# Prob1:

static String revert(String input) {

Stack<Character> s = new Stack();

String r = "";

for (int i = 0; i < input.length(); i++) {

char c = input.charAt(i);

if (c == ' ' || i == input.length() - 1) {

if (c != ' ')

s.push(c);

while (!s.isEmpty())

r += s.pop();

r += ' ';

} else

s.push(c);

}

return r;

}

# Prob2:

public int[] sort(int[] input) {

  for (int i : input) {

    this.insert(i);

  }

  sortPrint(root, input, 0);

  return input;

}

private int sortPrint(Node t, int[] output, int from) {

  if (t == null)

    return from;

  int to = sortPrint(t.left, output, from);

  output[to] = t.element;

  to++;

  sortPrint(t.right, output, to);

  return to;

}

Running time: O(nlogn)

Test result:

44 ms -> MergeSort

69 ms -> BSTSort

330 ms -> InsertionSort

520 ms -> SelectionSort

It’s a little slower than MergeSort (& QuickSort) and much faster than other

# Prob3:

|  |  |
| --- | --- |
| **Num nodes n** | **Does there exist a red-black tree with n nodes, all of which are black?** |
| 1 | Yes |
| 2 | No |
| 3 | Yes |
| 4 | No |
| 5 | No |
| 6 | No |
| 7 | Yes |

# Prob4:

|  |  |
| --- | --- |
| **Num nodes n** | **Does there exist a red-black tree with n nodes, where exactly one of the nodes is red?** |
| 1 | No |
| 2 | Yes |
| 3 | No |
| 4 | Yes |
| 5 | Yes |
| 6 | No |
| 7 | No |