Luigi Sacco h2y9a   André Tertzakian y6l0b

CPSC 221 – William Evans   November 24, 2016

Assignment 3

1. Draw the Dictionary data structure obtained after inserting:

120, 130, 70, 30, 50, 20, 40, 140, 150, 60, 10, 97, 110, 96, 93

one after the other into the following initially empty structures.

   (a) **(5 points) A Hash Table of size 11 that uses chaining (with unsorted linked-list chains, recently inserted items inserted at the front of the chain). Use hash function hash(k) = k mod 11.**

k	120	130	70	30	50	20	40	140	150	60	10	97	110	96	93
hash(k)	10	9	4	8	6	9	7	8	7	5	10	9	0	8	5

The Hash Table will look like this:

0	1	2	3	4	5	6	7	8	9	10
110				70	60 93	50	40 150	30 140 96	130 20 97	120 10

   (b) **(5 points) A Hash Table of size 23 that uses open addressing with double hashing. The first hash function is: hash$_1$(k) = k. The second is: hash$_2$(k) = 13 − (k mod 13). Use mod 23 to restrict the probe sequence to the Hash Table. For each key, indicate the table slots probed.**

| k	120	130	70	30	50	20	40	140	150	60	10	97	110	96	93
k mod 23	5	15	1	7	4	20	17	2	12	14	10	5	18	4	1
hash$_2$(k)	10	13	8	9	2	6	12	3	6	5	3	7	7	8	11
slots probed	5	15	1	7	4	20	17	2	12	14	10	5 12 17	18	4 12 20 5 13	1 12 0

The resulting Hash Table will look like this:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
93	70	140		50	120		30			10		150	96	60	15		40	110	97	20		

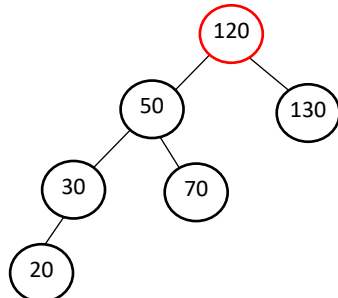(c) **(5 points) An AVL Tree. Draw the tree before and after every double or single rotation.**

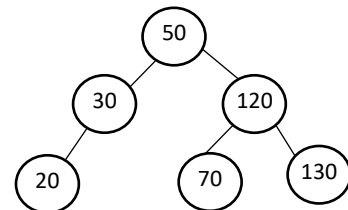insert 120, 130, 70, 30, 50 (unbalanced)
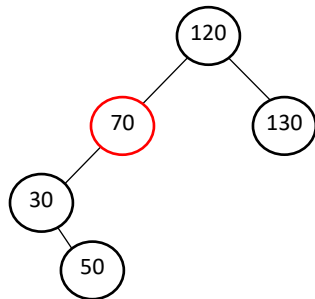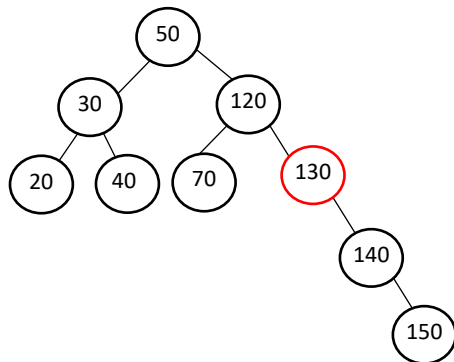

balance (double rotate right)
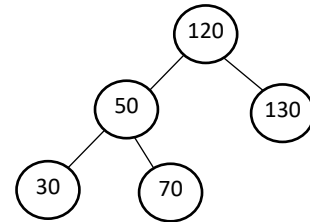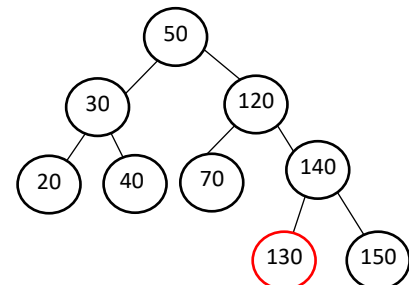

insert 20 (unbalanced)

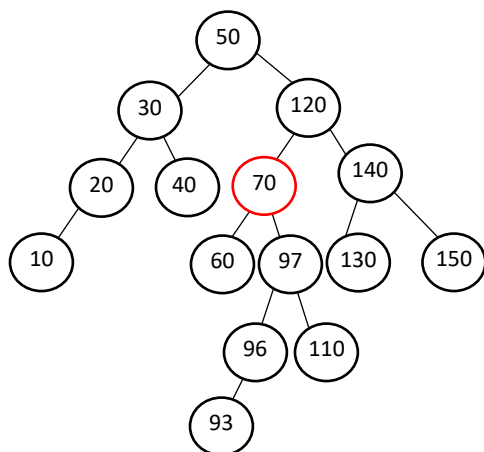balance (rotate right)

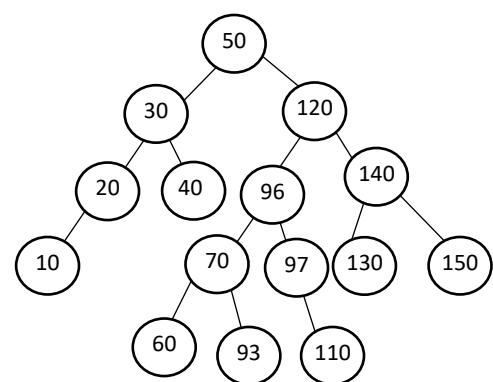inserting 40, 140, 150 (unbalanced)

balancing (rotate left)

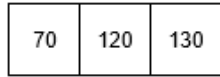inserting 60, 10, 97, 110, 96, 93 (unbalanced)
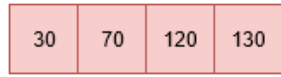

balancing (double rotate left)

The final tree

(d) **(5 points) A B+-Tree with M = 5 and L = 3. When nodes split, put half of the items on the left and half on the right, and put the extra item on the left if there is one. Draw the tree after every insertion that causes a split as well as the final tree.**
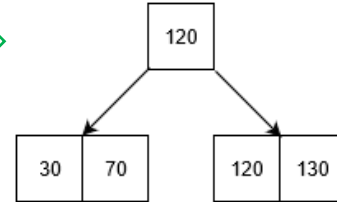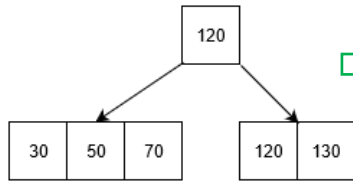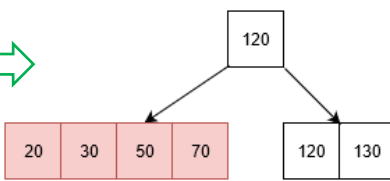
insert 120, 130, 70

| 70 | 120 | 130 |

⇒

insert 30 (overflow)

| 30 | 70 | 120 | 130 |

⇒

splitting tree

```
              120
          /        \
    30 | 70     120 | 130
```

insert 50

```
             120
          /        \
  30 | 50 | 70   120 | 130
```

⇒

insert 20 (overflow)

```
               120
            /        \
  20 | 30 | 50 | 70   120 | 130
```

⇒

splitting tree

```
              50 | 120
          /       |       \
  20 | 30   50 | 70    120 | 130
```
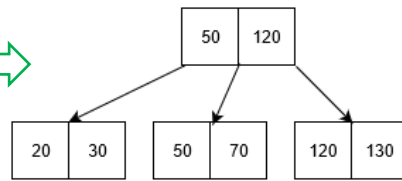
insert 40, 140, 150 (overflow)
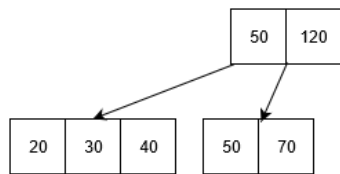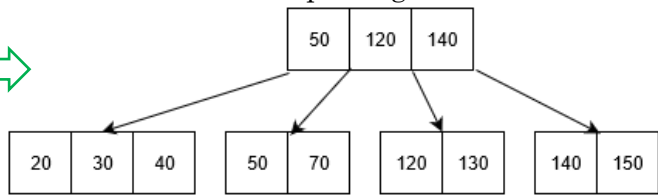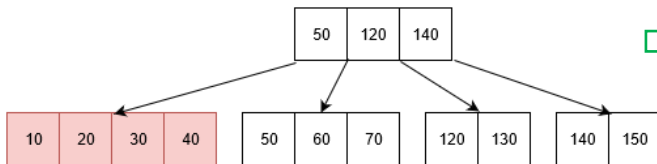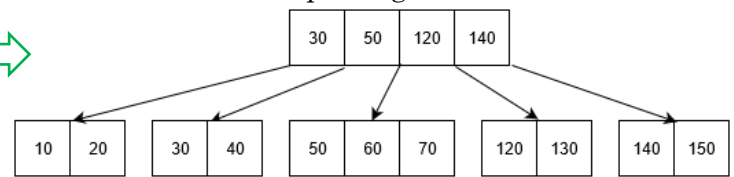
```
                50 | 120
          /        |        \
 20 | 30 | 40   50 | 70   120 | 130 | 140 | 150
```

⇒

splitting tree

```
                   50 | 120 | 140
          /        |        |        \
 20 | 30 | 40   50 | 70   120 | 130   140 | 150
```

insert 60, 10 (overflow)

```
                      50 | 120 | 140
          /            |         |          \
 10 | 20 | 30 | 40   50 | 60 | 70   120 | 130   140 | 150
```

⇒

splitting tree

```
                    30 | 50 | 120 | 140
       /        |          |          |         \
 10 | 20   30 | 40   50 | 60 | 70   120 | 130   140 | 150
```

insert 97 (overflow)

```
                  30 | 50 | 120 | 140
       /        |          |          |         \
 10 | 20   30 | 40   50 | 60 | 70 | 97   120 | 130   140 | 150
```

⇒

splitting tree

```
                          70
                    /            \
              30 | 50          120 | 140
          /     |     \         /        \
 10 | 20  30 | 40  50 | 60   70 | 97  120 | 130  140 | 150
```

insert 110, 96 (overflow)

```
                          70
                    /            \
              30 | 50          120 | 140
          /     |     \         /         \
 10 | 20  30 | 40  50 | 60   70 | 96 | 97 | 110   120 | 130   140 | 150
```

⇒

splitting tree

```
                          70
                    /            \
              30 | 50          97 | 120 | 140
          /     |     \       /      |       \
 10 | 20  30 | 40  50 | 60  70 | 96  97 | 110  120 | 130  140 | 150
```
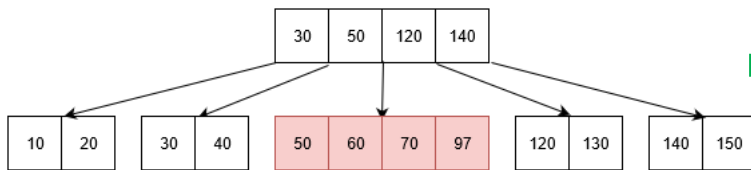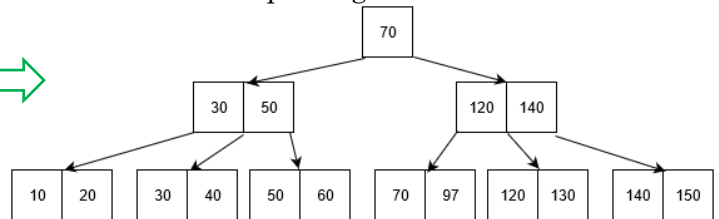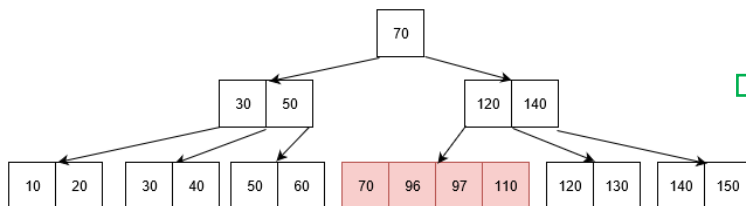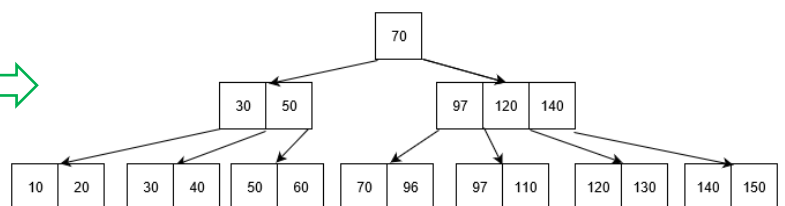
and finally! Insert 93 to complete the B$^+$-tree!
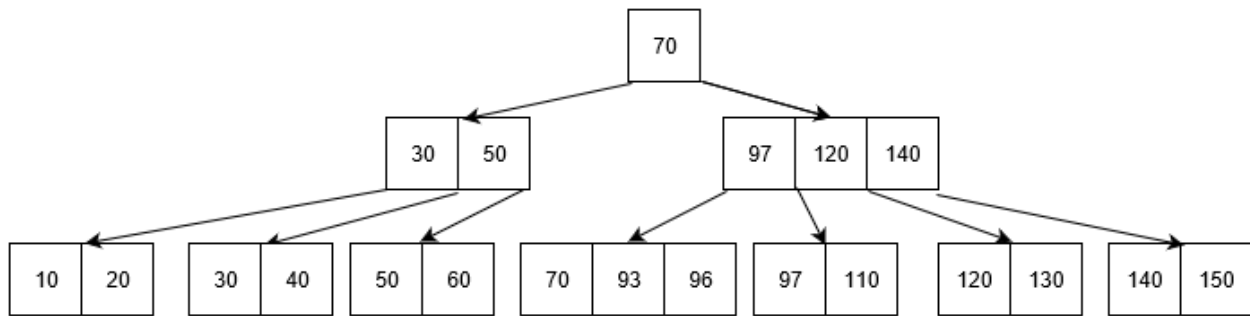
2. (5 points) A room contains 6 computers. Each computer is directly connected to 0 or more of the other computers in the room. Show that there are at least two computers in the room that are directly connected to the same number of other computers.

Hint: You might first try the problem assuming each computer is directly connected to 1 or more of the other computers.

If any computer is directly connected to 1 or more computers, then we can define a function $f : NumComputers \rightarrow PossibilitiesForEachComputer$ with the domain's size being 6 (because there are 6 computers) and the codomain's size being 5 (each computer is connected to 1, 2, 3, 4, or 5 computers). There are 6 pigeons and 5 holes. By the pigeonhole principle, there exists $x_1 \neq x_2$ such that $f(x_1) = f(x_2)$, i.e. two computers connected to the same number of other computers.

We can use this information to show that the statement is also true when one of the computers doesn't have any connections. Indeed, if we have a computer that isn't connected to any other computer, then those other computers can be connected with at most 4 other computers. With this construction, there are 6 computers (pigeons) and 5 ways to assign a value to those computers (they can be connected to 0,1,2,3, or 4 other computers). By the pigeonhole principle, at least two computers are directly connected with the same number of other computers.

We finish off the argument by stating that the previous two cases account for every non-trivial case of this problem. The latter case would not exist if none of the computers had zero connections (because that would turn into the first case). And if there are more than two computers with zero connections, then the answer is given right there. □

3. (5 points) Let $N(h)$ be the smallest number of nodes in an AVL tree of height h. For example, $N(0) = 1$ and $N(1) = 2$. Prove that $N(h) = F(h + 3)-1$ where $F(i)$ is the ith Fibonacci number ($F(0) = 0$, $F(1) = 1$, and $F(i) = F(i-1) + F(i-2)$). To do this, you need to come up with a recurrence relation that defines $N(h)$ and argue why it is correct. Then you need to prove, by induction, that $N(h) = F(h + 3) - 1$.

Working out the recurrence relation for $N(h)$:

$N(0) = 1$, $N(1) = 2$, $N(2) = 4$, $N(3) = 7$, $N(4) = 12$, $\underline{N(h) = N(h\text{-}1) + N(h\text{-}2) + 1}$

The recurrence relation above makes sense because to make an AVL tree of height h, you will need the amount nodes for a tree of height h-1 (for a left subtree) and h-2 (for a right subtree) plus one extra node that combines them (and will be the parent/root note) to get to a height of h. This must be the case since the difference Δd between the height of the subtrees must satisfy $|\Delta d| \leq 1$. Note that the right subtree could also have height h-1 but that would not yield the *minimum* number of nodes.

We already have two base cases for our problem. To reconfirm, $N(0) = F(3) - 1 = 2 - 1 = 1$ and $N(1) = F(4) - 1 = 3 - 1 = 2$. Now for $h > 2$, lets suppose the hypothesis

$$N(h) = F(h + 3) - 1.$$

is correct for all $h \leq k$, where k is any particular positive integer. Now consider

$$N(h+1) = N(h) + N(h-1) + 1$$

which is given by the recurrence relation. Since $h \leq k$ and $h-1 \leq k$, our hypothesis gives

$$N(h+1) = F(h + 3) - 1 + F(h + 2) - 1 + 1$$

$$= F(h + 4) - 1$$

which confirms the desired inductive step result.   □