HW 4

Tuesday, April 6, 2021 3:53 PM

CSCE 221-200: Honors Data Structures and Algorithms
Assignment Cover Page
Spring 2021

Name:	Alexander Akomer
Email:	alexakomer@tamu.edu
Assignment:	HW 4
Grade (filled in by grader):	

Please list below all sources (people, books, webpages, etc) consulted regarding this assignment (use the back if necessary):

CSCE 221 Students	Other People	Printed Material	Web Material (give URL)	Other Sources
1.	1.	1.	1. https://www.geeksforgeeks.org/insertion-and-deletion-in-heaps/	1.
2.	2.	2.	2.	2.
3.	3.	3.	3.	3.

Recall that TAMU Student Rules define academic misconduct to include acquiring answers from any unauthorized source, working with another person when not specifically permitted, observing the work of other students during any exam, providing answers when not specifically authorized to do so, informing any person of the contents of an exam prior to the exam, and failing to credit sources used. *Disciplinary actions range from grade penalty to expulsion.*

"On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work. In particular, I certify that I have listed above all the sources that I consulted regarding this assignment, and that I have not received or given any assistance that is contrary to the letter or the spirit of the collaboration guidelines for this assignment."

Signature:	Alexander Akomer
Date:	04/06/2021

Draw the 11-entry hash table that results from using the hash function h(x) = (3x+5) mod 11 when you are given the following keys, in the given order:
12, 44, 13, 88, 23, 94, 11, 39, 20, 15, 5.

Exercise D: Assume collisions are handled by linear probing.

h(12) = (3 x 12 + 5) mod 11 = 8
h(44) = (3 x 44 + 5) mod 11 = 5
h(13) = (3 x 13 + 5) mod 11 = 0
h(88) = (3 x 88 + 5) mod 11 = 5 (collision)
    (h(88) + f(1)) mod 11 = 11 (5 + 1) mod 11 = 6
h(23) = (3 x 23 + 5) mod 11 = 8 (collision)
    (h(23) + f(1)) mod 11 = (8 + 1) mod 11 = 9
h(94) = (3 x 94 + 5) mod 11 = 1
h(11) = (3 x 11 + 5) mod 11 = 5 (collision)
    (h(11) + f(1)) mod 11 = (5 + 1) mod 11 = 6 (collision)
    (h(11) + f(2)) mod 11 = (5 + 2) mod 11 = 7
h(39) = (3 x 39 + 5) mod 11 = 1 (collision)
    (h(39) + f(1)) mod 11 = (1 + 1) mod 11 = 2
h(20) = (3 x 20 + 5) mod 11 = 10
h(15) = (3 x 15 + 5) mod 11 = 6 (collision)
    (h(15) + f(1)) mod 11 = (6 + 1) mod 11 = 7 (collision)
    (h(16) + f(2)) mod 11 = (6 + 2) mod 11 = 8 (collision)
    (h(16) + f(3)) mod 11 = (6 + 3) mod 11 = 9 (collision)
    (h(16) + f(4)) mod 11 = (6 + 4) mod 11 = 10 (collision)
    (h(16) + f(5)) mod 11 = (6 + 5) mod 11 = 0 (collision)
    (h(16) + f(6)) mod 11 = (6 + 6) mod 11 = 1 (collision)
    (h(16) + f(7)) mod 11 = (6 + 7) mod 11 = 2 (collision)
    (h(16) + f(8)) mod 11 = (6 + 8) mod 11 = 3
h(5) = (3 x 5 + 5) mod 11 = 9 (collision)
    (h(5) + f(1)) mod 11 = (9 + 1) mod 11 = 10 (collision)
    (h(5) + f(2)) mod 11 = (9 + 2) mod 11 = 0 (collision)
    (h(5) + f(3)) mod 11 = (9 + 3) mod 11 = 1 (collision)
    (h(5) + f(4)) mod 11 = (9 + 4) mod 11 = 2 (collision)
    (h(5) + f(5)) mod 11 = (9 + 5) mod 11 = 3 (collision)
    (h(5) + f(6)) mod 11 = (9 + 6) mod 11 = 4

Index	**Keys**
0	13
1	94
2	39
3	15
4	5
5	44
6	88
7	11
8	12
9	23
10	20

Draw the 11-entry hash table that results from using the hash function h(x) = (3x+5) mod 11 when you are given the following keys, in the given order:
12, 44, 13, 88, 23, 94, 11, 39, 20, 15, 5.

Exercise E: Assume collisions are handled by quadratic probing.

h(12) = (3 x 12 + 5) mod 11 = 8
h(44) = (3 x 44 + 5) mod 11 = 5
h(13) = (3 x 13 + 5) mod 11 = 0
h(88) = (3 x 88 + 5) mod 11 = 5 (collision)
(h(88) + f(1)) mod 11 = 11 (5 + 1) mod 11 = 6
h(23) = (3 x 23 + 5) mod 11 = 8 (collision)
(h(23) + f(1)) mod 11 = (8 + 1) mod 11 = 9
h(94) = (3 x 94 + 5) mod 11 = 1
h(11) = (3 x 11 + 5) mod 11 = 5 (collision)
(h(11) + f(1)) mod 11 = (5 + 1) mod 11 = 6 (collision)
(h(11) + f(2)) mod 11 = (5 + 4) mod 11 = 9 (collision)
(h(11) + f(3)) mod 11 = (5 + 9) mod 11 = 3
h(39) = (3 x 39 + 5) mod 11 = 1 (collision)
(h(39) + f(1)) mod 11 = (1 + 1) mod 11 = 2
h(20) = (3 x 20 + 5) mod 11 = 10
h(15) = (3 x 15 + 5) mod 11 = 6 (collision)
(h(16) + f(1)) mod 11 = (6 + 1) mod 11 = 7
h(5) = (3 x 5 + 5) mod 11 = 9 (collision)
(h(5) + f(1)) mod 11 = (9 + 1) mod 11 = 10 (collision)
(h(5) + f(2)) mod 11 = (9 + 4) mod 11 = 2 (collision)
(h(5) + f(3)) mod 11 = (9 + 9) mod 11 = 7 (collision)
(h(5) + f(4)) mod 11 = (9 + 16) mod 11 = 3 (collision)
(h(5) + f(5)) mod 11 = (9 + 25) mod 11 = 1 (collision)
(h(5) + f(6)) mod 11 = (9 + 36) mod 11 = 1 (collision)
(h(5) + f(7)) mod 11 = (9 + 49) mod 11 = 3 (collision)
(h(5) + f(8)) mod 11 = (9 + 64) mod 11 = 7 (collision)
(h(5) + f(9)) mod 11 = (9 + 81) mod 11 = 2 (collision)
(h(5) + f(10)) mod 11 = (9 + 100) mod 11 = 10 (collision)
(h(5) + f(11)) mod 11 = (9 + 121) mod 11 = 9 (collision)
(h(5) + f(12)) mod 11 = (9 + 144) mod 11 = 10 (collision)
(h(5) + f(13)) mod 11 = (9 + 169) mod 11 = 2 (collision)
(h(5) + f(14)) mod 11 = (9 + 196) mod 11 = 7 (collision)
(h(5) + f(15)) mod 11 = (9 + 225) mod 11 = 3 (collision)
(h(5) + f(16)) mod 11 = (9 + 256) mod 11 = 1 (collision)
(h(5) + f(17)) mod 11 = (9 + 289) mod 11 = 1 (collision)
(h(5) + f(18)) mod 11 = (9 + 324) mod 11 = 3 (collision)
(h(5) + f(19)) mod 11 = (9 + 361) mod 11 = 7 (collision)
(h(5) +(20)) mod 11 = (9 + 300) mod 11 = 1 (collision)
(h(5) + f(21)) mod 11 = (9 + 441) mod 11 = 10 (collision)
(h(5) + f(22)) mod 11 = (9 + 484) mod 11 = 9 (collision)
(h(5) + f(23)) mod 11 = (9 + 529) mod 11 = 10 (collision)

Index	Keys
0	13
1	94

2	39
3	11
4	
5	44
6	88
7	15
8	12
9	23
10	20

(Quadratic probing failed to insert the key 5 into the hash table)

Draw the 11-entry hash table that results from using the hash function h(x) = (3x+5) mod 11 when you are given the following keys, in the given order:
12, 44, 13, 88, 23, 94, 11, 39, 20, 15, 5.

Exercise F: Assume collisions are handled by double hashing using the secondary function h'(x) = (x mod 7).

h(12) = (3 x 12 + 5) mod 11 = 8
h(44) = (3 x 44 + 5) mod 11 = 5
h(13) = (3 x 13 + 5) mod 11 = 0
h(88) = (3 x 88 + 5) mod 11 = 5 (collision)
(h(88) + 1(7 - (88 mod 7)) mod 11 = (5 + 3) mod 11 = 8 (collision)
(h(88) + 2(7 - (88 mod 7)) mod 11 = (5 + 6) mod 11 = 0 (collision)
(h(88) + 3(7 - (88 mod 7)) mod 11 = (5 + 9) mod 11 = 3
h(23) = (3 x 23 + 5) mod 11 = 8 (collision)
(h(23) + 1(7 - (23 mod 7)) mod 11 =(8 + 5) mod 11 = 2
h(94) = (3 x 94 + 5) mod 11 = 1
h(11) = (3 x 11 + 5) mod 11 = 5 (collision)
(h(11) + 1(7 - (11 mod 7)) mod 11 = (5 + 3) mod 11 = 8 (collision)
(h(11) + 2(7 - (11 mod 7)) mod 11 = (5 + 6) mod 11 = 0 (collision)
(h(11) + 3(7 - (11 mod 7)) mod 11 = (5 + 9) mod 11 = 3 (collision)
(h(11) + 4(7 - (11 mod 7)) mod 11 = (5 + 12) mod 11 = 6
h(39) = (3 x 39 + 5) mod 11 = 1 (collision)
(h(39) + 1(7 - (39 mod 7)) mod 11 = (1 + 3) mod 11 = 4
h(20) = (3 x 20 + 5) mod 11 = 10
h(15) = (3 x 16 + 5) mod 11 = 6 (collision)
(h(15) + 1(7 - (15 mod 7)) mod 11 = 11 (6 + 6) mod 11 = 1 (collision)
(h(15) + 2(7 - (15 mod 7)) mod 11 = 11 (6 + 12) mod 11 = 7
h(5) = (3 x 5 + 5) mod 11 = 9

Index	**Keys**
0	13
1	94
2	23
3	88

4	39
5	44
6	11
7	15
8	12
9	5
10	20

Exercise A: (10 pts) Redo the perfect hashing example in the lecture notes but with the set of keys {10, 22, 37, 40, 52, 60, 70, 72} and prime 73. Be sure to explain clearly how you chose N, p, a, b, and the second-level hash functions.

N = 8 (because the table size directly matches the number of keys given, which is a property of perfect hash tables).

p = 73 (given, qualifies because it is larger than any key)

a = 3 (random attempt, values were distributed well so no need to try again)

b = 41 (which is the midpoint of the range)

Therefore the top level hash function is h(x) = ((3*x + 41) mod 73) mod 8.

((3 * 10 + 41) mod 73) mod 8 = 7
((3 * 22 + 41) mod 73) mod 8 = 2
((3 * 37 + 41) mod 73) mod 8 = 6
((3 * 40 + 41) mod 73) mod 8 = 7
((3 * 52 + 41) mod 73) mod 8 = 3
((3 * 60 + 41) mod 73) mod 8 = 2
((3 * 70 + 41) mod 73) mod 8 = 0
((3 * 72 + 41) mod 73) mod 8 = 6

Main Hash table

Index	**Keys**
0	
1	
2	22, 60
3	52
4	
5	
6	37, 72
7	10, 40

Secondary hash table for T[3] have size 1^2 = 1
and the hash function h_3 (x) = ((1*x + 0) mod 73) mod 1

Secondary hash tables for T[2], T[6], T[7] have size 2^2 = 4
and the hash function h_2 (x) = h_6 (x) = h_7 (x) = ((5*x + 3) mod 73) mod 4

A random set of a and b values were attempted to reach a secondary hash function that hashed every value separately. As predicted by the probability in the notes, it took me only two or three attempts of random a and b values to determine a secondary hash function that worked for every value that needed to be rehashed.

The previous p value was maintained. N was reassigned to the size of the elements of the secondary hash table, squared.

((1 * 52 + 0) mod 73) mod 1 = 0

T[3] Secondary Hash table

Index	**Keys**
0	52

((13 * 22 + 3) mod 73) mod 4 = 2
((13* 60 + 3) mod 73) mod 4 = 1

T[2] Secondary Hash table

Index	**Keys**
0	
1	60
2	22
3	

((13 * 37 + 3) mod 73) mod 4 = 2
((13 * 72 + 3) mod 73) mod 4 = 3

T[6] Secondary Hash table

Index	**Keys**
0	
1	
2	37
3	72

((13 * 53 + 3) mod 73) mod 4 = 3
((13 * 40 + 3) mod 73) mod 4 = 0

T[7] Secondary Hash table

Index	**Keys**
0	40
1	

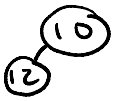
2	
3	53

Exercise 6.2:
a. Show the result of inserting 10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 7, 4, 11, 13, and 2, one at a time, into an initially empty binary heap.
b. Show the result of using the linear-time algorithm to build a binary heap using the same input.

a)

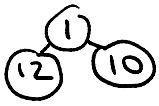
Iteration 1:

Iteration 2:

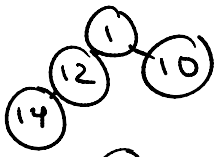


Iteration 3:

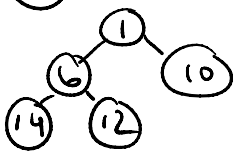
Iteration 4:



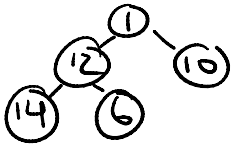
Iteration 5:



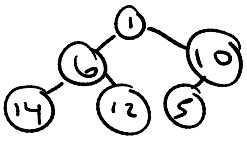
Iteration 6:



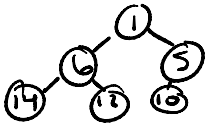
Iteration 7:



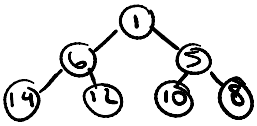
Iteration 8:



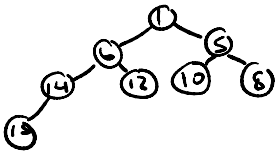
Iteration 9:



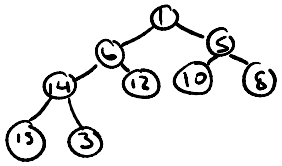
Iteration 10:



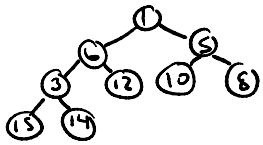
Iteration 11:



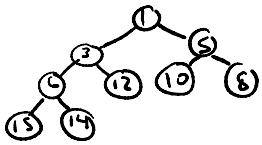
Iteration 12:



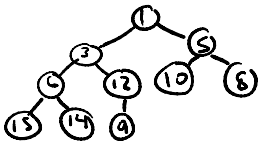
Iteration 13:



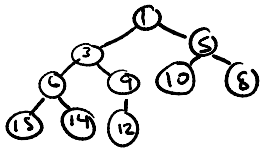
Iteration 14:



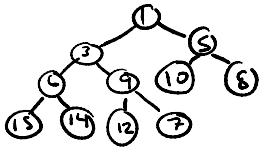
Iteration 15:



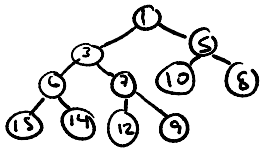
Iteration 16:



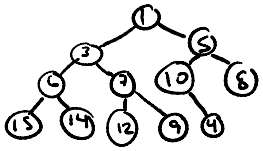
Iteration 17:



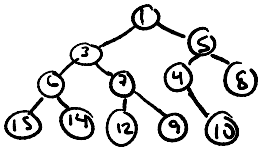
Iteration 18:



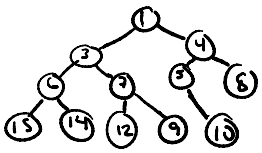
Iteration 19:



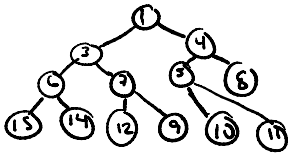
Iteration 20:



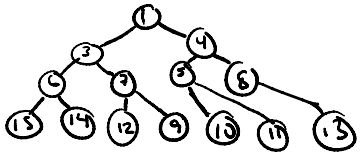
Iteration 21:



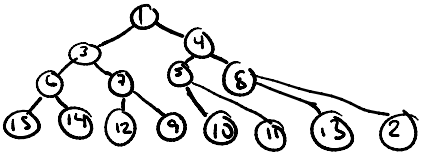
Iteration 22:



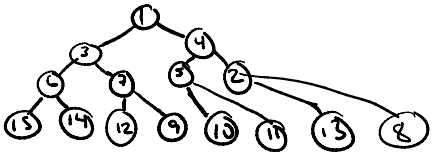
Iteration 23:



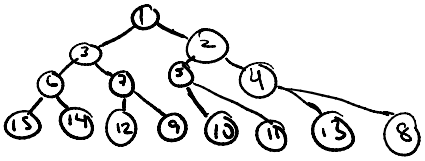
Iteration 24:



Iteration 25:

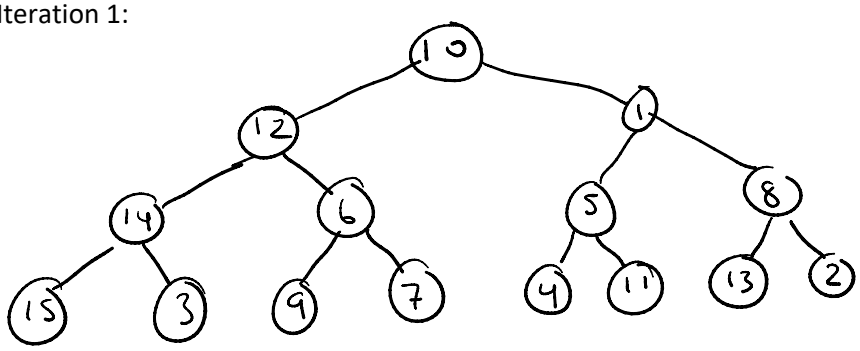


Iteration 26:

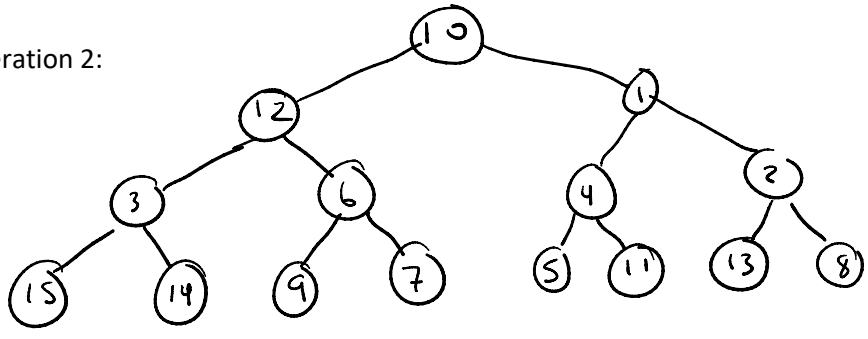


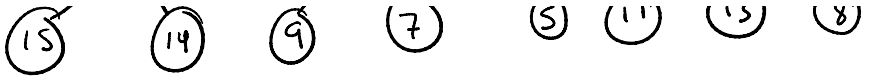
b)

Iteration 1:

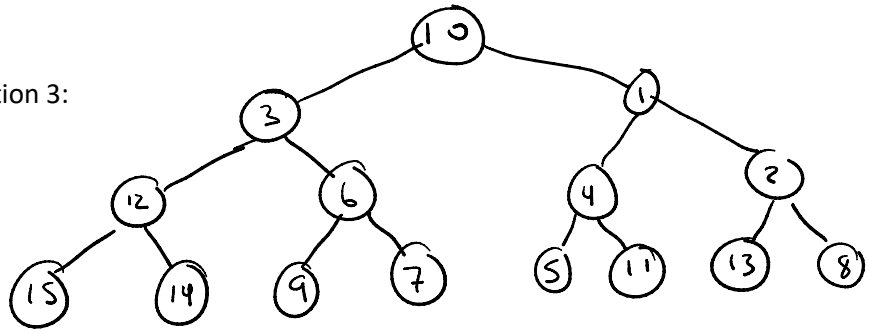


Iteration 2:

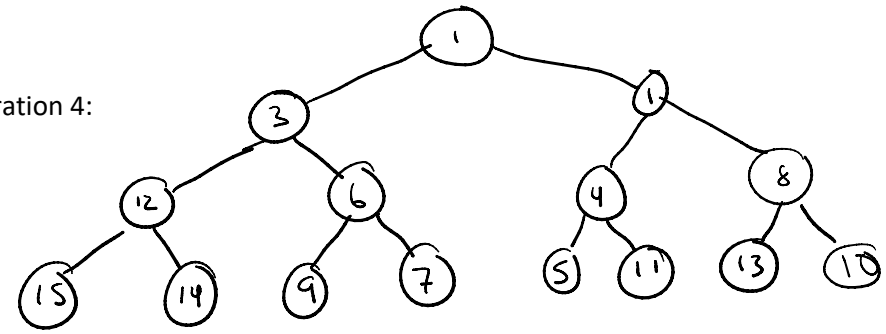




Iteration 3:

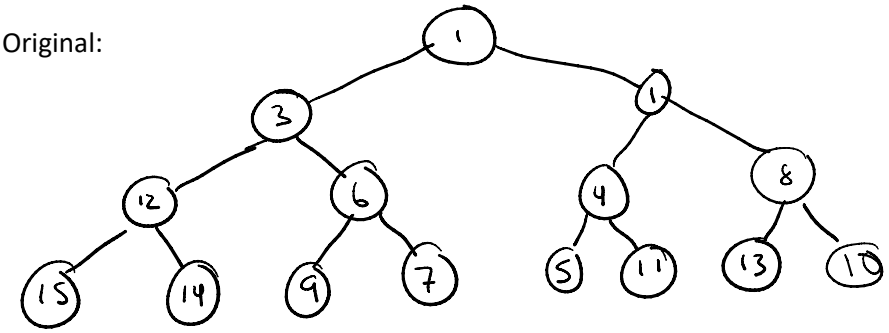


Iteration 4:

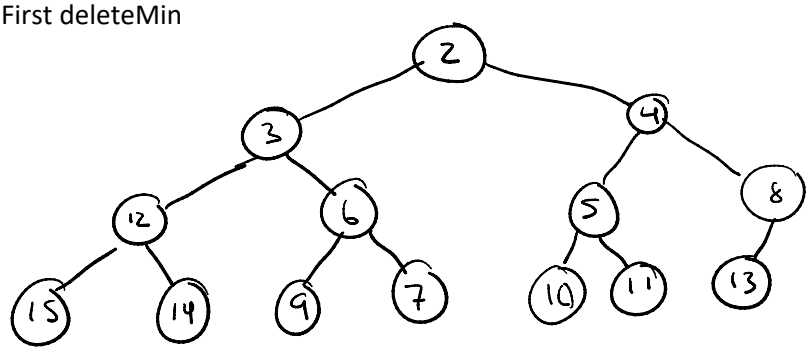


Exercise 6.3:
Show the result of performing three deleteMin operations in the heap of the previous exercise.

Original:

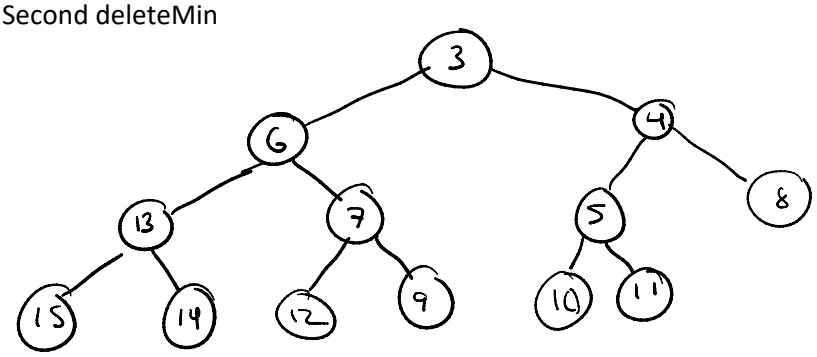


First deleteMin

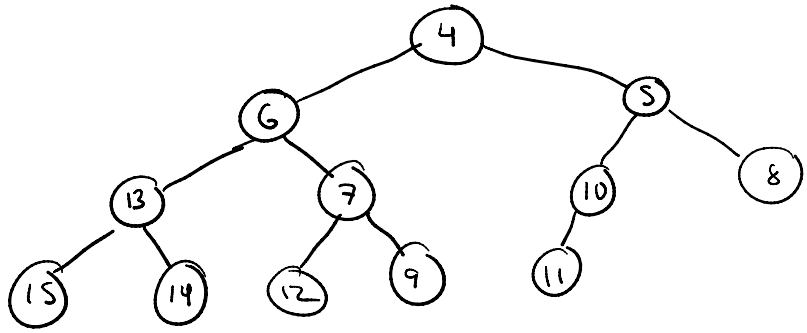


Second deleteMin

Second deleteMin



Third deleteMin



Exercise 6.8:
Show the following regarding the maximum item in the heap:
a. It must be at one of the leaves.
b. There are exactly N/2 leaves.
c. Every leaf must be examined to find it.

a)
In case of min Heap, maximum element is always one of the leaves.

Proof:
Because of the heap property that states that the key at the parent node is smaller than elements at children nodes, this can be proven through the following logic.

If the number of nodes N in a heap is 1 (which has a depth of 0), then that is the only element of the heap and therefore the given statement is true for a single node.

The maximum will be one of its children.
If this maximum key does not have any children, that means it is a leaf, which can be the maximum element of the tree.
Keys at depth 2 or greater will be greater than their parent nodes as well. The maximum element at depth 2 will be smaller than its children still. Therefore, the max element of that depth is at one of them.
Here, as well, the max of that depth can be the max element of the tree if it is a leaf.

Similarly for tree with height log N (all nodes are leaves on this level), the leaves at this level will be greater than their parents. The maximum of all leaves at depth log N is therefore the maximum of the heap.

Therefore, the maximum element is always in one of the leaf nodes in a heap.

b)
A heap is a complete binary tree and can never be skewed. Therefore, any node of the heap can have either 0 or 2 children besides a single node. Only one node could possibly have a single child in a heap due to this property.

Given the height of the heap is h.
At the 0th level, there is only the root. Therefore, the number of nodes = 1 = 2^0
At the 1st level, two nodes are there. Therefore, the number of nodes = 2 = 2^1
At the 2nd level, four nodes exist. Therefore, the number of nodes = 2 = 2^2
At the hth level, the number of nodes = 2^h

If total nodes = N,
N = 2^0 + 2^1 + 2^2 + … + 2^h
N = 2^(h+1) - 1
N + 1 = 1 2^(h + 1)
h + 1 = log(N + 1)
h = log(N + 1) - log(2)
h = log((N + 1)/2)
This implies that the number of leaf nodes is 2^h.

2^h = 2^(log((N + 1)/2)
2^h = (N + 1) / 2
2^h = N/2 + 1/2
2^h = N/2

Therefore, the number of leaf nodes = N/2.

c)
Part (a) proved that the maximum element is always a leaf node.
At any depth, the elements (d + 1) can be smaller than element at depth d, if that is not its parent.
The heap property only relates parent-child relationships.
This implies that the leaf node may be at any depth in which a leaf occurs, and therefore every leaf must be scanned to find the maximum of the heap.

Exercise 6.10:
Give an algorithm to find all nodes less than some value, X, in a binary heap.
Your algorithm should run in O(K), where K is the number of nodes output.

This algorithm assumes that a binary heap is implemented using the standard vector class in C++. The vector implemented maintains the root at index 1 and connects parent-children relationship through the idea that a left child is at double the index of a parent and the right child is at double the index + 1 from a parent. This algorithm also assumes that the data stored in the heap is of the integer type.

The vector implemented will be referred to as heap.

```
void nodesSmallerThan(int x, int index = 1)
{
    if (index >= heap.size())
        return;  // to avoid out-of-bounds errors
    if (heap.at(index) >= x)
```

```
    return;  // skip larger nodes and children nodes
  cout << heap.at(index) << " "; // prints the current node
  nodesSmallerThan(x, 2*index) // moves to left child
  nodesSmallerThan(x, 2*index + 1) // moves to right child
```

Exercise B: (10 pts) Prove using induction on h that:

$$\sum_{i=0}^{h}(2^i) * (h - i) = 2^{h+1} - 1 - (h + 1)$$

Where h = 0.

Basis Step:
Let h = 0 and show that P(0) holds.
2^0 = 1 2^1 - 1 - 1
1 * 0 = 0 2 - 1 - 1
0 = 0
Therefore, the equation holds true when h = 0.

Inductive Step:
Assume h = k is true, where

$$\sum_{i=0}^{k}(2^i) * (k - i) = 2^{k+1} - 1 - (k + 1)$$

An exhaustive form of the above equation would appear as:
1(k) + 2(k - 1) + 4(k - 2) + … + (2^(k + 1))(1) + (2^k)(0)

To prove that h = k + 1 is true, simple substitutions can be performed to create the equation:

$$\sum_{i=0}^{k+1}(2^i) * (k + 1 - i) = 2^{k+1+1} - 1 - (k + 1 + 1)$$

The above summation can be rewritten in an exhaustive form as well:
1(k + 1) + 2(k) + 4(k - 1) + 8(k - 2) + … + (2^k)(1) + (2^(k + 1))(0)
1(k + 1) + 2((k) + 2(k-1) + 4(k - 2) + … + (2^k)(1) + (2^(k + 1))(0))

With a simple substitution into the original equation,
(k + 1) + 2(2^(k + 1) - 1 - (k + 1)) = 2^(k + 2) - 1 - (k + 2)
(k + 1) + 2(2^(k + 1) - 1 - (k + 1)) = 2^(k + 2) - 1 - (k + 2)
(k + 1) + 2(2^(k + 1) - 1 - (k + 1)) = 2^(k + 2) - k - 3
2k + 4 + 2^(k + 1) - 2 - 2k - 2 = -2 2^(k + 2)
2^(k + 2) = 2^(k + 2).

And therefore, P(k + 1) holds when P(k) is assumed to be true.
Thus, the equation is true by inductive proof.