Algorithm assignment 4
Nisharg Gosai

Problems

1. (10pts) Explain why the worst-case running time for bucket sort is Θ(n2).) What simple change to the algorithm preserves its linear average-case running time and makes its worst-case running time O(nlgn)?

Bucket sort is a sorting algorithm that sorts elements by putting the elements into different buckets and then merging the buckets together, however, if all elements end up in a single bucket (non-uniform distribution) then we need to sort them (usually with insertion sort algorithm) which makes worst-case = $\Theta(n^2)$.

We can use merge sort in the buckets to reduce the running time to O(nlgn).

2. (15pts) Let X be a random variable that is equal to the number of heads in two flips of a fair coin. What is E[X2]? What is E2[X]?

Since X is equal to the number of heads in two flips of a fair coin it can take values 0 no heads, 1 head, 2 heads ie

x=0,1,2

P(x)=for x=0 its ¼, for x=1 its ½ , for x=2 its ¼

Formula, E(X)=∑xP(x)

E[X]=2*¼ +1*½+0*¼= 1

E[X^2] = 4*¼+1*½+0*¼ = 1.5

E^2[X]=E[X].E[X]= 1

3. (10pts) Given n > 2 distinct numbers, you want to find a number that is neither the minimum nor the maximum. What is the smallest number of comparisons that you need to perform?

Pairwise Comparisons:
Divide the n numbers into pairs and compare each pair. This step requires n/2 comparisons if n is even, or (n-1)/2 comparisons if n is odd (one number is left unpaired in this case).

Finding Potential Candidates for Neither Minimum nor Maximum:
Minimum Candidate: Take all the losers of these comparisons (since they cannot be the maximum). If n is even, we have n/2 losers, and if n is odd, we have (n-1)/2 losers.
Maximum Candidate: Similarly, take all the winners (since they cannot be the minimum). The count of winners is the same as that of losers.

Finding the Neither Minimum nor Maximum:
Perform comparisons among the losers to find the minimum of the losers. This takes n/2 - 1 comparisons if n is even, or (n-1)/2 - 1 if n is odd.
Perform comparisons among the winners to find the maximum of the winners. The number of comparisons is the same as above.

Total Comparisons: Summing up, we get:
If n is even: n/2 + 2(n/2-1)=(3n/2)-2
If n is odd: (n-1)/2 + 2{(n-1)/2 - 1} = 3n/2 - 5/2

4. (15pts) Show that RANDOMIZED_SELECT never makes a recursive call to a 0-length array.

In RANDOMIZED_SELECT, a 0-length array never receives a recursive call because:

The algorithm seeks the kth smallest element within the valid range (1 to n) in an array.

Partitioning always produces non-empty subarrays for recursion, as the pivot's placement ensures at least one element remains.

Recursive calls are made on these non-empty subarrays, avoiding any 0-length array scenario.

5. (20pts) Write an iterative version of RANDOMIZED_SELECT.

```
ITERATIVE-RANDOMIZED-SELECT(A, p, r, i)
    while true
        q = RANDOMIZED-PARTITION(A, p, r)
        k = q - p + 1
        if i == k
            then return A[q]
        else if i < k
            then r = q - 1
        else
            p = q + 1
            i = i - k
```

6. (20pts) Show how to implement a queue using two stacks. Analyze the running time of the queue operations.

Implementing a queue using two stacks can be done by having one stack for enqueue operations (stack1) and another stack for dequeue operations (stack2).

Enqueue (push to stack1): Every new element is pushed onto stack1.

Dequeue (pop from stack2):
If stack2 is not empty, pop the element from stack2.
If stack2 is empty, pop all elements from stack1 and push them onto stack2, then pop the top of stack2.

The order of elements in stack2 will be the reverse of their order in stack1, which is what is needed for queue behavior (FIFO - First In, First Out). Popping from stack2 will then give the elements in the correct order.

7. (20pts) Give a $\Theta(n)$ time nonrecursive procedure that reverses a singly linked list of n elements. The procedure should use no more than constant storage beyond that needed for the list itself.

```
REVERSE-LIST(head):
    previous = null
    current = head
    while current is not null:
        next = current.next
        current.next = previous
        previous = current
        current = next
    head = previous
    return head
```