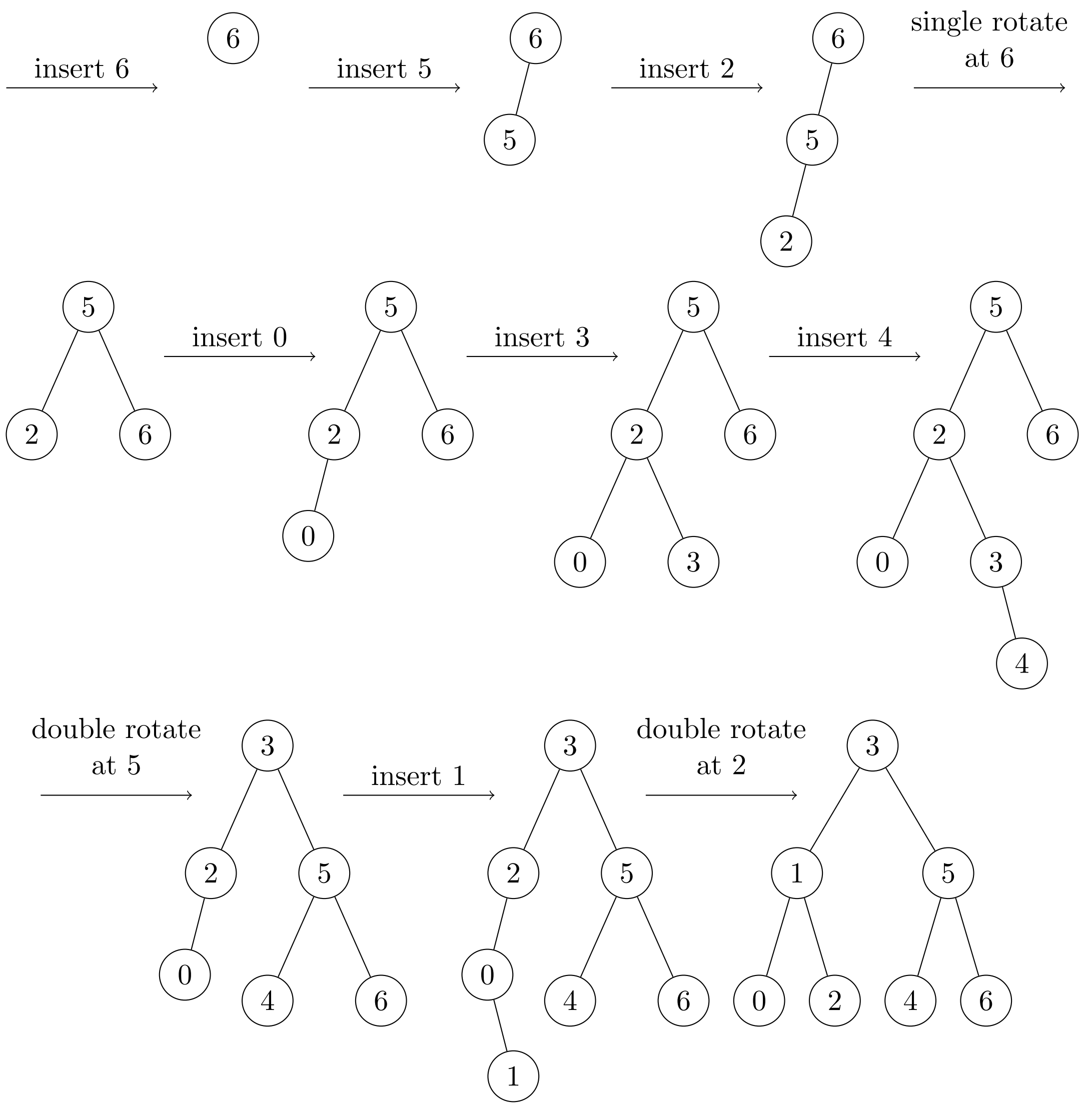
Suggested Solutions to HW #6

1. Perform insertions of the numbers 6, 5, 2, 0, 3, 4, 1 (in this order) into an empty AVL tree. Show each AVL tree after a number has been inserted. If re-balancing operations are performed, please also show the tree before re-balancing and indicate what type of rotation is used in the re-balancing.

Solution.



□

2. The *Partition* procedure for the `Quicksort` algorithm discussed in class is as follows, where *Middle* is a global variable.

Partition $(X, \mathit{Left}, \mathit{Right})$;

begin

  pivot := X[*Left*];

  L := *Left*; R := *Right*;

  while $L < R$ **do**

    while $X[L] \leq$ *pivot* and $L \leq$ *Right* **do** $L := L + 1$;

    while $X[R] >$ *pivot* and $R \geq$ *Left* **do** $R := R - 1$;

    if $L < R$ **then** *swap*($X[L], X[R]$);

  Middle := R;

  swap(X[*Left*], X[*Middle*])

end

(a) Apply the *Partition* procedure to the following array (assuming that the first element is chosen as the pivot).

9	14	6	10	13	12	2	11	1	7	15	3	5	8	16	4

Show the result after each exchange (swap) operation.

Solution.

9	14	6	10	13	12	2	11	1	7	15	3	5	8	16	4
9	(4)	6	10	13	12	2	11	1	7	15	3	5	8	16	(14)
9	4	6	(8)	13	12	2	11	1	7	15	3	5	(10)	16	14
9	4	6	8	(5)	12	2	11	1	7	15	3	(13)	10	16	14
9	4	6	8	5	(3)	2	11	1	7	15	(12)	13	10	16	14
9	4	6	8	5	3	2	(7)	1	(11)	15	12	13	10	16	14
(1)	4	6	8	5	3	2	7	(9)	11	15	12	13	10	16	14

□

(b) Apply the `Quicksort` algorithm to the above array. Show the result after each partition operation.

Solution.

9	14	6	10	13	12	2	11	1	7	15	3	5	8	16	4
1	4	6	8	5	3	2	7	(9)	11	15	12	13	10	16	14
(1)	4	6	8	5	3	2	7	(9)	11	15	12	13	10	16	14
(1)	3	2	(4)	5	8	6	7	(9)	11	15	12	13	10	16	14
(1)	2	(3)	(4)	5	8	6	7	(9)	11	15	12	13	10	16	14
(1)	2	(3)	(4)	(5)	8	6	7	(9)	11	15	12	13	10	16	14
(1)	2	(3)	(4)	(5)	7	6	(8)	(9)	11	15	12	13	10	16	14
(1)	2	(3)	(4)	(5)	6	(7)	(8)	(9)	11	15	12	13	10	16	14
(1)	2	(3)	(4)	(5)	6	(7)	(8)	(9)	10	(11)	12	13	15	16	14
(1)	2	(3)	(4)	(5)	6	(7)	(8)	(9)	10	(11)	(12)	13	15	16	14
(1)	2	(3)	(4)	(5)	6	(7)	(8)	(9)	10	(11)	(12)	(13)	15	16	14
(1)	2	(3)	(4)	(5)	6	(7)	(8)	(9)	10	(11)	(12)	(13)	14	(15)	16

□

3. (6.10) Find an adequate loop invariant for the main while loop in the *Partition* procedure of the `Quicksort` algorithm, which is sufficient to show that after the execution of the last two assignment statements the array is properly partitioned by $X[Middle]$. Please express the loop invariant as precisely as possible, using mathematical notation.

Solution. The algorithm assumes that $Left < Right$. This condition holds throughout the algorithm and we will keep it implicit. A suitable loop invariant for the main while loop is as follows:

$$\begin{array}{ll} & pivot = X[Left] \\ \wedge & \forall i(Left \leq i < L \implies X[i] \leq pivot) \\ \wedge & \forall j(R < j \leq Right \implies pivot < X[j]) \\ \wedge & Left \leq L \leq Right + 1 \\ \wedge & Left \leq R \leq Right \\ \wedge & (L \not< R) \implies (L - 1 = R) \end{array}$$

This loop invariant is maintained before and after every iteration of the loop. Note that the inequalities $i < L$ and $R < j$ in the second and third conjuncts are strict. This is so because when the condition $L < R$ does not hold, the statement $swap(X[L], X[R])$ will not be performed. After the while loop terminates with $L \not< R$ and the following two statements are executed, we can conclude:

$$\begin{array}{ll} & pivot = X[Middle] \\ \wedge & \forall i(Left \leq i \leq Middle \implies X[i] \leq pivot) \\ \wedge & \forall j(Middle < j \leq Right \implies pivot < X[j]) \end{array}$$

which is the (post-)condition desired of the *Partition* algorithm, indicating that the algorithm is indeed correct. □