PRINCETON UNIVERSITY

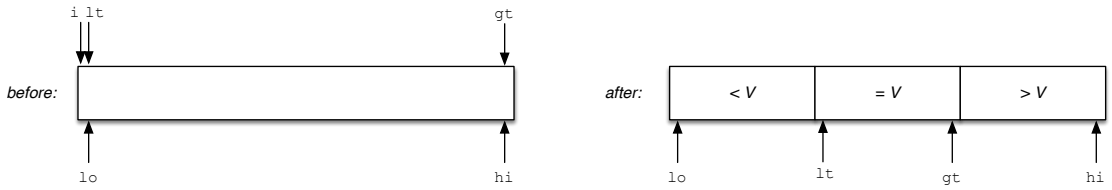
COS 226–Algorithms and Data Structures

Week 4: *3-way Quicksort & 8-Puzzle (Video §6.C & Algorithms §2.3)*

Version: October 5, 2017

Exercise 1 – Quicksort

Apply Dijkstras 3-way partitioning algorithm to *partition* the array below (not sorting). This algorithm groups elements into three groups as shown here, where v is the current pivot element.



Recall that in this algorithm:

- Let `v` be partitioning item which is initially `a[lo]` (`v` never changes afterwards).
- Scan `i` from left to right.
  - `(a[i] < v)`: exchange `a[lt]` with `a[i]`; increment both `lt` and `i`
  - `(a[i] > v)`: exchange `a[gt]` with `a[i]`; decrement `gt`
  - `(a[i] == v)`: increment `i`.
- The algorithm terminates when `i > gt`.

A. When would you use 3-way partitioning, instead of 2-way partitioning? Explain briefly.

B. Suppose that the 3-way partitioning algorithm is applied to the array below: Given is the final outcome and 4 initial iterations of the 3-way partitioning algorithm. Complete the 5 missing rows after each compare and increment or swap.(Consider that every occurrence of R_i is the same letter; the indices are just provided to clarify which R's are involved in each exchange.)

R_1	R_2	E	A	L	S	O	R_3	R_4	T
R_1									
R_1	R_2								
E	R_2	R_1							
E	A	R_1	R_2						
E	A	L	O	R_1	R_4	R_2	R_3	T	S

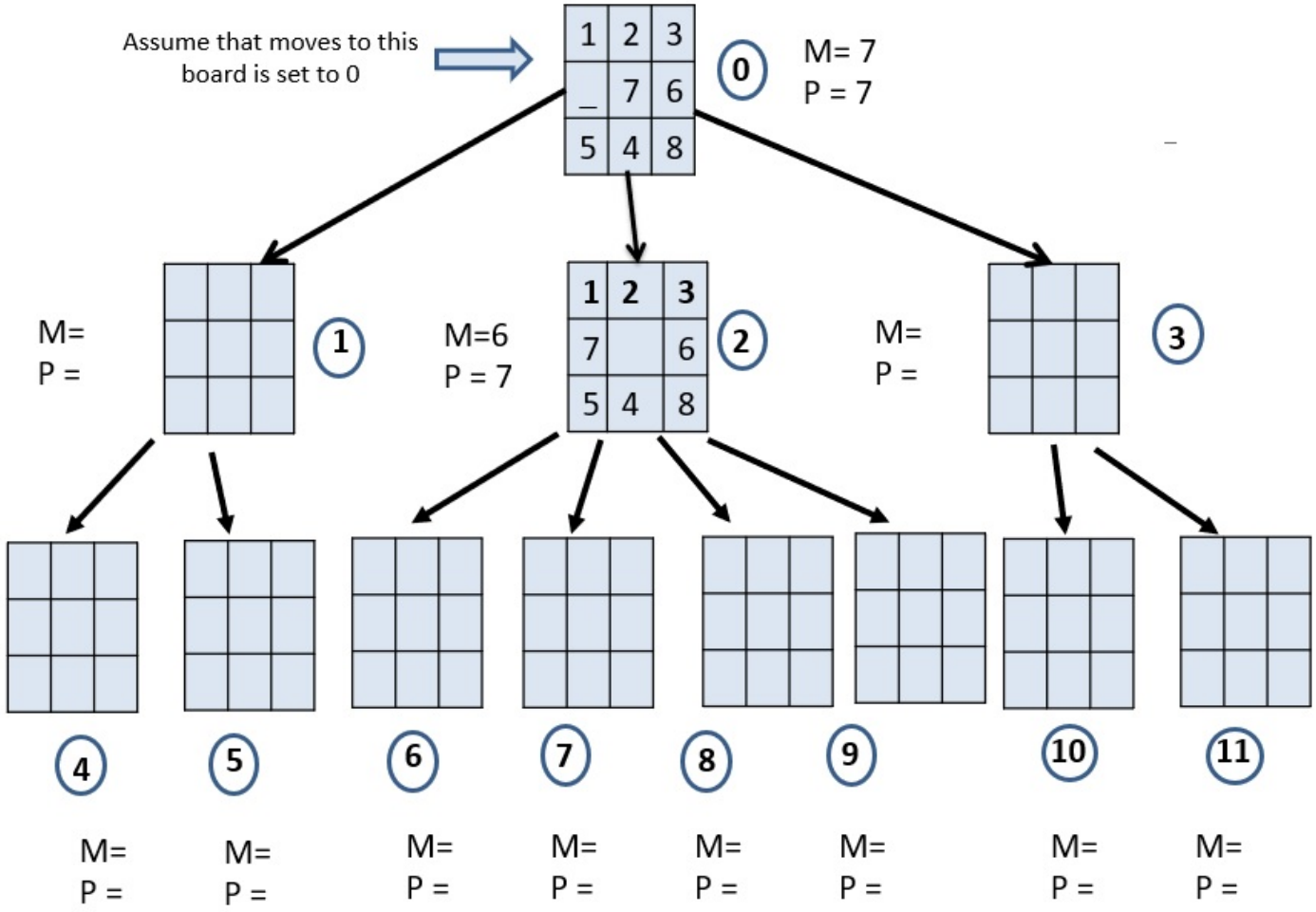
C. What is the order of growth of the number of comparisons and exchanges in 3-way partitioning, in the best and worst cases?

	Best	Worst
Comparisons		
Swaps		

Exercise 2 – 8-Puzzle

The 8-puzzle problem is a puzzle played on a 3-by-3 grid with 8 square tiles labelled 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. Given below is a state of an 8-puzzle algorithm.

A. Complete the unfilled boards in the following partial game tree for 8-puzzle.



B. The Manhattan priority function is defined as the sum of the Manhattan distances (sum of the vertical and horizontal distances) from the tiles to their goal positions, plus the number of moves made so far to get to the search node. Compute Manhattan distance(M) and priority(P) for at least **3 other boards**. Write M and P next to 3 boards in the tree shown in Part A. Some distances are already given.

C. Circle the boards in Part A, that will not be considered further (i.e the critical optimization).

D. Boards are numbered in the game tree starting with 0 (initial board) as shown. Show the order in which the boards are inserted into (and deleted from) the priority queue (PQ) during the execution of the A* algorithm. Use one figure each for a round of deleteMin() and insert() operations. We note that not all boards are inserted to the PQ at the same time. When two boards have the same priority, pick the one with the smallest board index. Show the board with the minimum priority (in the circle) and place the indices of other boards currently in the priority queue in any order inside the triangular part. You may stop when a board from the lowest level has been removed from the priority queue (there are no more boards to enqueue as shown in part A)