CMSC 351 Summer 2025 Homework 3

Due Thursday 24 July 2025 by 11:59pm on Gradescope.

Directions:

- Homework must be done on printouts of these sheets and then scanned properly, or via Latex, or by downloading, writing on the PDF, and uploading. If you use Latex please do not change the Latex formatting.
- Do not use your own blank paper!
- The reason for this is that Gradescope will be following this template to locate the answers to the problems so if your answers are organized differently they will not be recognized.
- Tagging is automatic, you will not be able to manually tag.

---

0 1 2 3 4

1. Insertion Sort is called on the list `[8,1,7,3,2]`. What will the list look like at the very end of each iteration of the `for` loop? If it helps you can reference the code in Question 3. [5 pts]

	0	1	2	3	4
End of `i=1`	1	8	7	3	2
End of `i=2`	1	7	8	3	2
End of `i=3`	1	3	7	8	2
End of `i=4`	1	2	3	7	8

2. Suppose the list A has all distinct elements and suppose the pair (x, y) is not an inversion. Prove the reverse of the list has an inversion. [10 pts]

   Note: This proof is short, it definitely fits easily in this space!

   Solution:

   Well, an inversion by definition is a pair of indices in a list that are out of order (unsorted might be a better term). If a pair of elements at indices (x,y) do not create an inversion, that means the elements at indices x and y are sorted in relation to each other. (x<y && a[x]>a[y]).
   If two elements in a distinct list are SORTED, then the REVERSE of the list would make that same pair UNSORTED, so x<y but a[x]<a[y], which is an inversion by definition.

3. Here is the pseudocode for Insertion Sort with some print statements and a counter added:

```
function insertionsort(A):
    counter = 0
    n = len(A)
    for i = 1 to n-1 inclusive:
        key = A[i]
        j = i - 1
        while j >= 0 and key < A[j]:
            A[j+1] = A[j]
            j = j - 1
            counter = counter + 1
        end while
        print(A)
        A[j+1] = key
    end for
    print(counter)
end function
```

(a) What value of counter will the second print statement print for each of the following lists:

𝍸 𝍸

[5, 4, 3, 2, 1]

4 5 3 2 1

3 4 5 2 1

2 3 4 5 1

List	Value	
[1,2,3,4,5]	0	[2 pts]
[4,3,1,0,2,6,5]	9	[2 pts]
[5,4,3,2,1]	10	[2 pts]
A reverse-sorted list of length n.	$\binom{n}{2}$	[4 pts]

or $\frac{n(n-1)}{2}$

(b) For each of the following lists, how many times will the first print statement print a list with at least one repeated element?

List	Value	
[1,2,3,4,5]	0	[2 pts]
[1,2,3,5,4]	1	[2 pts]
[5,4,3,2,1]	4	[2 pts]
A reverse-sorted list of length n.	$n-1$	[4 pts]

4. Here is the pseudocode for Binary Search with some code added:

```
function binarysearch(A,TARGET)
    L = 0
    R = n-1
    steps = 0
    while L <= R:
        steps = steps + 1
        C = floor((L+R)/2)  round down
        if A[C] == TARGET
            print(steps)  ← last check is counted too
            return C
        elif TARGET < A[C
            R = C-1
        elif TARGET > A[C
            L = C+1
        end if
    end while
    return FAIL
end function
```

(a) Suppose we call `binarysearch` on the following list: |||| [10 pts]

0 1 2 3 4 5 6 7 8 9

[1,2,3,4,5,6,7,8,9,10]

+1 w/

For each of the following values of `TARGET` what values of `steps` will be printed?

`TARGET`	1	2	3	4	5	6	7	8	9	10
`steps`	3	2	3	4	1	3	4	2	3	4

(b) For a list of length $n = 2^k - 1$ with $k \geq 0$, assuming `TARGET` is in the list, what are the maximum and minimum possible values of `steps`? Answers should either be integers or expressions involving n. [10 pts]

Minimum `steps`	1
Maximum `steps`	lg(n) + 1

first "half index" check

5. Here is the pseudocode for a modified version of Binary Search. Take some time to understand the modification.

```
function binarysearch(A,TARGET)
    L = 0
    R = n-1
    i = 0
    while L <= R:
        i = i + 1
        C[i] = ceiling((L+R)/2)
        print(C[i])
        if A[C[i]] == TARGET
            return C[i]
        elif TARGET < A[C[i]]
            R = C[i]-1
        elif TARGET > A[C[i]]
            L = C[i]+1
        end if
    end while
    return FAIL
end function
```

round up instead of down.

$(4+0)/2 = 2$

$(2+0)/2 = 1$

$(1+0)/2 = 0.5 \rightarrow 1$

0 1 2 3 4 5 6 7

1, 2, 3, 4, 5, 6, 7, 8

$L = 0$

$R = 7$

$(7+0)/2 = 3.5 \rightarrow 4$

(a) Suppose we call binarysearch on a list of length 8 containing all distinct elements and with the TARGET equal to A[0]. What values will the print(C[i]) statement output? You will not need all the spaces below. [4 pts]

start at C[1] X	4	2	1	1			
0	1	2	3	4			

(b) Suppose we call binarysearch on a list of length 2^k with $k \geq 1$. Write down an expression for C[i]. [10 pts]

Solution:

when the length of the list n is 2k, this binarysearch always does max k+1 checks.

$C[1] = 7/2 = 4$

$C[2] = 4/2 = 2$

$C[3] = 2/2 = 1$

$C[4] = 1/2 = 1$

$C[i] = \text{ceiling}\left(\frac{n-1}{2}\right)$

$2 \cdot T(2-2)+1 = (2 \cdot 3)+1$

$= 6+1 = 7$

$2 \cdot T(3-2)+1 = 2 \cdot T(1)+1$

$2 \cdot 3+1 = 7$

6. Consider this recurrence relation:

$$T(n) = 2T(n-2)+1 \quad \text{with } T(0) = T(1) = 3$$

(a) Calculate the following values: [6 pts]

$T(4)$

$= 2 \cdot T(4-2)+1$

$= 2 \cdot T(2)+1$

$= (2 \cdot 7)+1 = 14+1 = 15$

$T(2) =$	7
$T(3) =$	7
$T(4) =$	15

(b) Use digging down to solve the recurrence relation for the case when n is even. Simplify your answer. [10 pts]

Solution:

2T([illegible]-2)+1

$T(n) = 2T(n-2)+1$

$T(n) = 2[2T(n-4)+1]+1 = 2^2 T(n-4)+2+1$

$T(n) = 2^2[2T(n-6)+1]+2+1$

$= 2^3 T(n-6)+2^2+2+1$

$= 2^3 \cdot T(n-6)+2^2+2^1+2^0$

Rec.Rel $= 2^k T(n-2k) + \sum_{i=0}^{k-1} 2^i$

7. For each of the following recurrence relations with base cases given, if we dig down k steps calculate which k (as an expression involving n) would yield the base case. The assumption on n is given which guarantees the base case is always reached.

Relation	Assumption on n	k expression	
$T(n) = 2T(n/3) + n$ with $T(1) = 3$	$n \in \{1, 3, 9, 27, ...\}$	$\log_3 n$	[2 pts]
$T(n) = 3T(n-3)$ with $T(0) = 5$	$n \in \{0, 3, 6, 9, 12, ...\}$	$n/3$	[2 pts]
same but $T(n-1k)$ $T(n) = T(n-1) + n^2 + n$ with $T(\underline{2}) = 7$	$n \in \{2, 3, 4, ...\}$	$-2+n$	[2 pts]
$T(n) = 2T(n-2)$ with $T(1) = 10$	$n \in \{1, 3, 5, 7, 9, ...\}$	$\frac{-1+n}{2}$	[2 pts]
✱ $T(n) = aT(n/b) + cn$ with $T(1) = 2$	$n \in \{1, b, b^2, b^3, ...\}$	$\frac{n}{b}$	[2 pts]
0 $T(n) = 4T(n/2-1)$ with $T(0) = 17$	$n \in \{2, 6, 14, 30, ...\}$	$\sqrt{\frac{n}{2}}$	[5 pts]

$n-k=2$
$-n \quad -n$
$-k=2-n$
$\rightarrow k=-2+n$

Scratch work for (f); If your answer is incorrect then we will attempt to find partial credit in here!

same as ex. from notes (stop at $\frac{n}{3k}=1$)

b/c $T(\square) = T(\frac{n}{3k})$

$3^k = n$

$k = \underline{\log_3 n}$

$$\boxed{\begin{aligned} &0 \quad T(\tfrac{n}{2k}-k) \quad T(0)=17 \\ &\text{stop@} \quad \frac{n}{2k}-k=0 \\ &\frac{n}{2k}=k \\ &\frac{n}{2}=k^2=\sqrt{\frac{n}{2}} \end{aligned}}$$

$n-3k$

so base case $=0$

so $T(0)=5$

and stop when $n-3k=0$
$+3k \quad +3k$
$n=3k$
$\frac{n}{3}=k$

$2T(n-2) \quad T(1)=bc$

✱ $T(n)=aT(\underline{\tfrac{n}{bk}})+cn \quad T(1)=2$

stop @ $\frac{n}{bk}=1$
$+k$

$\boxed{\frac{n}{b}}=k$

$T(n-2k)$ stop @ $n-2k=1$
$-n$
$\frac{-2k}{-2}=\frac{1-n}{-2}$
$=k=\frac{-(1-n)}{2}=k=\frac{-1+n}{2}$