Nathan Bickel
CSCE 350: Section H01
Professor: Dr. Narayanan
April 9, 2023

CSCE 350 Homework 5

Problem 1 Consider the problem of finding the distance (either Manhattan or Euclidean distance) between the two closest numbers in an array of n numbers. (10 points)

(a) Design a presorting-based algorithm for solving this problem and determine its efficiency class (write the pseudo-code with supporting description).

(b) Compare the efficiency of this algorithm with that of the brute-force algorithm.

Solution.

(a) We design an algorithm:

```
// Input: Zero-indexed array A with n real numbers
// Output: Distance between the closest two points
function ClosestPoints(A)
    A ← MergeSort(A)
    min ← ∞
    for i ← 0 to n − 2 do
        d ← |A[i + 1] − A[i]|
        if d < min then
            if d = 0 then
                return 0
            end if
            min ← d
        end if
    end for
    return min
end function
```

(b) Since Merge Sort is in $\Theta(n \log n)$ and we iterate through the array at most $n - 1$ times after sorting, this is in $\Theta(n \log n)$. The brute force algorithm is in $\Theta(n^2)$, so our algorithm is better.

Problem 2 For each of the following lists, construct an AVL tree by inserting their elements successively, starting with the empty tree. (15 points)

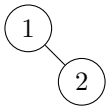
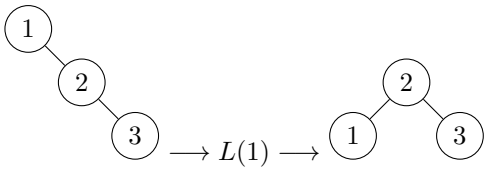
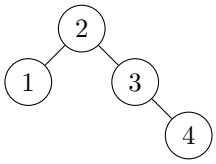
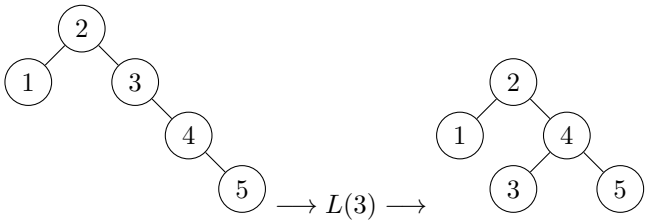
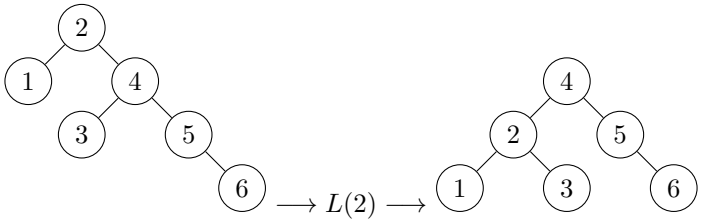
(a) $1, 2, 3, 4, 5, 6$

(b) $6, 5, 4, 3, 2, 1$

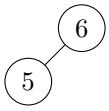
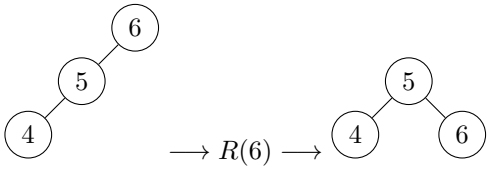
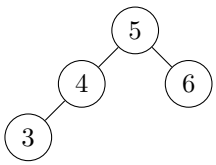
(c) $3, 6, 5, 1, 2, 4$

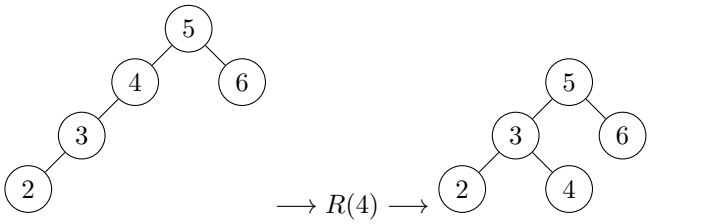
Solution.

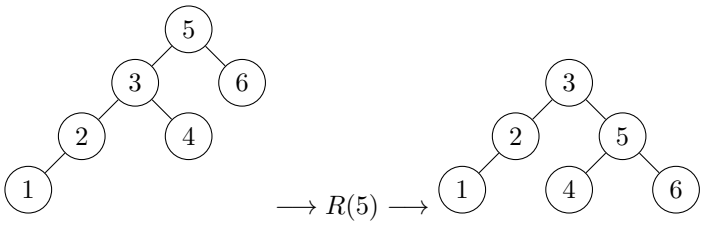
(a)

- 
1
- 
1
2
- 
1
2
3
$\longrightarrow L(1) \longrightarrow$
2
1
3
- 
2
1
3
4
- 
2
1
3
4
5
$\longrightarrow L(3) \longrightarrow$
2
1
4
3
5
- 
2
1
4
3
5
6
$\longrightarrow L(2) \longrightarrow$
4
2
5
1
3
6

(b)

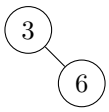
- 
6
- 
6
5
- 
6
5
4
$\longrightarrow R(6) \longrightarrow$
5
4
6
- 
5
4
6
3

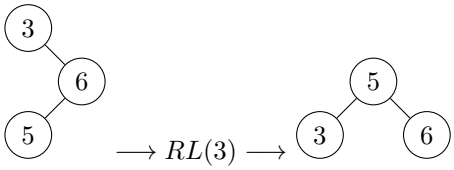
- 

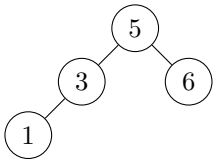
- 

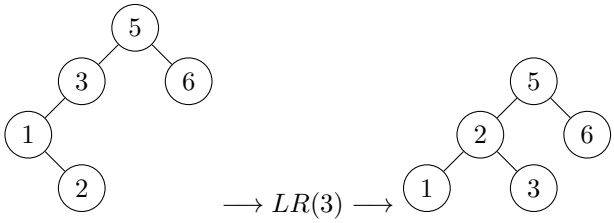
(c)

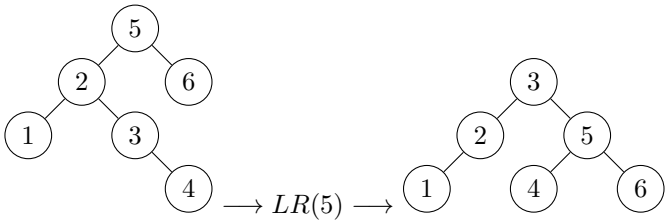
- 

- 

- 

- 

- 

- 

Problem 3 Heaps: (15 points)

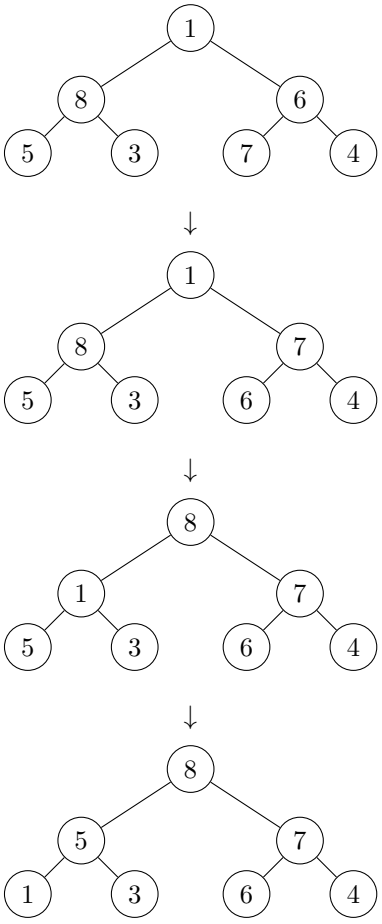
(a) Construct a heap for the list $1, 8, 6, 5, 3, 7, 4$ by the bottom-up algorithm.

(b) Outline an algorithm for checking whether an array $H[1 \dots n]$ is a heap and determine its time efficiency. (Pseudo-code)

(c) Read Section 6.4 in the book. Explain (in your own words) and derive the worst-case efficiency of the algorithms HeapBottomUp and HeapSort.

Solution.

(a)



(b) We design an algorithm below:

We make two comparisons for $n/2$ elements, so this algorithm is in $\Theta(n)$.

(c) For HeapBottomUp, the worst case is that the array is full, and each key needs to move. Since moving down a level requires two comparisons, we will need at most $2n$ comparions. Thus, this is in $O(n)$. For HeapSort, we know that making the array into a heap is in $\Theta(n)$, and since we have to remove $n-1$ vertices, each of which takes $O(\log n)$, this is in $O(n) + O(n \log n) = O(n \log n)$.

Problem 4 Horner's Rule: (10 points)

(a) Apply Horner's rule to evaluate the polynomial $p(x) = 3x^4 - x^3 + 2x + 5$ at $x = -2$.

(b) Use the results of the above application of Horner's rule to find the quotient and remainder of the division of $p(x)$ by $x + 2$.

```
// Input: One-indexed array H with n real numbers
// Output: True if and only if H is a heap
function IsHeap(H)
    for i ← 1 to ⌊n/2⌋ do
        if H[i] < H[2i] or (2i + 1 ≤ n and H[i] < H[2i + 1]) then
            return false
        end if
    end for
    // Parental dominance holds everywhere, so this is a heap
    return true
end function
```

Solution.

(a) We construct a table as in the textbook:

coefficients	3	-1	0	2	5
$x = -2$	3	$(-2)(3) - 1 = -7$	$(-2)(-7) + 0 = 14$	$(-2)(14) + 2 = -26$	$(-2)(-26) + 5 = 57$

So $p(-2) = 57$.

(b) Using these coefficients, we have

$$\frac{3x^4 - x^3 + 2x + 5}{x - 2} = 3x^3 - 7x^2 + 14x - 26 + \frac{57}{x - 2}.$$

Problem 5 Consider the problem of searching for genes in DNA sequences using Horspool's algorithm. A DNA sequence is represented by a text on the alphabet $\{A, C, G, T\}$, and the gene or gene segment is the pattern. (10 points)

(a) Construct the shift table for the following gene segment of your chromosome 10: $TCCTATTCTT$

(b) Explain the solution using step-by-step application of Horspool's algorithm to locate the above pattern in the DNA sequence: $TTATAGATCTCGTATTCTTTTATAGATCTCCTATTCTT$

Solution.

(a) We construct a table as in the textbook:

character c	A	C	G	T
shift $t(c)$	5	2	10	1

(b) We will apply Horspool's algorithm. At each step, the substring of the text being compared to the pattern ($TCCTATTCTT$) will be indicated with dividing lines:

- $|TTATAGATCT|CGTATTCTTTTATAGATCTCCTATTCTT$
- $T|TATAGATCTC|GTATTCTTTTATAGATCTCCTATTCTT$

- $TTA|TAGATCTCGT|ATTCTTTTATAGATCTCCTATTCTT$
- $TTAT|AGATCTCGTA|TTCTTTTATAGATCTCCTATTCTT$
- $TTATAGATC|TCGTATTCTT|TTATAGATCTCCTATTCTT$
- $TTATAGATCT|CGTATTCTTT|TATAGATCTCCTATTCTT$
- $TTATAGATCTC|GTATTCTTTT|ATAGATCTCCTATTCTT$
- $TTATAGATCTCG|TATTCTTTTA|TAGATCTCCTATTCTT$
- $TTATAGATCTCGTATTC|TTTTATAGAT|CTCCTATTCTT$
- $TTATAGATCTCGTATTCT|TTTATAGATC|TCCTATTCTT$
- $TTATAGATCTCGTATTCTTT|TATAGATCTC|CTATTCTT$
- $TTATAGATCTCGTATTCTTTTA|TAGATCTCCT|ATTCTT$
- $TTATAGATCTCGTATTCTTTTAT|AGATCTCCTA|TTCTT$
- $TTATAGATCTCGTATTCTTTTATAGATC|TCCTATTCTT|$

In the last comparison, the pattern matches the whole substring of the text, so we are done.

Problem 6 How many character comparisons will be made by Horspool's algorithm in searching for each of the following patterns in the binary text of 1000 zeros? (15 points)

(a) 00001

(b) 10000

(c) 01010

Solution.

(a) In the shift table, $t(0) = 1$, since there is a 0 in the second to right element. So each time the 0 in the substring doesn't match the 1 in the right position, the substring will be shifted over by one. Since there are $1000 - 5 + 1 = 996$ possible locations for the pattern to match, Horspool's algorithm will make 996 character comparisons.

(b) In the shift table, $t(0) = 1$, since there is a 0 in the second to right element. Each substring the pattern is compared to, the last four zeroes in the pattern will match the last four zeroes in the substring, and when the 1 in the pattern doesn't match the 0 in the substring, the substring will be shifted over by one. Since there are 996 possible locations for the pattern to match, Horspool's algorithm will make $(996)(5) = 4980$ comparisons.

(c) In the shift table, $t(0) = 2$, since there is a 0 in the third to right element. Each substring the pattern is compared to, the last 0 of the pattern and the substring will match, and then the 1 will not match. So the substring will be shifted over two. Thus, 448 possible substrings will be checked with two comparisons in each, so Horspool's algorithm will make $(448)(2) = 996$ comparisons.

Problem 7 For the input $30, 20, 56, 75, 31, 19$ and hash function $h(K) = K \mod 11$: (25 points)

(a) construct the open hash table.

(b) construct the closed hash table.

(c) find the largest number of key comparisons in a successful search in both the tables.

(d) find the average number of key comparisons in a successful search in both the tables.

(a) We construct the open hash table:

keys	30	20	56	75	31	19
hash addresses	8	9	1	9	9	8

(b) Then, we construct the closed hash table by moving the colliding keys forward until an empty spot is found:

keys	30	20	56	75	31	19
hash addresses	8	9	1	10	0	2

In the open hash table:

- 30 takes 1 comparison, since 30 is the first item in the linked list at key 8.
- 20 takes 1 comparison, since 20 is the first item in the linked list at key 9.
- 56 takes 1 comparison, since 56 is the first item in the linked list at key 1.
- 75 takes 2 comparisons, since 75 is the second item in the linked list at key 9.
- 31 takes 3 comparisons, since 31 is the third item in the linked list at key 9.
- 19 takes 2 comparisons, since 19 is the second item in the linked list at key 8.

In the closed hash table:

- 30 takes 1 comparison, since 30 is at key 8.
- 20 takes 1 comparison, since 20 is at key 9.
- 56 takes 1 comparison, since 56 is at key 1.
- 75 takes 2 comparisons, since 75 is 1 key after key 9.
- 31 takes 3 comparisons, since 31 is 2 keys after key 9.
- 19 takes 6 comparisons, since 19 is 5 keys after key 8.

(c) From these computations, the largest number of key comparisons in the open hash table is 3, and the largest in the closed table is 6.

(d) From these computations, in the open hash table, the average number of key comparisons is $\frac{1+1+1+2+3+2}{6} = \frac{5}{3}$ comparisons. In the closed hash table, the average number of key comparisons is $\frac{1+1+1+2+3+6}{6} = \frac{7}{3}$ comparisons.

Problem Bonus 3 Prove that the number of directed paths of length $k > 0$ from the i-th vertex to j-th in a graph (undirected or directed) equals the (i, j)-th element of A^k, where A is the adjacency matrix of the graph.

Solution.

We will induct on k. First, let $k = 1$, and let $i, j \in [n]$. If v_i and v_j are neighbors, there is 1 v_iv_j-walk of length 1, and 0 v_iv_j-walks of length 1 otherwise. ${A^1}_{ij} = 1$ if and only if v_i and v_j are neighbors, so the claim holds for $k = 1$.

Next, let $k \in \mathbb{N}$ and assume the claim holds for k. We have $A^{k+1} = \left(A^k\right)(A)$. By definition, we have

$$ {A^{k+1}}_{ij} = \sum_{m=1}^{n} {A^k}_{im} A_{mj}. $$

We claim this counts the number of v_iv_j-walks of length $k+1$. To see this, let $m \in [n]$. ${A^k}_{im}$ is the number of v_iv_m-walks of length k by the induction hypothesis. Thus, if $v_mv_j \in E(G)$, there are ${A^k}_{im}$ possible v_iv_j-walks of length $k+1$ whose second-to-last vertices are v_m (simply walk to v_m and then to v_j). Otherwise, there are no possible v_iv_j-walks of length $k+1$ whose second-to-last vertices are v_m.

To get the total number of v_iv_j-walks of length $k+1$, we should sum ${A^k}_{im}$ for all m such that $v_mv_j \in E(G)$. Since $A_{mj} = 1$ if and only if $v_mv_j \in E(G)$, this is equal to the sum above. So if the claim holds for k, it also holds for $k+1$, and therefore it is true for all $k \in \mathbb{N}$. □