eecs281 Data Structures and Algorithms

Discussion 4: Week of Sep 26, 2011

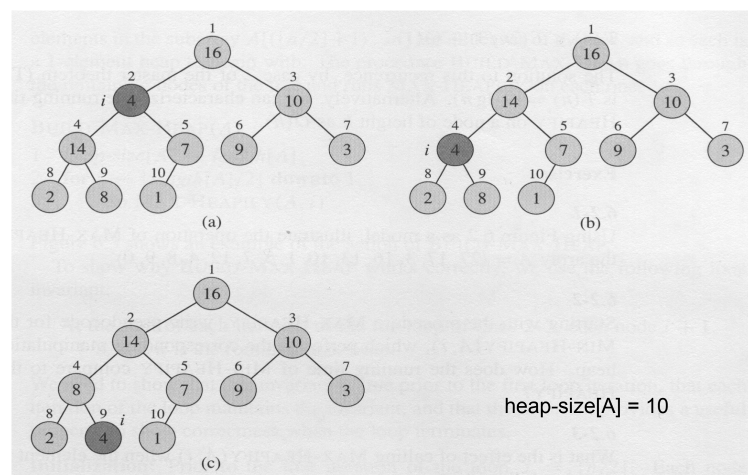
Outline

- Heaps
- Huffman Encoding
- AVL Trees

Heap Review

- Heapify
  - PercolateUp, TrickleDown
  - O(logN) time for both cases
- BuildHeap
  - O(N) time, proof is non-trivial
- DequeueMax
  - Replace root with last node, then fix down
  - O(logN) time

Max-Heapify(2)



Building a heap

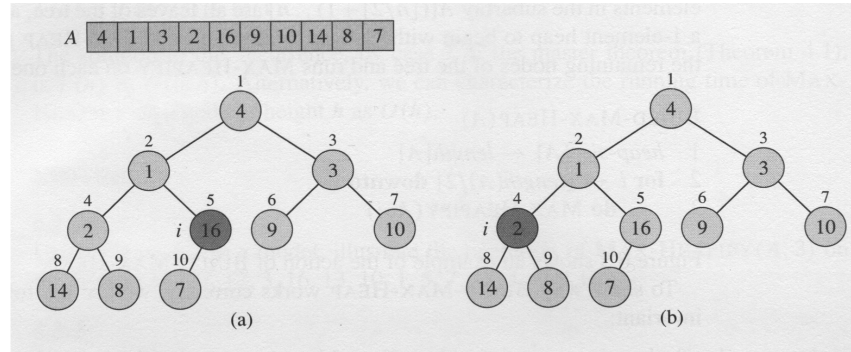
Build-Max-Heap(A)

heap-size[A] ← length[A]

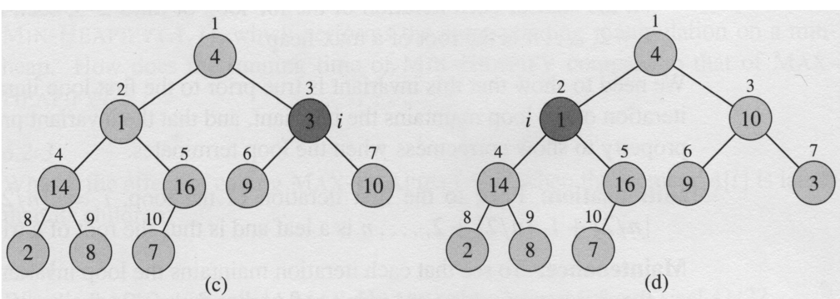
for i ← ⌊length[A]/2⌋ **downto** 1

do Max-Heapify(A, i)

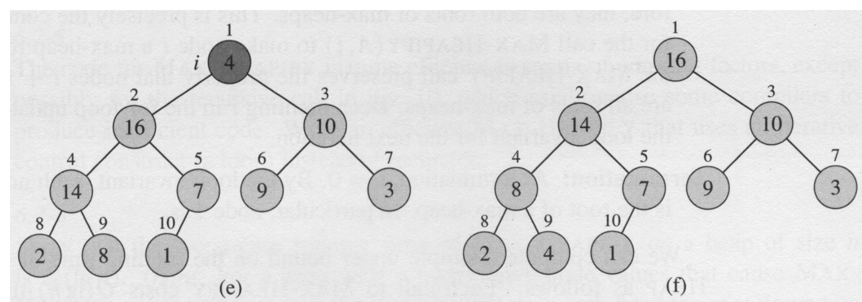
Building a heap



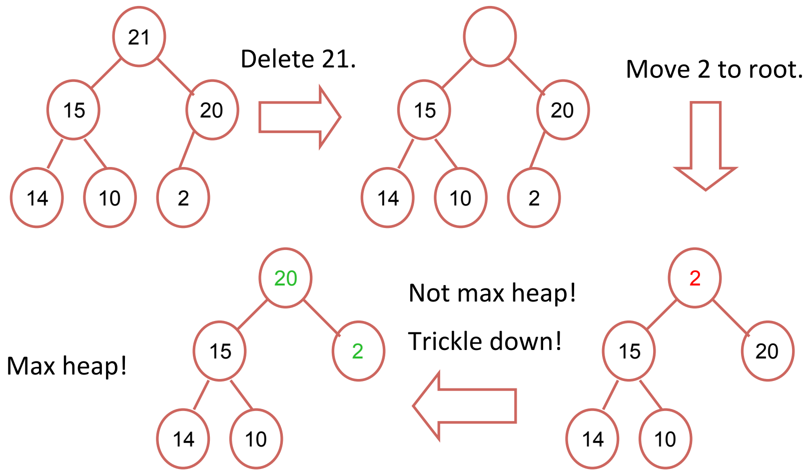
Building a heap



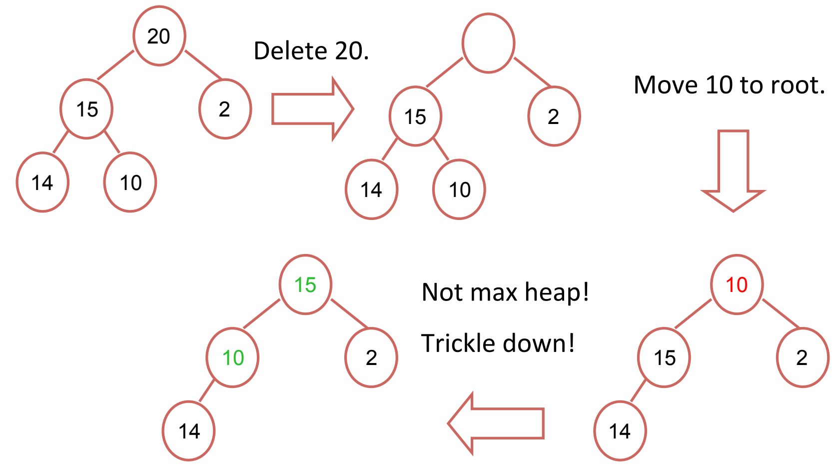
Building a heap



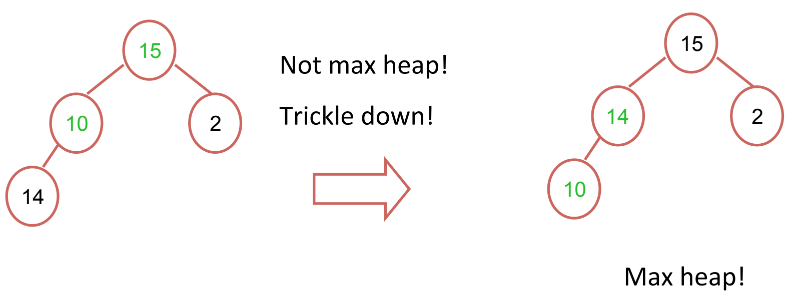
DequeueMax



DequeueMax



DequeueMax



Huffman Encoding

- Compresses data to use less space.
- Works best on text data, not binary data.
  - Why?

Huffman Encoding

- Compresses data to use less space.
- Works best on text data, not binary data.
  - Huffman encoding uses a character's frequency in order to encode things – the more frequent a character, the better the compression.
  - Binary data looks like: páW.ÈDŽOŸuæ`ç\Iß
  - Text data is relatively uniform and predictable.

Huffman Encoding

Given the text ALIBABA which of the following can be the Huffman prefix code for compressing this text?

A. A=001, B=000, I=01, L=10.

B. A=00, B=01, I=10, L=11.

C. A=0, B=11, I=001, L=000.

D. A=0, B=01, I=100, L=110.

E. A=1, B=01, I=001, L=000.

Huffman Encoding

- ALIBABA
  - A: 3
  - B: 2
  - L : 1
  - I : 1
- A=001, B=000, I=01, L=10.

Huffman Encoding

- ALIBABA
  - A: 3
  - B: 2
  - L : 1
  - I : 1
- A=001, B=000, I=01, L=10. No.

Huffman Encoding

- ALIBABA
  - A: 3
  - B: 2
  - L : 1
  - I : 1

- A=00, B=01, I=10, L=11.

Huffman Encoding

- ALIBABA
  - A: 3
  - B: 2
  - L : 1
  - I : 1

- A=00, B=01, I=10, L=11. No.

Huffman Encoding

- ALIBABA
  - A: 3
  - B: 2
  - L : 1
  - I : 1

- A=0, B=11, I=001, L=000.

Huffman Encoding

- ALIBABA
  - A: 3
  - B: 2
  - L : 1
  - I : 1

- A=0, B=11, I=001, L=000. Yes.

Huffman Encoding

- ALIBABA
  - A: 3
  - B: 2
  - L : 1
  - I : 1

- A=0, B=01, I=100, L=110

Huffman Encoding

- ALIBABA
  - A: 3
  - B: 2
  - L : 1
  - I : 1

- A=0, B=01, I=100, L=110. Yes.

Huffman Encoding

- ALIBABA
  - A: 3
  - B: 2
  - L : 1
  - I : 1

- A=1, B=01, I=001, L=000.

Huffman Encoding

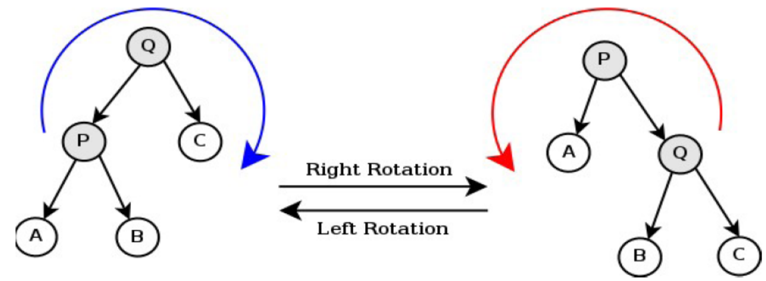
- ALIBABA
  - A: 3
  - B: 2
  - L : 1
  - I : 1

- A=1, B=01, I=001, L=000. Yes.

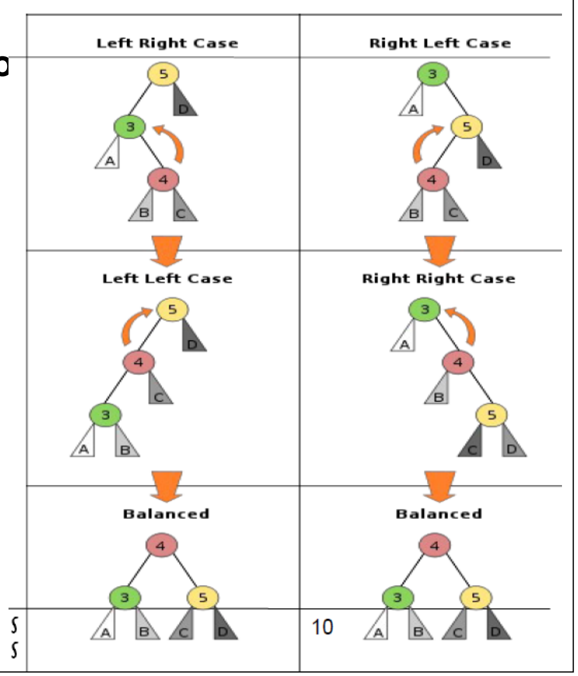
AVL Trees

- A self-balancing binary search tree
- Balance Factor
  - Height(left-sub-tree) – Height(right-sub-tree)
- Balance factor of -1, 0, or 1 for each node
- Rotations needed for maintain balance
  - Single Rotation (LL or RR)
  - Double Rotation(LR or RL)

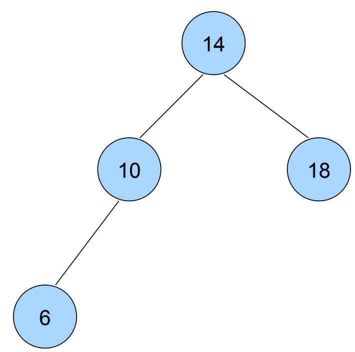
Single Rotation



Do

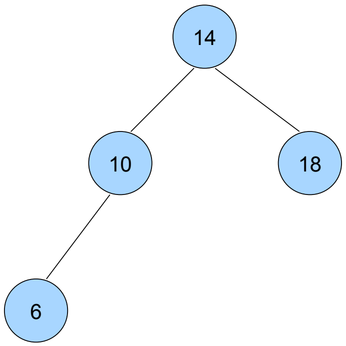


Example



- Is this AVL balanced?
- Why/why not?

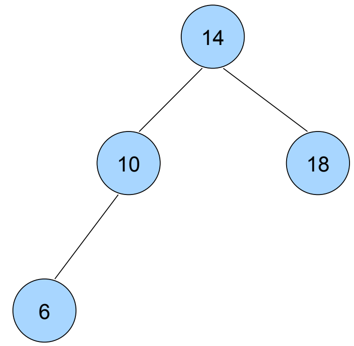
Example



- What happens when you add (one after the other):
  - 4
  - 12
  - 20

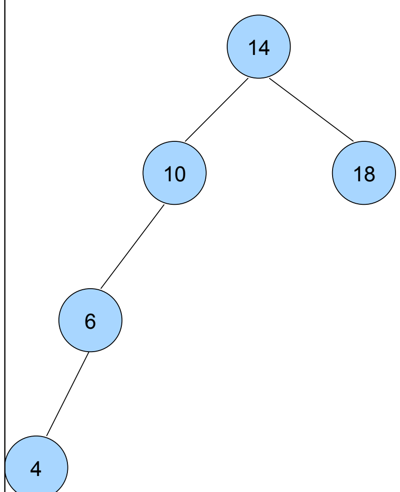
AVL Trees

- Adding 4



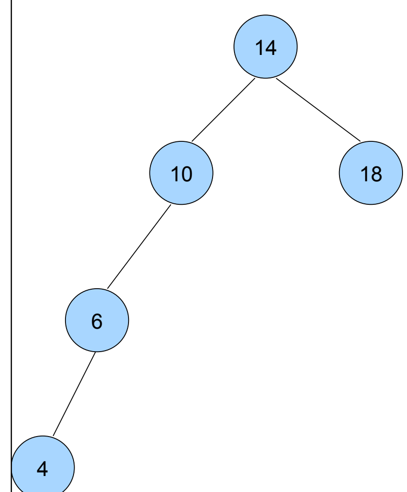
Example

- Adding 4

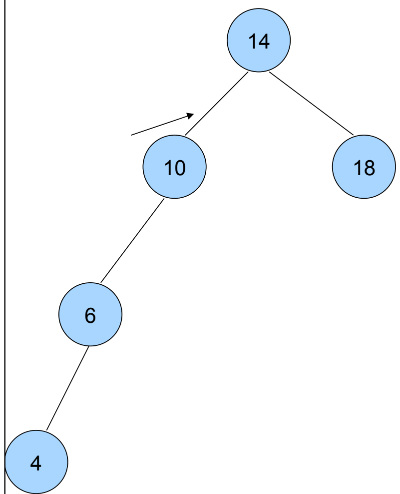


Example

- Adding 4
  - Not AVL balanced
  - What do we do?

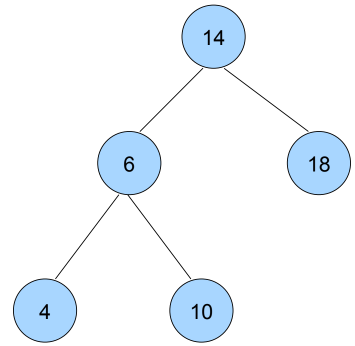


Example



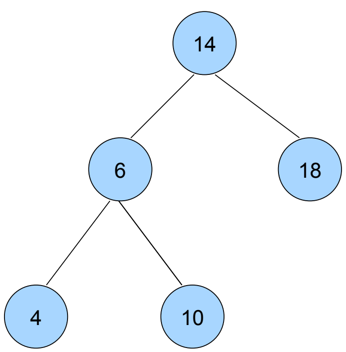
- Adding 4
  - Not AVL balanced
  - What do we do?
  - Right rotation around 10

Example



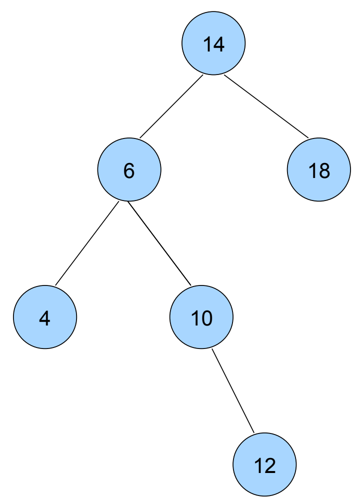
- Adding 4
  - Not AVL balanced
  - What do we do?
  - Right rotation around 10

Example



- Adding 12

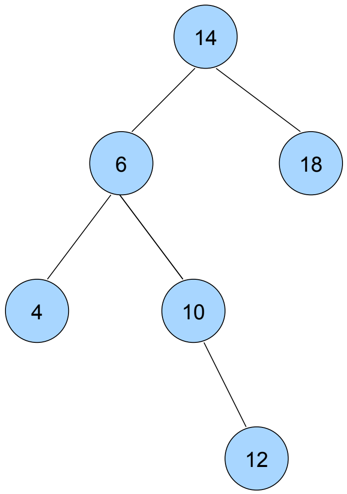
Example



- Adding 12

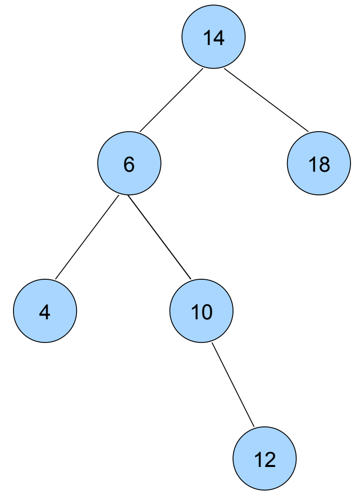
Example

- Adding 12
  - Not AVL balanced
  - What do we do?



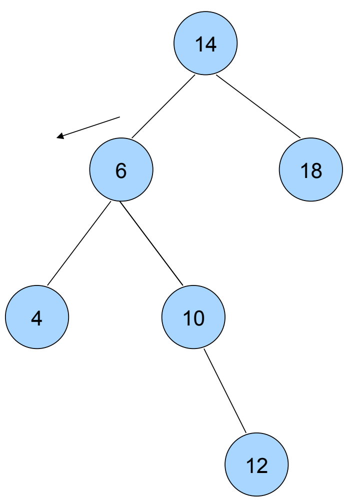
Example

- Adding 12
  - Not AVL balanced
  - What do we do?
  - Double rotation



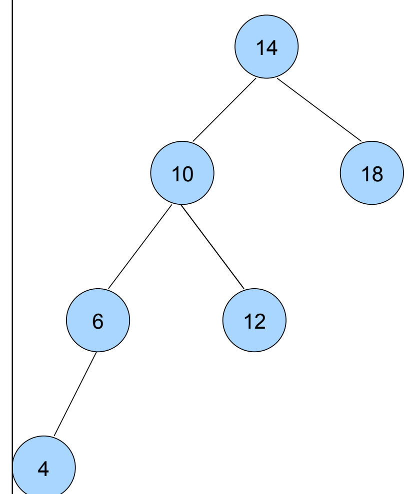
Example

- Adding 12
  - Not AVL balanced
  - What do we do?
  - Double rotation
  - Left rotation around 6



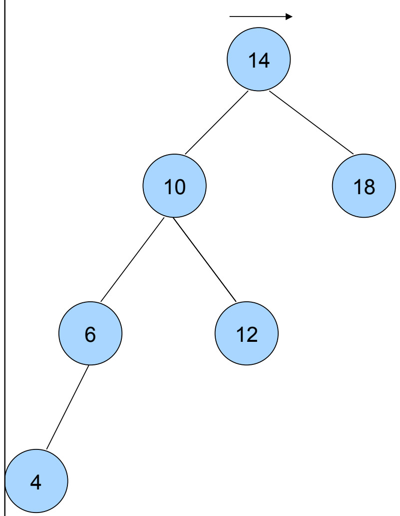
Example

- Adding 12
  - Not AVL balanced
  - What do we do?
  - Double rotation
  - Left rotation around 6



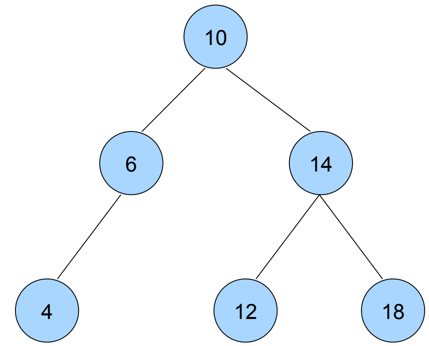
Example

- Adding 12
  - Not AVL balanced
  - What do we do?
  - Double rotation
  - Left rotation around 6
  - Right rotation around 14



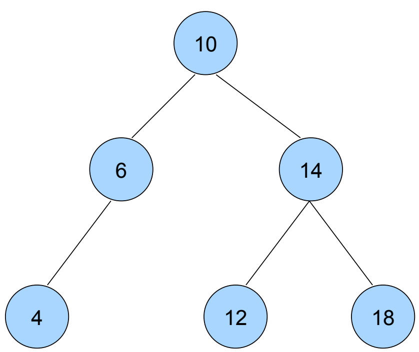
Example

- Adding 12
  - Not AVL balanced
  - What do we do?
  - Double rotation
  - Left rotation around 6
  - Right rotation around 14



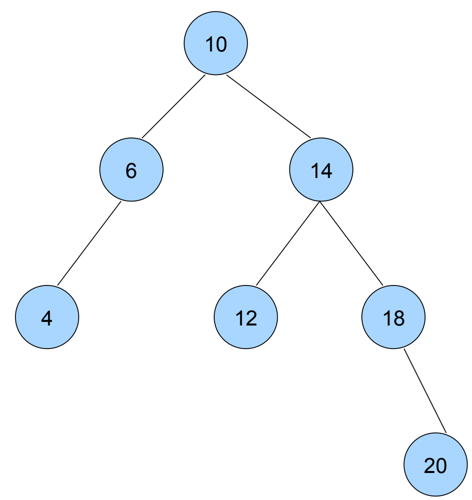
Example

- Adding 20



Example

- Adding 20



Example

- Adding 20
  - Still AVL Balanced