Problem Set 5

Sequential Search, Binary Search

---

1. Given the following sequence of integers:

   ```
   3, 9, 2, 15, -5, 18, 7, 5, 8
   ```

   1. What is the average number of comparisons for a successful search assuming all entries are searched with equal probability? Show your work.
   2. Suppose the search probabilities for the elements of this list are, respectively:

      ```
      0.1, 0.3, 0.05, 0.2, 0.05, 0.1, 0.05, 0.1, 0.05
      ```

      What is the average number of comparisons for successful search with these search probabilities? Show your work.
   3. Rearrange the list entries in a way that would result in the lowest number of comparisons on the average for successful search, given the above probabilities of search. What is this lowest number of comparisons? Show your work.

---

2. An adaptive algorithm to lower average match time for sequential search is to move an item by one spot toward the front every time it is matched. (Unless it is already at the front, in which case nothing is done on a match.) Complete the following modified sequential search on a linked list with this move-toward-front adaption. Assume a generic `Node` class with `data` and `next` fields.

   ```
   public class LinkedList<T> {
       private Node<T> front;
       int size;
       ...
       // moves the target one place toward the front
       // doesn't do anything if target is in the first node
       // returns true if target found, false otherwise
       public boolean moveTowardFront(T target) {
           // COMPLETE  THIS METHOD
       }
   }
   ```

---

3. Draw the comparison tree for binary search on a sorted array of length 11.
   1. What is the worst case number of comparisons for success? For failure?
   2. What is the average number of comparisons for success, assuming equal likelihood of success at any spot?
   3. What can you say about the average number of comparisons for failure? Can you approximate it within a small range? Does it depend on the distribution of the probabilities of failing at the various failure spots?

---

4. * A variant of binary search, called *lazy* binary search, works as described in the following algorithm, where `t` is the target to search, and `n` is the size of the array:

   ```
   left <-- 0
   right <-- n-1
   while (left < right) do
      mid <-- (left + right)/2
      if (t > A[mid]) then
         left <-- mid + 1
   ```

```
    else
       right <-- mid
    endif
 endwhile
 if (t == A[left]) then
    display "found at position", left
 else
    display "not found"
 endif
```

1. Trace this algorithm on the following array, with 46 as the search target:

   ```
   10   15   25   30   45   46   48   72   76   80   93
   ```

   How many comparisons are made by the time a match is found? How does your answer compare with that for regular binary search?
2. Repeat with 40 as the target. How many comparisons are made until failure is detected? How does your answer compare with that for regular binary search?
3. Draw the comparison tree for lazy binary search on an array of length 11 (same length as the example array above).
   1. What is the worst case number of comparisons for success? For failure?
   2. What can you say about the range of values for the average number of comparisons for success? For failure?
   3. Under what conditions is it preferable to use lazy binary search over the regular binary search?

---

5. An alternative algorithm for searching on a sorted array of size *n* works as follows. It divides the array into *m* contiguous blocks each of size *s*. (Assume that *s* divides into *n* without remainder).

   Here is the algorithm to search for a key *k* in sorted array *A*.

   ```
   Compare k with the last entry in the first block, i.e. A[s-1]
   If there is match, then stop with success

   Otherwise, check if k < A[s-1]
   If so, perform a sequential search on the block of entries from
   A[0] to A[s-2]. If there is a match, stop with success,
   otherwise stop with failure.

   If k is not < A[s-1] then continue the process by
   repeating the above on the second block, and so on.
   ```

   - a) What is the **worst** case number of searches for success?
   - b) ** What is the **average** case number of searches for success?