PRINCETON UNIVERSITY

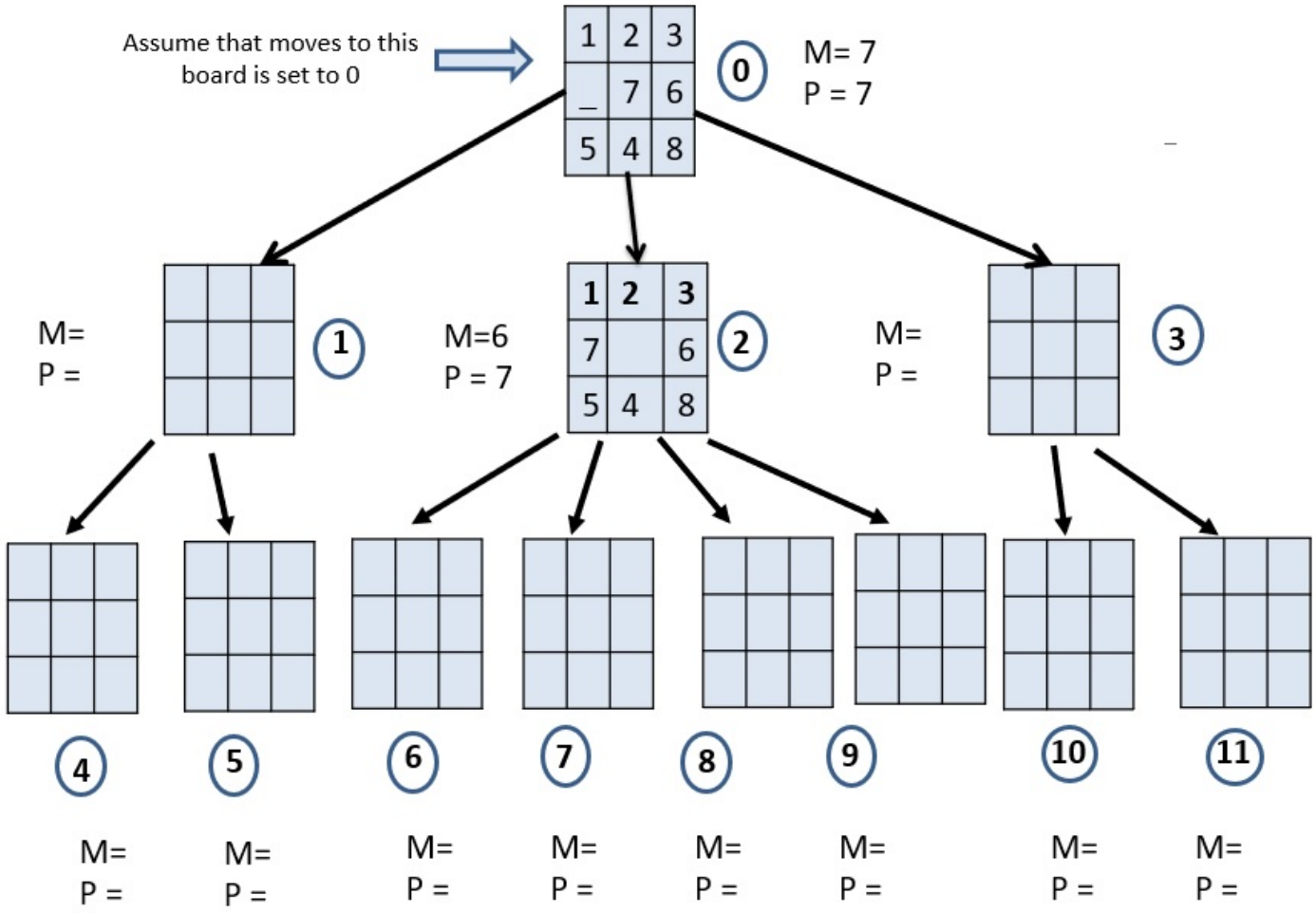
COS 226–Algorithms and Data Structures
Week 4: *8-Puzzle (Algorithms §2.3)*

Version: February 27, 2018

Exercise 1 – 8-Puzzle

An example of the 8-puzzle problem for a 3-by-3 grid is shown below along with the first few levels of the tree of potential moves. Your program will find the branch of the tree with as few moves as possible. (This question is continued on the next page.)

A. Complete the unfilled boards in the following partial game tree.

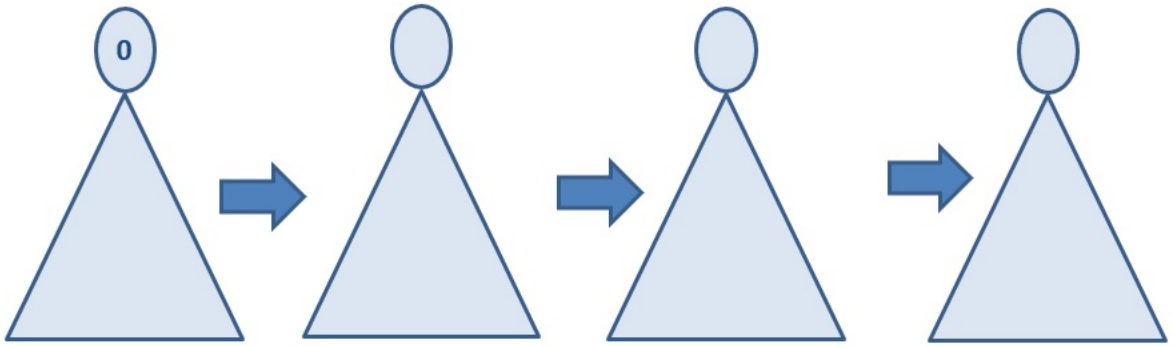


B. The Manhattan priority function is defined as the sum of the Manhattan distances (sum of the vertical and horizontal distances) from the tiles to their goal positions, plus the number of moves made so far to get to the search node. Compute Manhattan distance(M) and priority(P) for at least **3 other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



Exercise 2 – Priority Queues

Suppose we have the following code which uses a binary-heap based minimum priority queue (`MinPQ`). Assume that $N > k$, and that `a[]` is an array containing random integers.

```
1  int k = 5;  // For instance (k could take another value).
2  int a[] = { 0, 1, 2, 3, 4, 5, 6}; // for instance
3  MinPQ<Integer> pq = new MinPQ<Integer>();
4  int N = a.length;
5
6  for (int i = 0; i < N; i++) {
7      pq.insert(a[i]);
8      if (pq.size() > k) pq.delMin();
9  }
10
11 for (int i = 0; i < k; i++)
12     System.out.println(pq.delMin());
```

A. Describe what the code outputs in terms of the array `a[]` and the parameter k.

B. What is the order of growth of the running time of the code as a function of both N and k?

C. Suppose we were to remove line 7, what would the code output?

D. With line 7 removed, what would the order of growth of the running time now be?

