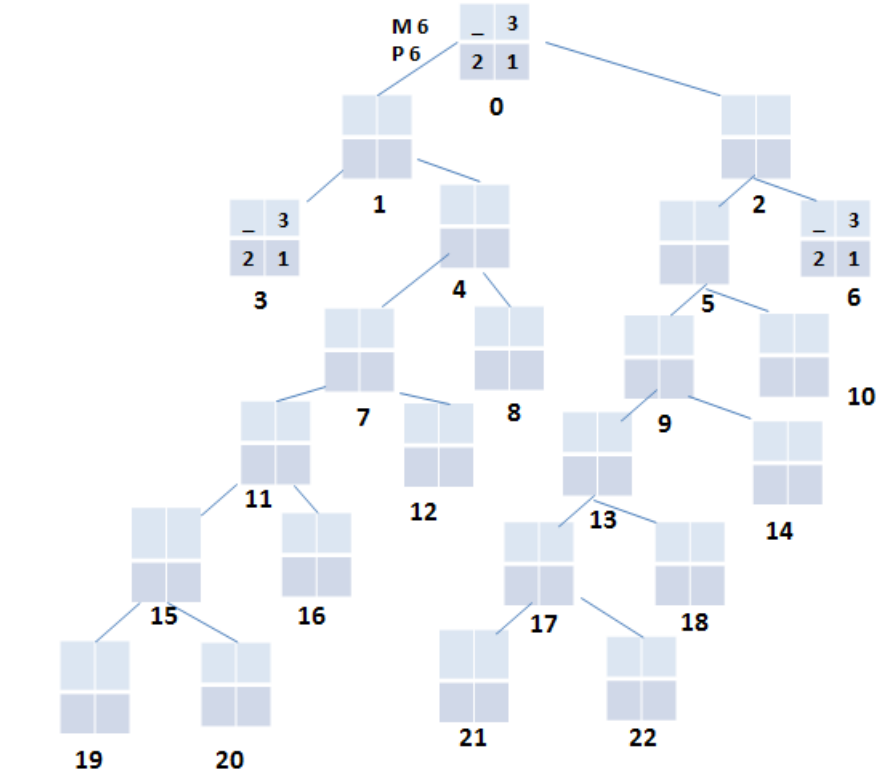
COS 226 Data Structures and Algorithms
Computer Science Department
Princeton University
Fall 2015

Precept - Week 4

1. Programming Assignment (8-Puzzle)
   The 8-puzzle problem is a puzzle played on a 3-by-3 grid with 8 square tiles labeled 1 through 8 and a blank square. Your goal is to rearrange the tiles so that they are in order, using as few moves as possible. You are permitted to slide tiles horizontally or vertically into the blank square.
   First, we will study a simpler version of the problem with 2-by-2 grid. We can call this 3-puzzle as we move 3 square tiles labelled 1 through 3 and a blank square to obtain the target board.

```
 8-puzzle                          3-puzzle
8 1 3          1 2 3          -  3          1  2
4 _ 2          4 5 6          2  1          3  _
7 6 5          7 8 _          initial       goal
initial        goal
```

   (a) Complete the unfilled boards in the following game tree for 3-puzzle. Note that some internal boards(leaf nodes 3 and 6) are filled and will not be considered further (critical optimization).



(b) Compute manhattan distance(M) and priority(P) for each board and Write M and P next to each board in the tree shown in Part(a). (initial board has M=6 P=6). The priority(P) is equal to manhattan + moves. Circle the boards that will not be considered further (eg: critical optimization).

(c) Using A* algorithm, find the solution. Boards are numbered in the game tree starting with 0(initial board) and show the boards that will be inserted (and deleted) into priority queue.

What is the minimum number of moves required to solve the board?

(d) What if we change the priority to use Hamming distance instead. Compute hamming distance (H) for few boards. Will A* algorithm still finds the optimal solution despite which function is used as priority?

(e) Show that the following puzzle is unsolvable

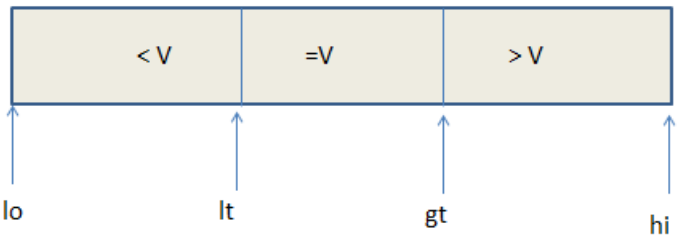
1	_	3
2	5	4
7	6	8

2. *Quicksort.*

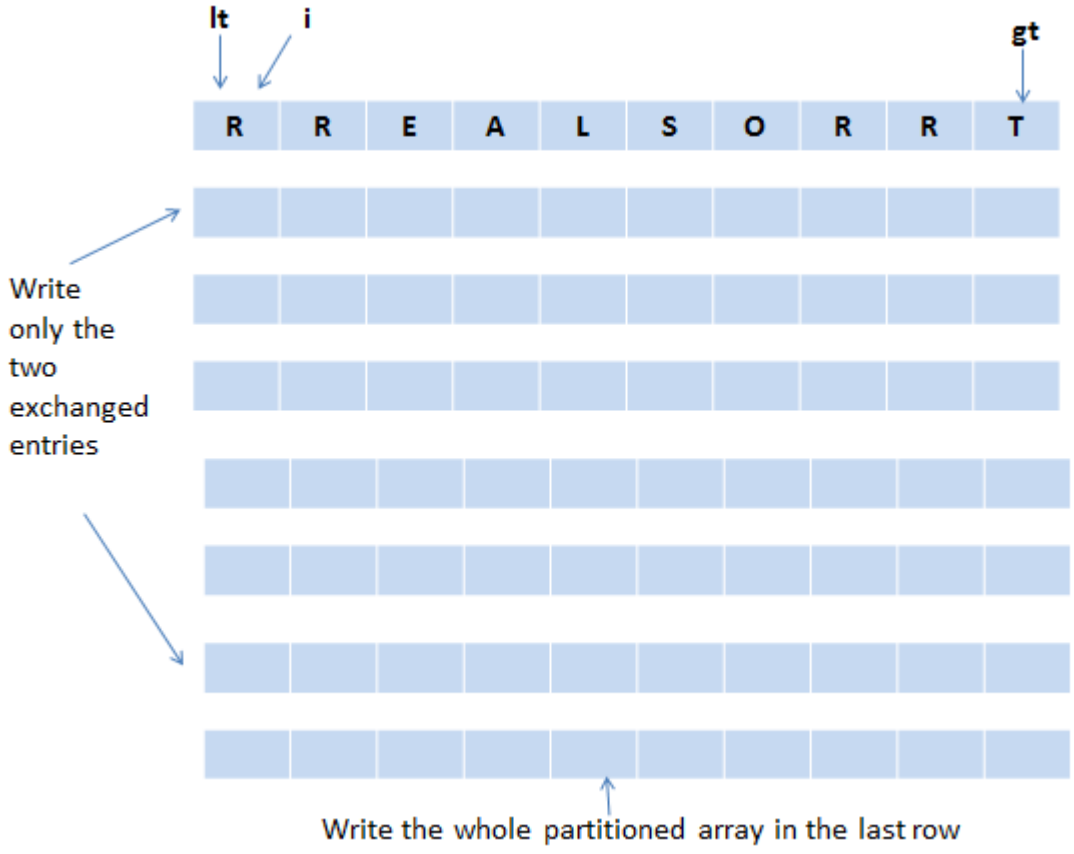
Suppose that the following array result from the shuffle in Algorithm 2.5

`R R E A L S O R R T`

We apply Dijkstra's `3-way partitioning algorithm` to sort the array. The `3-way partitioning algorithm` groups elements into 3 groups as shown below, where v is the current pivot element.



Show the result of the `first` call to `partition()` by giving the contents of the array after each exchange. You can provide only the two elements that were exchanged. You can write the entire partitioned array after the final exchange.



3. *Priority Queues*

Suppose we have the following code that uses a binary heap-based MinPQ. Assume $N > k$.

```
int k = 10;
MinPQ<Integer> pq = new MinPQ<Integer>();
int N = a.length;

for (int i = 0; i < N; i++) {
   pq.insert(a[i]);
   if (pq.size() > k) pq.delMin(); /* MARK */
}

for (int i = 0; i < k; i++)
   System.out.println(pq.delMin());
```

(a) What does this code output?

(b) What is the run time of the code in terms of N and k?

(c) Suppose we remove the entire line marked with /* MARK */. What does the code output?

(d) What is the run time of the code in terms of N and k?