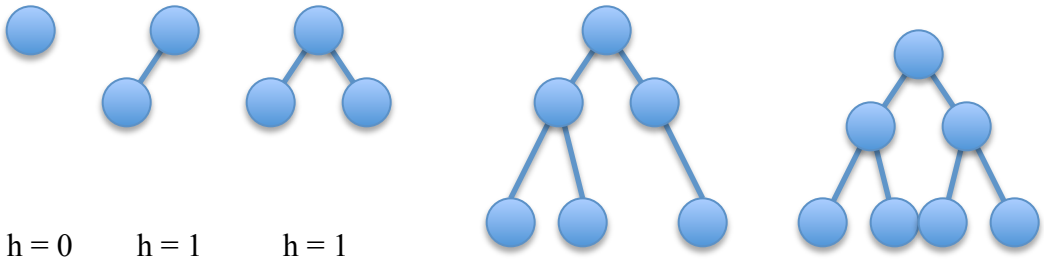
Adam Jablonski

CS260

1. A binary tree has *n* nodes. What is the maximum height of the tree? What is the minimum height? Explain your answers.

n = 1 n = 2 n = 3 ….. n = 6 n = 7



h = 0 h = 1 h = 1 h = 2 h = 2

Based on observation the minimum height and max height have a direct correlation to the number of nodes. It's a binary tree and recursive therefore the base is 2.

Maximum:

$$\sum_{i=0}^{h} 2^h$$

Based on Definition:

$$2^0 + 2^1 + 2^2 + 2^3 + \ldots\ldots + 2^{h-1} = 2^{h+1} - 1$$

Maximum is then 2^h simply by observation.

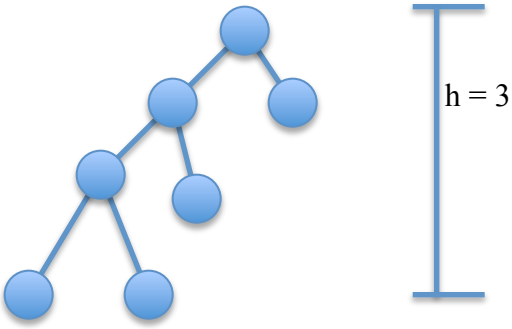
The final equation is: $2^{h+1} - 1 \leq n \leq 2^h$

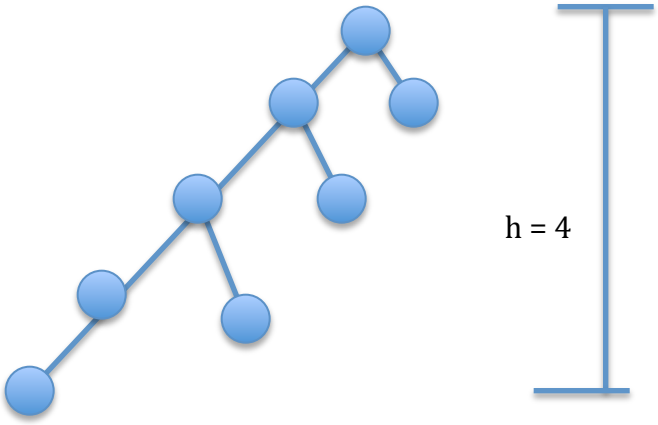
Exponential to log conversions:

$y = b^x$ and $x = \log_b y$

Therefore the minimum height for even is: $h = 2^{h+1} - 1 = \log_2 n$

Therefore the minimum height for odd is: $h = 2^h = \log_2 (n+1) - 1$





n = 7

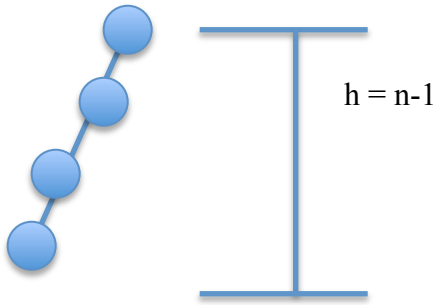
n = 8

The maximum height is : h = n/2

The maximum height : (h-1)/2

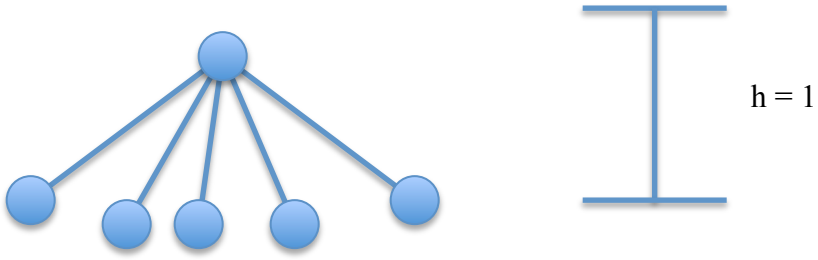
2. A tree (not necessarily a binary tree) has *n* nodes. What are the minimum and maximum possible heights of the tree? Explain your answers.

This is a general tree therefore is not restricted to a binary trees rules. Because of this the max height can be all nodes in a row (see below)



The height is n-1 because the tree root is considered 0.

The minimum height would be 1. See figure below.



3. Give the big-O running time of each of the following set operations:

Sorted:

MAKENULL: O(1)

UNION: O(n)

INTERSECTION: O(n)

MEMBER: O(n)

MIN: O(1)

INSERT: O(n)

DELETE: O(n)

Open Hash Table:

MAKENULL: O(1)

UNION: O(n)

INTERSECTION: O(n)

MEMBER: O(1)

MIN: O(1)

INSERT: O(1)

DELETE: O(1)

4. Problem **4.7** from the text

   h(i) = i mod 7 on list [1, 8, 27, 64, 125, 216, 343]

   1 mod 7 -> 1 8 mod 7 -> 1 27 mod 7 -> 2 64 mod 7 -> 1

   125 mod 7 -> 2 216 mod 7 -> 2 343 mod 7 -> 0

   a. Open Hash Table:

0	1	2	3	4	5	6
343	1	27				
	8	125				
	64	216				

   b. Closed Hash Table with linear resolution:

0	125
1	1
2	8
3	64
4	216
5	343
6	27

5. Explain why the following hash functions are not very good:
    a. Hash keys are character strings. The hash function h1(x) computes the length of the string.
    b. The hash function h2(x) computes a random number r with $1 \leq r \leq B$, where B is the number of buckets.

Worse Case Scenario Analysis:

a. There are a few problems with using the length of the string as a hash.
    a. If all the strings are the same length the hash table is useless all the answers are in the same bucket. Runtime O(n).
    b. If all the strings are different lengths then once again its useless, it would be the same as putting them in a list. Runtime O(n).
    c. On a designers note what if you had cat, bat, sat, rat, rhino, dishes. Rhino and dishes are in the same bucket and cat, bat, sat, and rat are in the same bucket. The buckets aren't organized by anything, it would make more sense to put all the animals in one bucket.

b. There are a few problems with using a random number generation for a hash.
    a. Whatever the odds are, what happens if the same random number is chosen every time. Its basically just a list at that point since each answer is in a different bucket. Its has a runtime of O(n).
    b. What if a different number is chosen each time. At that point all the answers are put in the same bucket and if the collision is done right it's a list again. The runtime is O(n).
    c. A designers chose to have some relation between the items in the hash bucket. As seen above.

6. Design an efficient data structure for representing a subset S of the integers from 1 to n. The operations that must be supported are:

list = [1, 3, 5]

1	2	3	4	5
1	0	1	0	1

list = [1, 2, 4]

1	2	3	4	5
1	1	0	1	0

By using the encoding scheme of 1 meaning its in the list and 0 if its not the run time for deletion and insertion is O(1). It works by looking up list[1] and if its 1 that means the item is there and if its 0 that means its not. The look up is constant time making the selecting and adding constant.