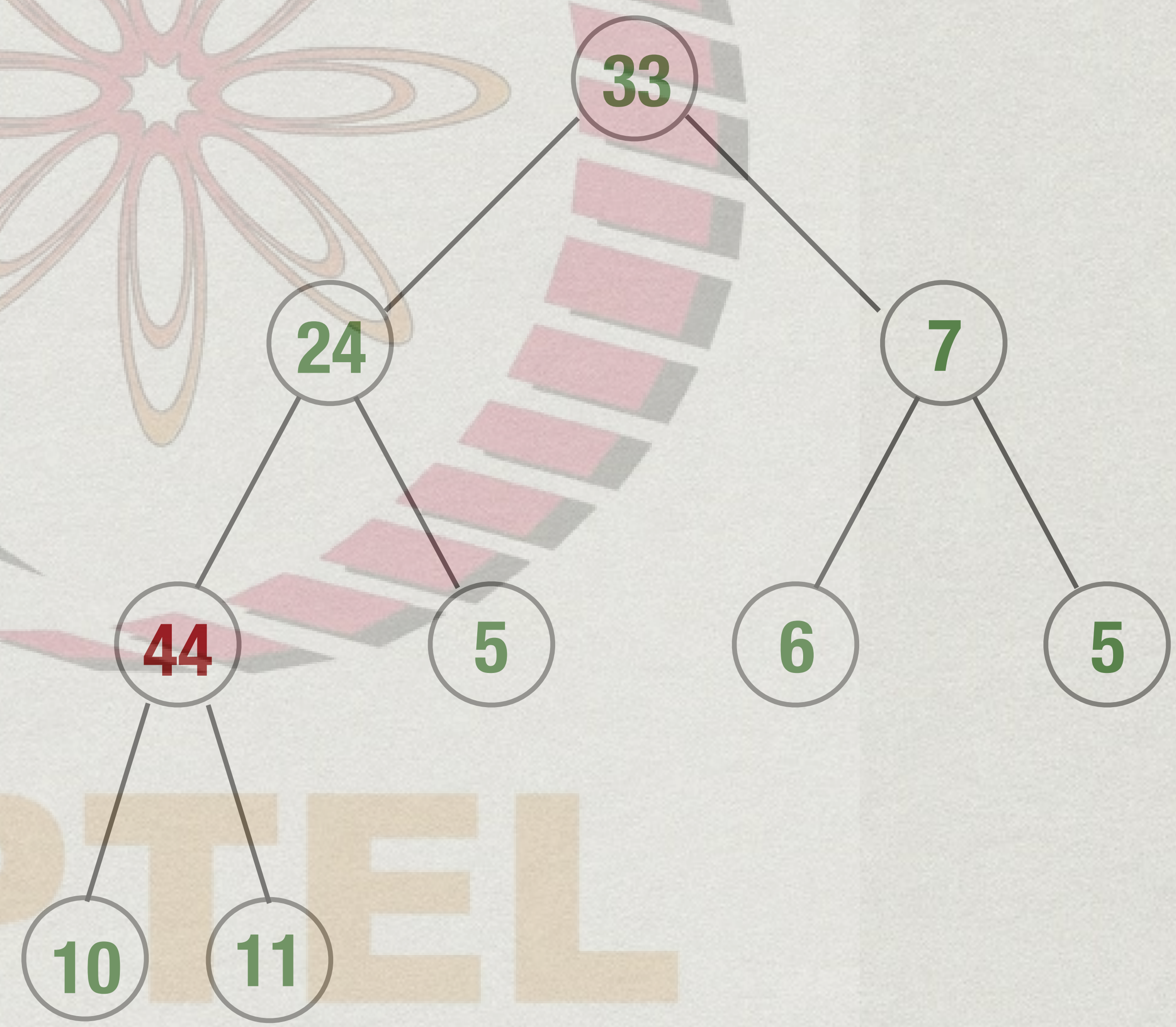
Updating values

- Change 12 to 44
  - Increasing a value can create heap violation with parent
  - Fix violations upwards, to root



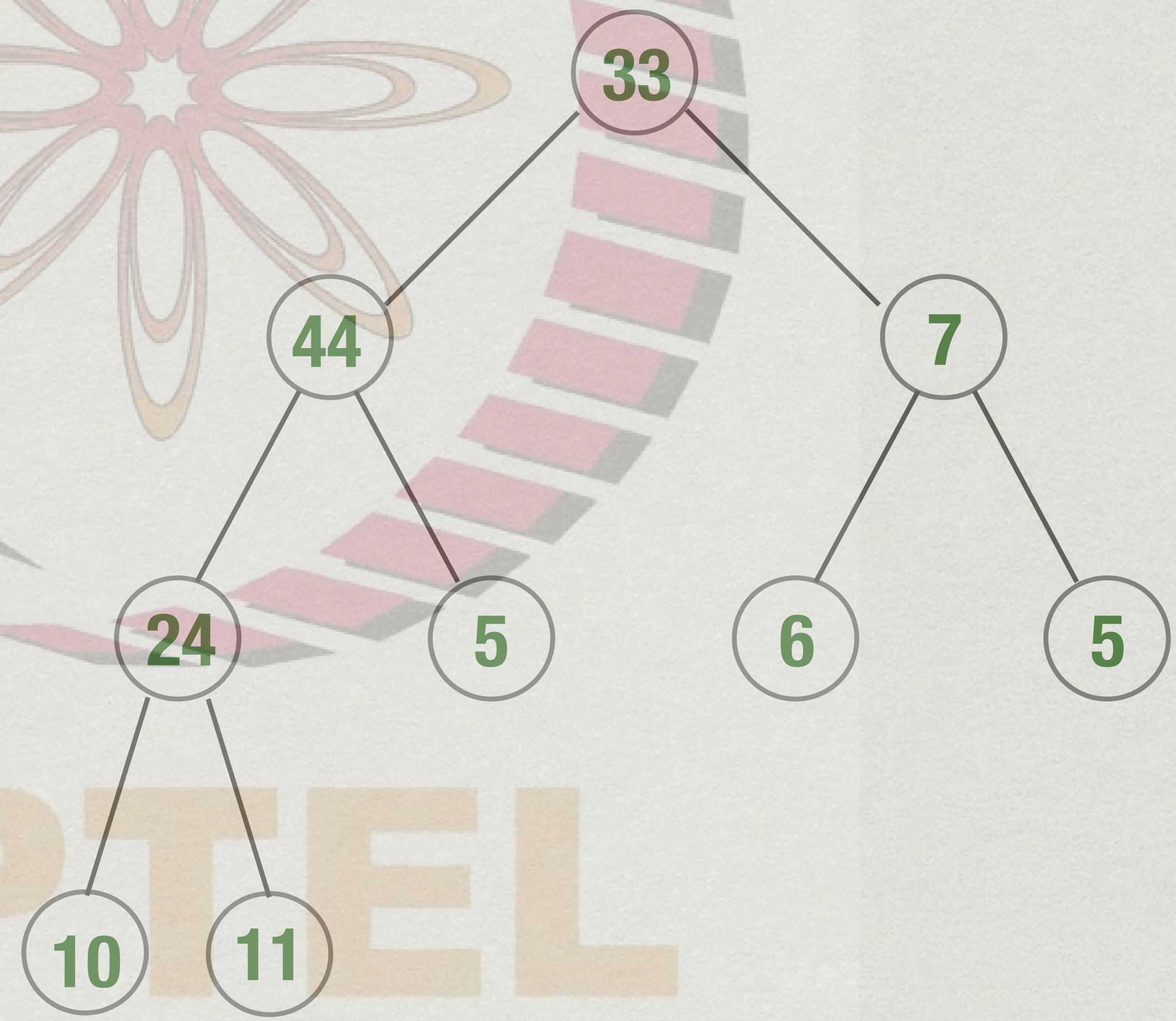
Updating values

- Change 12 to 44
  - Increasing a value can create heap violation with parent
  - Fix violations upwards, to root



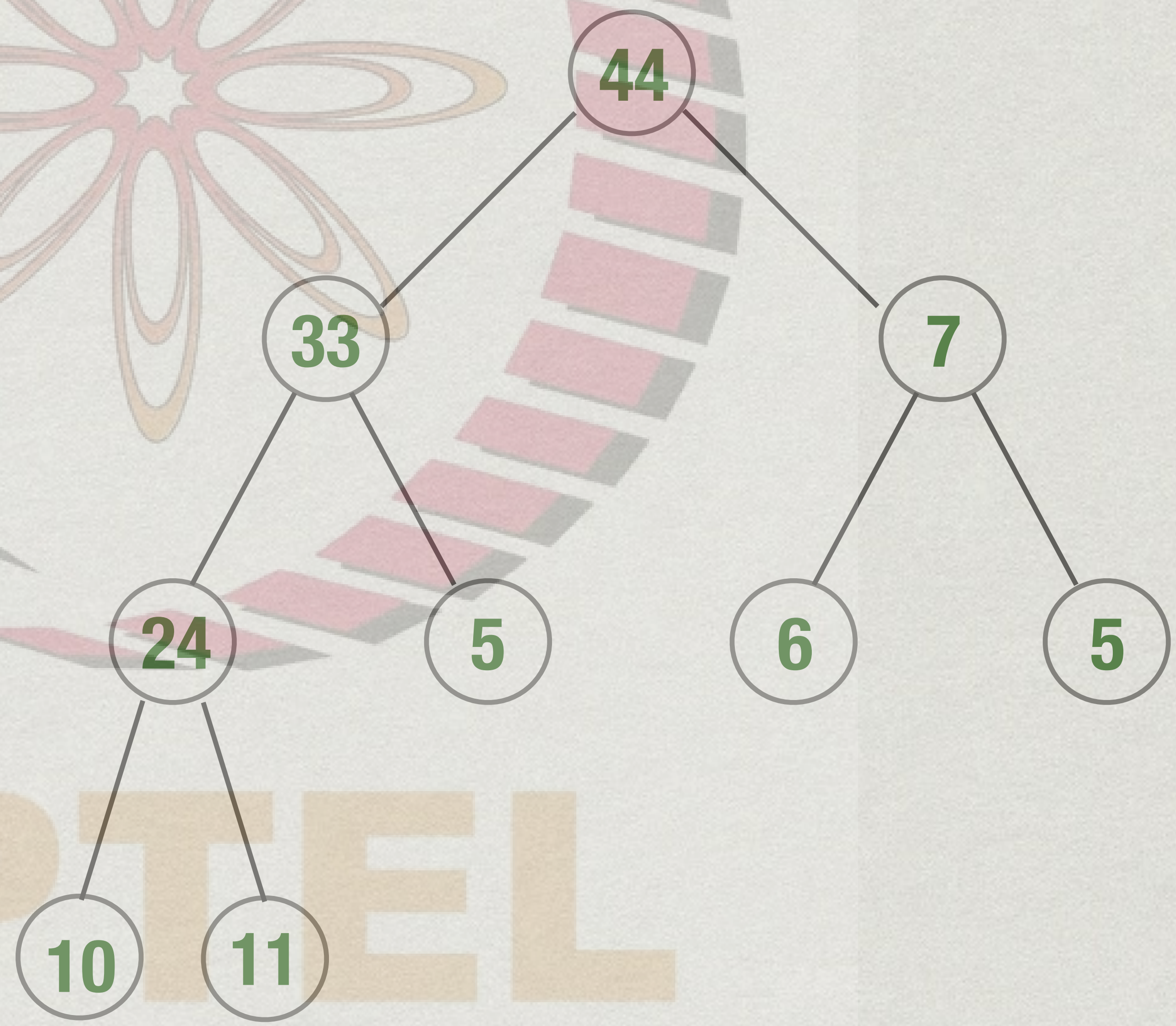
Updating values

- Change 12 to 44
  - Increasing a value can create heap violation with parent
  - Fix violations upwards, to root



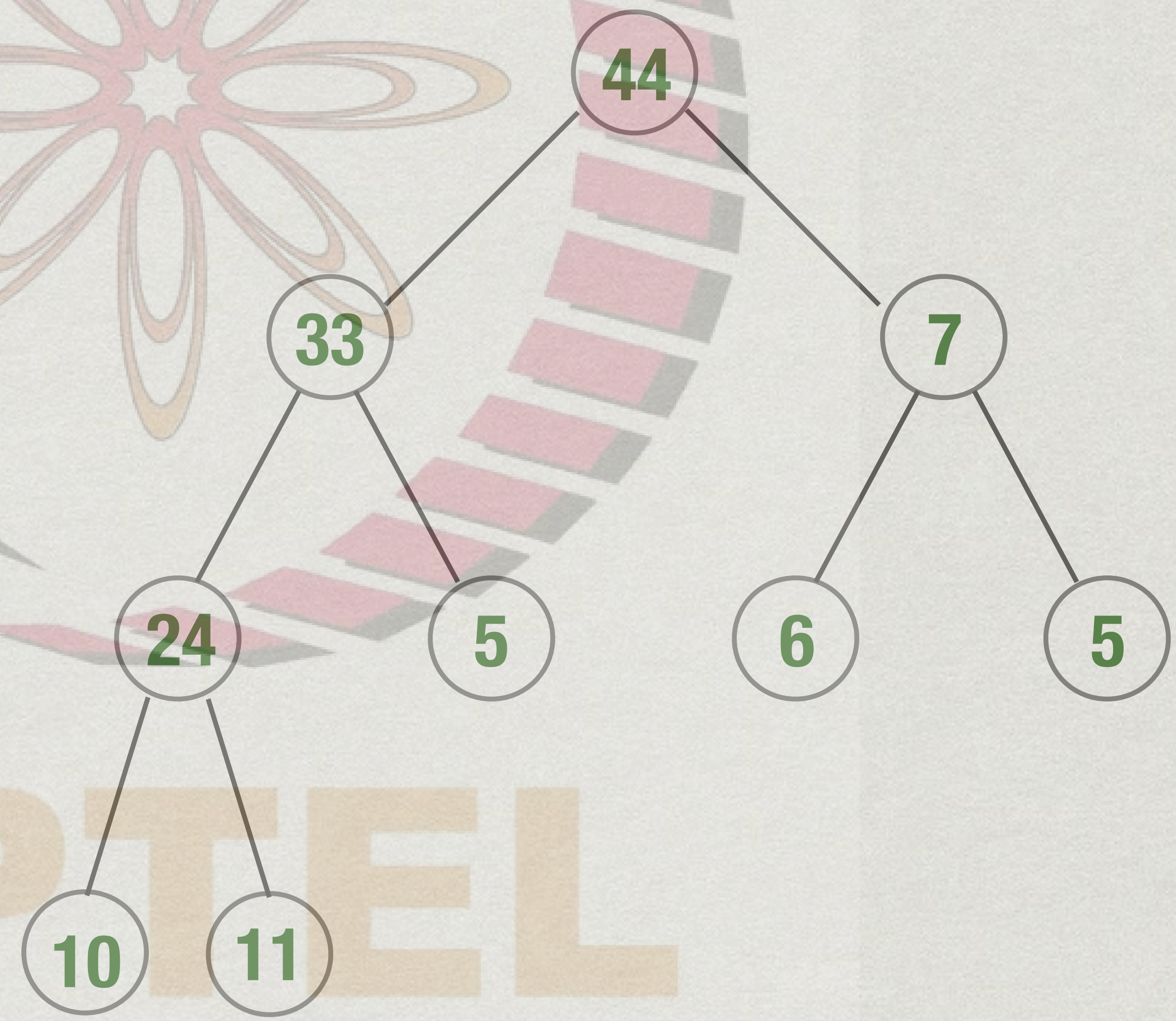
Updating values

- Change 12 to 44
  - Increasing a value can create heap violation with parent
  - Fix violations upwards, to root



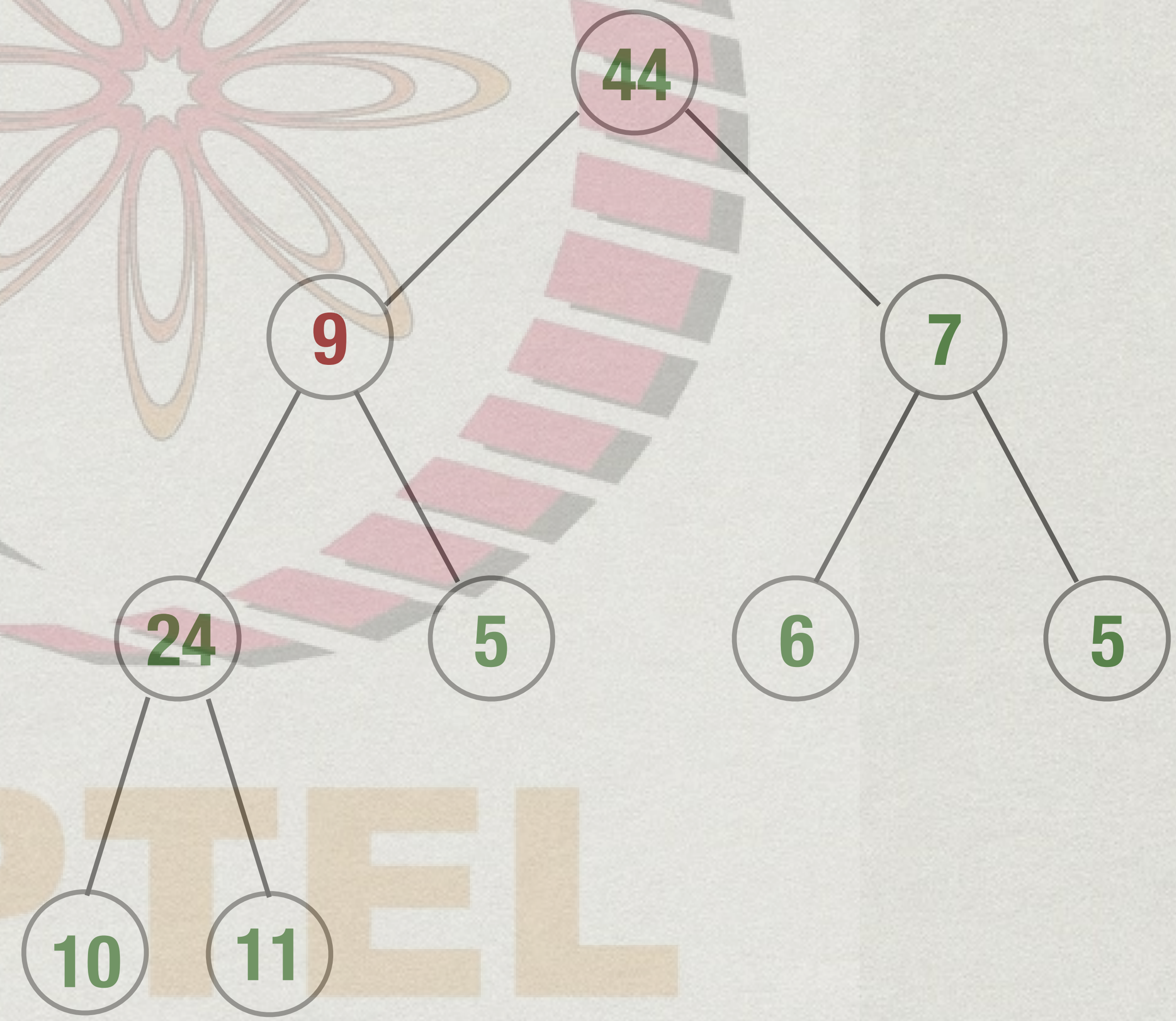
Updating values

- Change 33 to 9
  - Decreasing a value can create heap violation with children
  - Fix violations downwards, to leaves



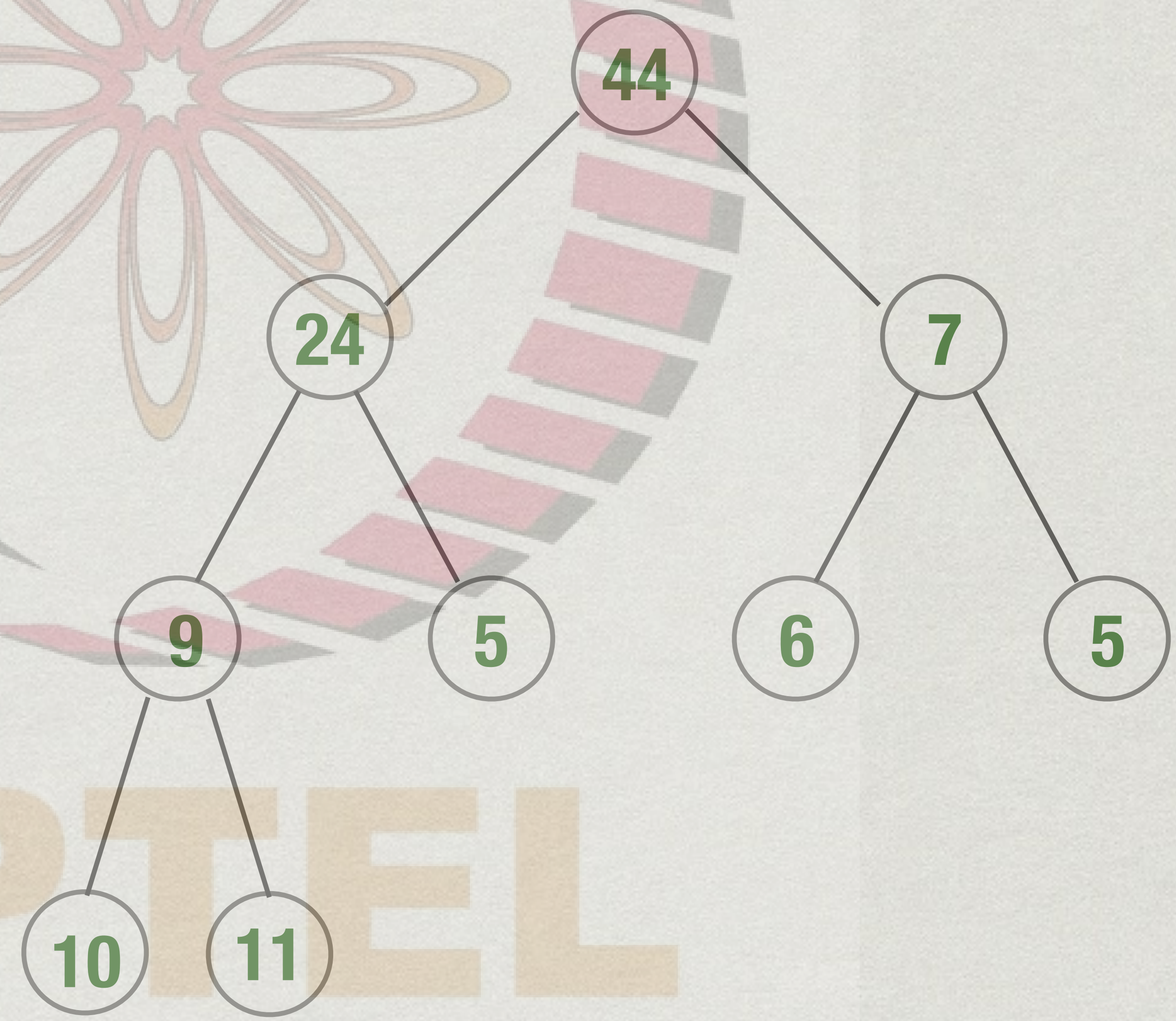
Updating values

- Change 33 to 9
  - Decreasing a value can create heap violation with children
  - Fix violations downwards, to leaves



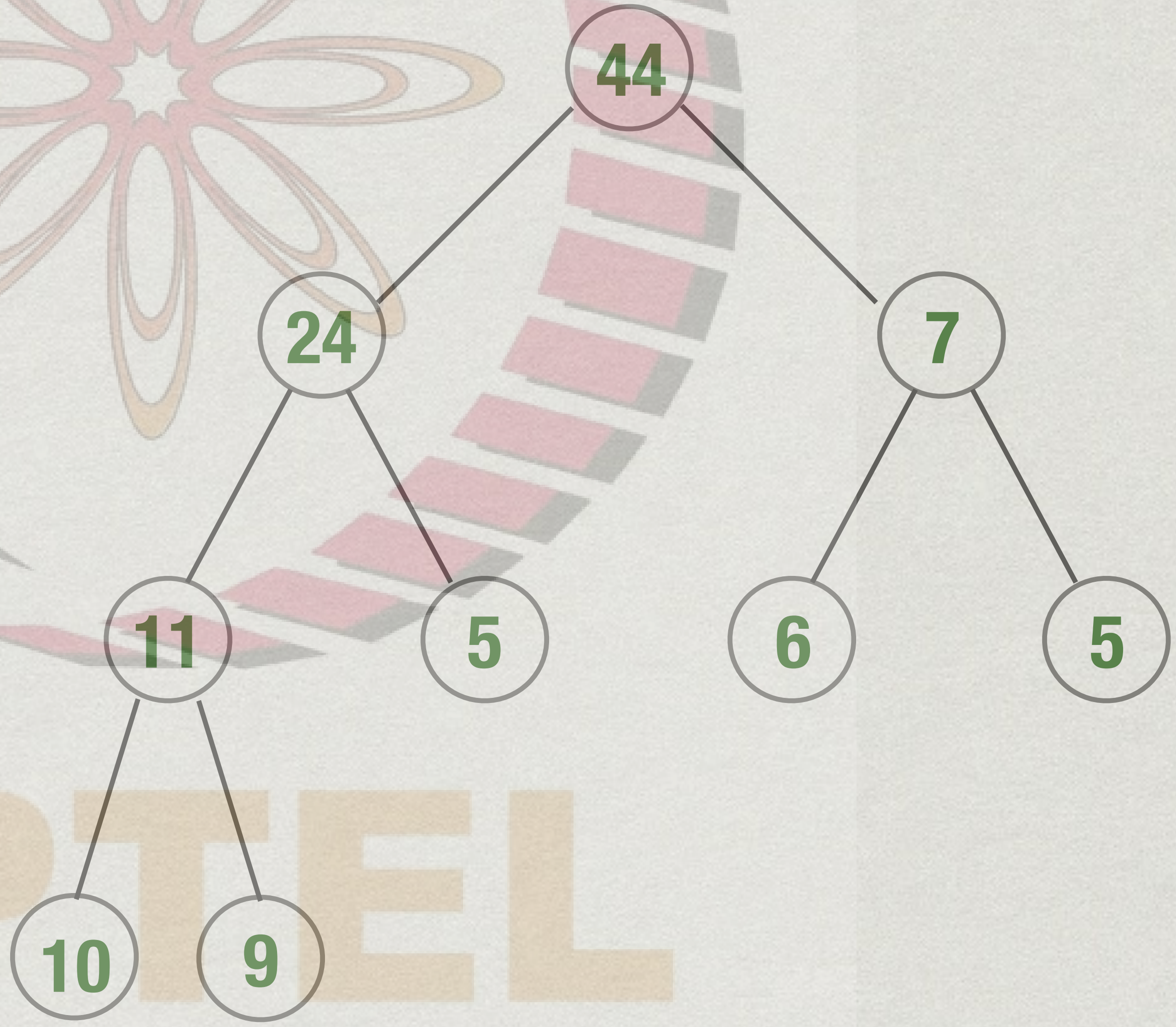
Updating values

- Change 33 to 9
    - Decreasing a value can create heap violation with children
    - Fix violations downwards, to leaves



Updating values

- Change 33 to 9
  - Decreasing a value can create heap violation with children
  - Fix violations downwards, to leaves



Updating values

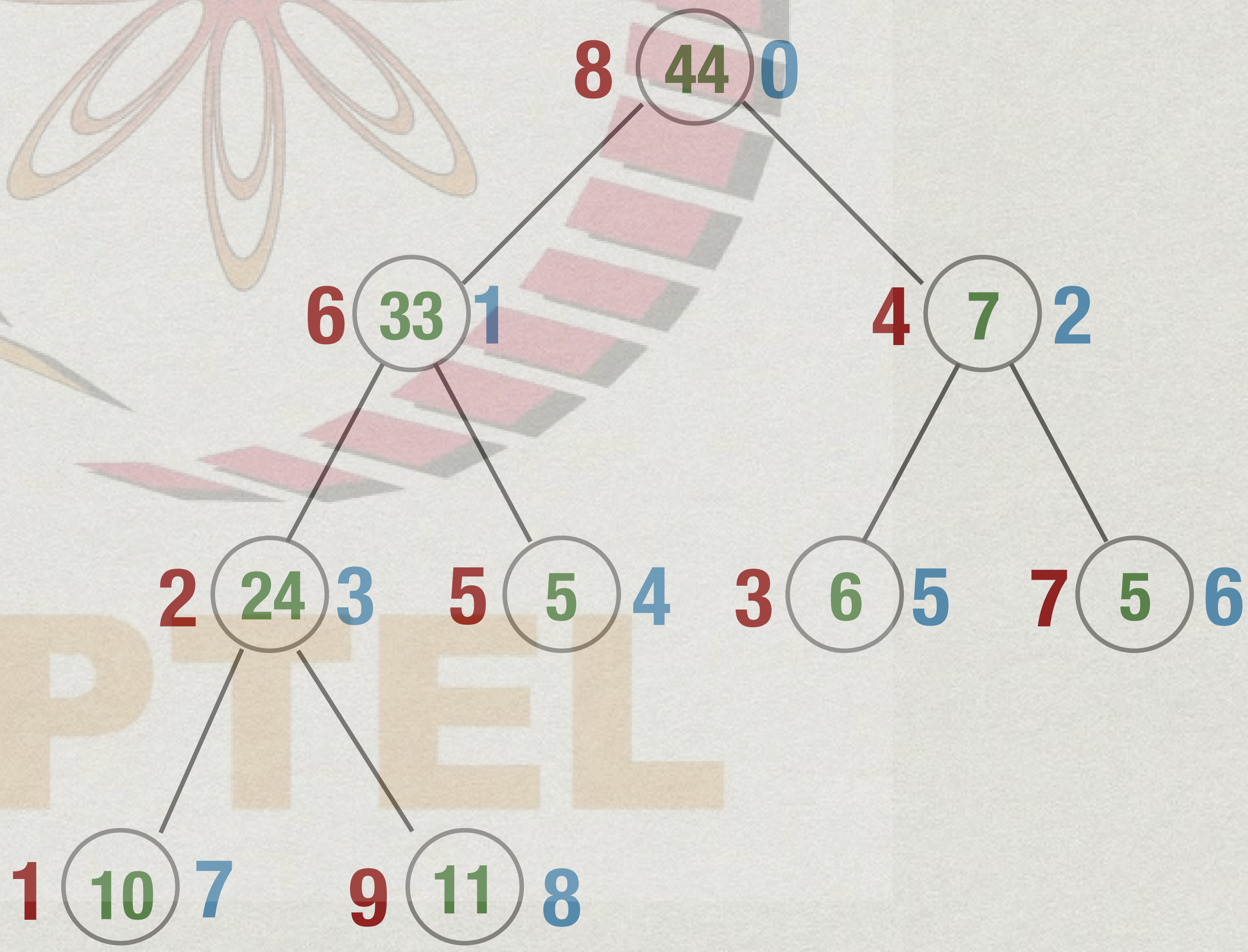
- Update Distance[j]
  - Where is Distance[j] in heap?
- Two additional arrays, NodeToHeap[ ], HeapToNode[ ]

NodeToHeap

1	2	3	4	5	6	7	8	9
7	3	5	2	4	1	6	0	8

HeapToNode

0	1	2	3	4	5	6	7	8
8	6	4	2	5	3	7	1	9



Updating values

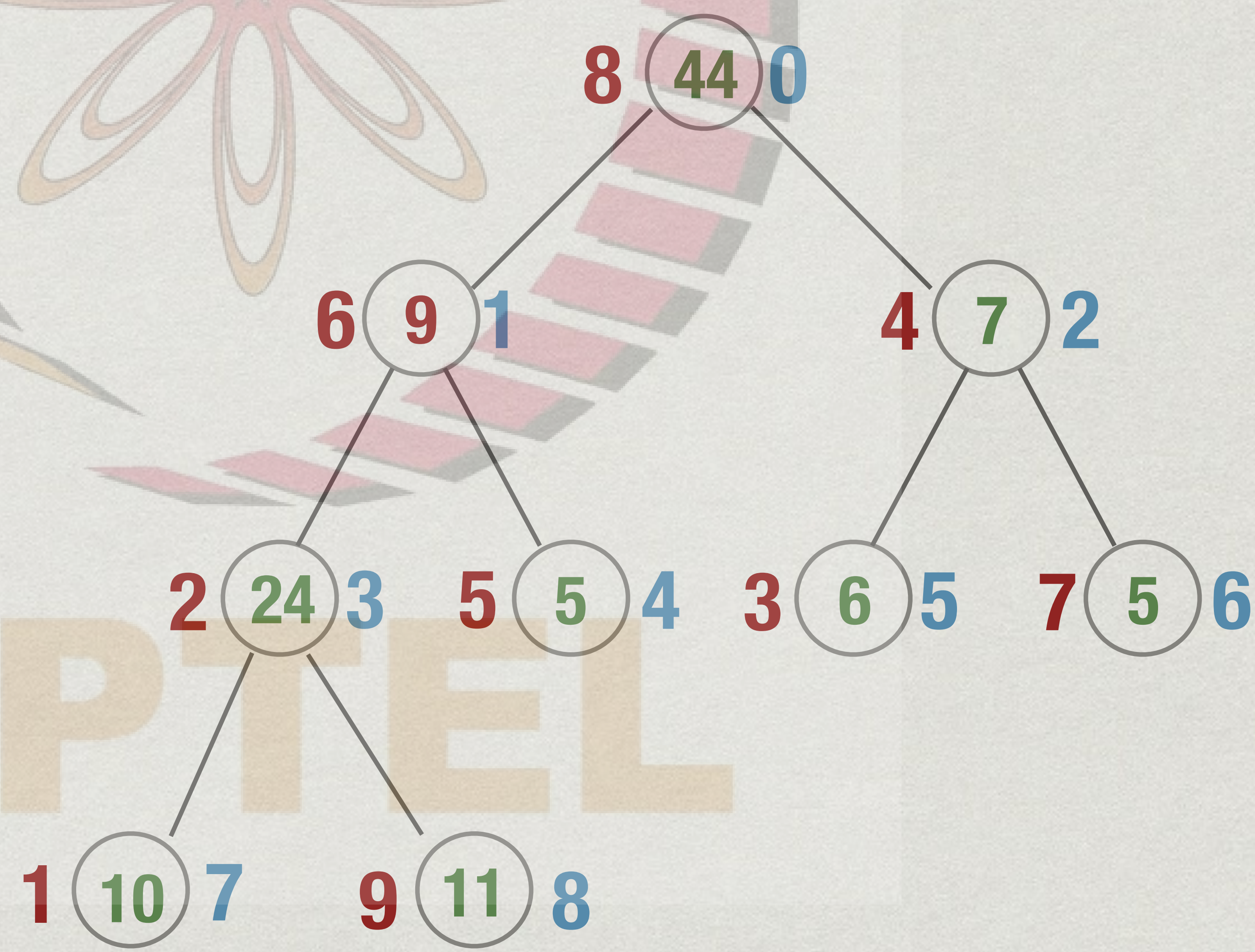
NodeToHeap

1	2	3	4	5	6	7	8	9
7	3	5	2	4	1	6	0	8

HeapToNode

0	1	2	3	4	5	6	7	8
8	6	4	2	5	3	7	1	9

- Update Distance[j]
  - Where is Distance[j] in heap?
- Two additional arrays, NodeToHeap[ ], HeapToNode[ ]



Updating values

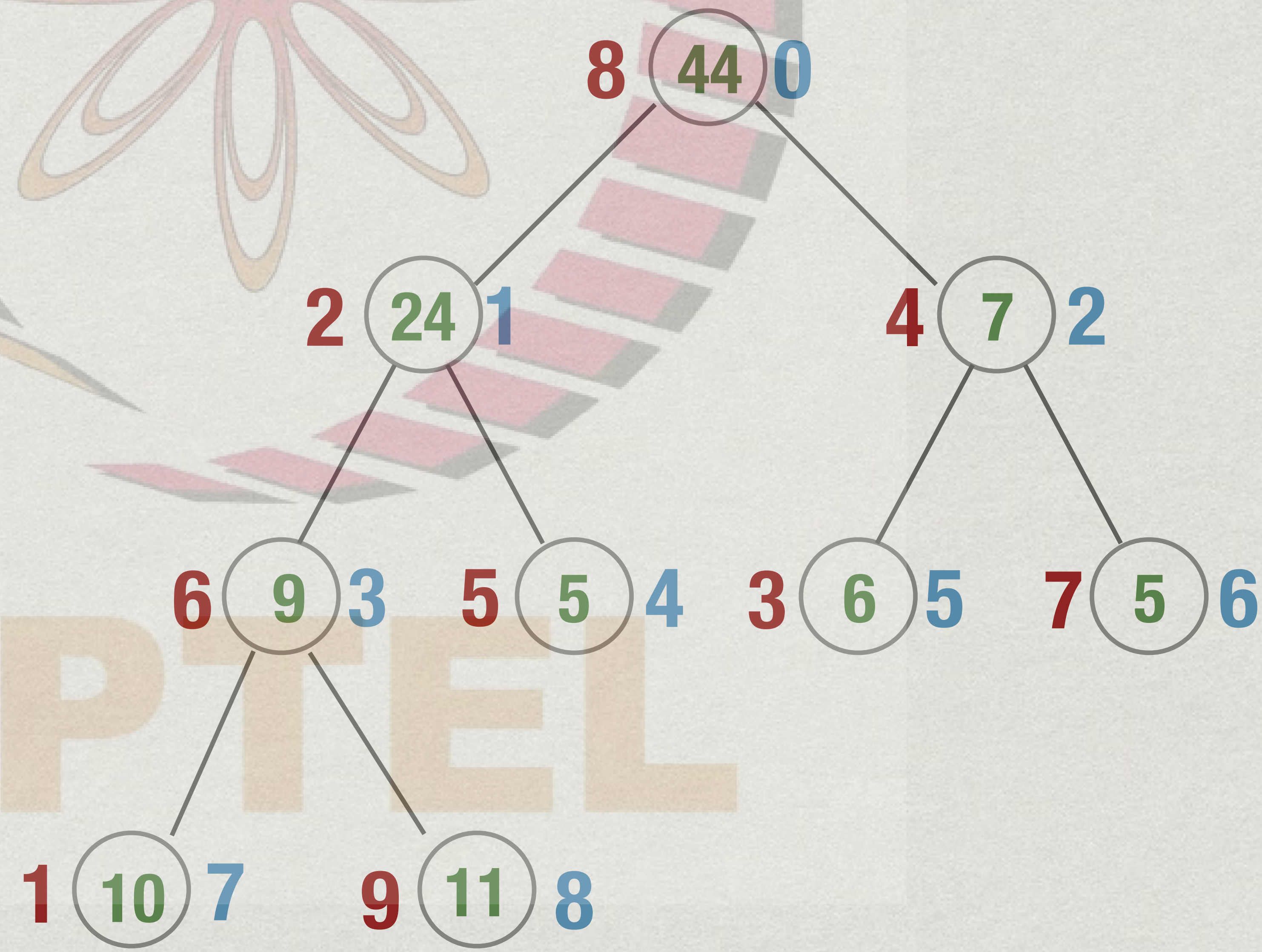
- Update Distance[j]
  - Where is Distance[j] in heap?
- Two additional arrays, NodeToHeap[ ], HeapToNode[ ]

NodeToHeap

1	2	3	4	5	6	7	8	9
7	1	5	2	4	3	6	0	8

HeapToNode

0	1	2	3	4	5	6	7	8
8	2	4	6	5	3	7	1	9



Updating values

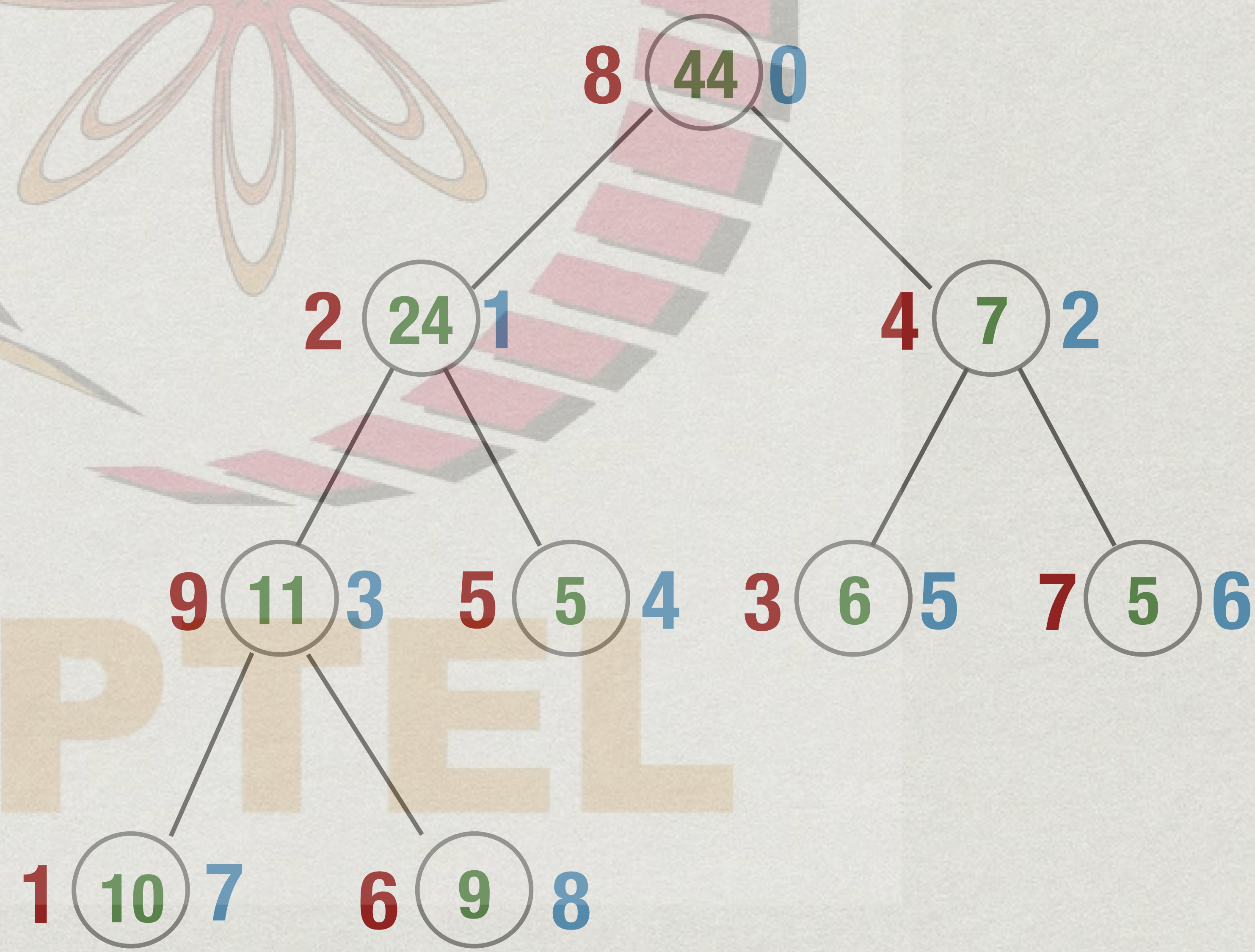
- Update Distance[j]
  - Where is Distance[j] in heap?
- Two additional arrays, NodeToHeap[ ], HeapToNode[ ]

NodeToHeap

1	2	3	4	5	6	7	8	9
7	1	5	2	4	8	6	0	3

HeapToNode

0	1	2	3	4	5	6	7	8
8	2	4	9	5	3	7	1	6



Dijkstra's algorithm: Complexity

- Using heaps with updates
  - Finding minimum burn time vertex takes O(log n)
  - With adjacency list, updating burn times take O(log n) each, total O(m) edges
- Overall O(n log n + m log n) = O((n+m) log n)
- Similar strategy works for Prim's algorithm for minimum cost spanning tree

Heap sort

- Start with an unordered list
- Build a heap — O(n)
- Call delete_max( ) n times to extract elements in descending order — O(n log n)
- After each delete_max( ), heap shrinks by 1
  - Store maximum value at the end of current heap
  - In place O(n log n) sort