CSCI 4041
Section 001
Kelsey Neis
neis@umn.edu
Due Oct 5

2.3: 2, 3, 4, 5, 7
7.2: 1
7.3: 1
8.1: 1, 3

2.3

2) MERGE (A, p, q, r)

```
n1 = q - p + 1
n2 = r - q
let L[1...n1+1] and R[1...n2+1] be new arrays
for i = 1 to n1
    L[i] = A[p+i-1]
for j = 1 to n2
    R[j] = A[q+j]
i = 1
j = 1
for k = p to r
    if i > L.length
        A[k] = R[j] and j++
    else if j > R.length
        A[k] = L[i] and i++
    else if L[i] ≤ R[j]
        A[k] = L[i] and i++
    else if R[j] ≤ L[i]
        A[k] = R[j] and j++
```

just check if one of the arrays is spent and select the other value until the loop is finished

3) $T(n) = \begin{cases} 2 & \text{if } n = 2, \\ 2T(n/2) + n & \text{if } n = 2^k, \text{ for } k > 1 \end{cases}$ $\quad T(n) = n \lg n$

$k = 2: \quad n = 4$

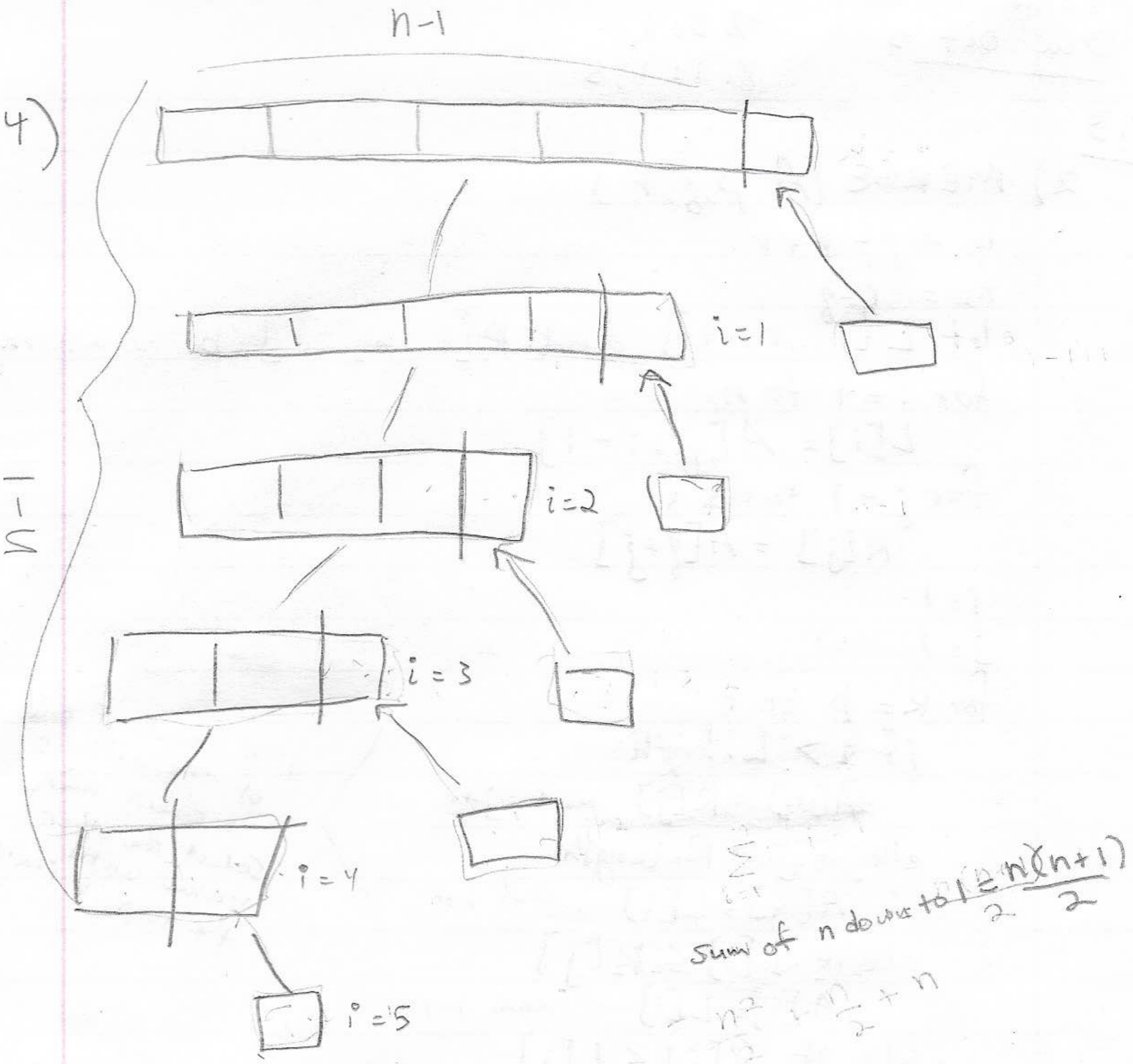
$T(4) = 2\underbrace{T(2)}_{2} + 4 = \underline{8} = \underline{4 \lg 4} = n \lg n$

$k = 3: \quad n = 8$

$T(8) = 2\underbrace{T(8/2)}_{8} + 8 = 24 = 8 \lg 8 = n \lg n$

$T(16) = 2\underbrace{T(16/2)}_{24} + 16 = 64 = 16 \lg 16 = n \lg n$

4)



Divide: compute $n-1 \Rightarrow \Theta(1)$

Conquer: recursively solve the problem for $A[1 \ldots n-1]$
each level $T(n-1)$

Combine: insert 1 element into sorted array
$\Theta(n-1)$

$$\boxed{T(n) = \begin{cases} \Theta(1), & n = 1 \\ T(n-1) + \Theta(n-1), & n > 1 \end{cases}}$$

5) Input: $A = \langle a_1, a_2, \ldots, a_n \rangle$, value $= v$

Output: index i such that $v = A[i]$ or NIL

- eliminate 1/2 array, which does not contain v until v is found. preserve index

```
BinarySearch(A, p, r, v)  → pass in first and last index on first call
1     if A[p] = v
2         return p
3     else if A[r] = v
4         return r
5     else if p=r
6         return NIL
7     midpoint = (r-p)/2
8     if A[midpoint] < v
9         binarySearch(A, midpoint+1, r, v)
10    else
11        binarySearch(A, p, midpoint, v)
```

Lines 1-7 take constant time. The recursive call to binarySearch will be made at most the number of times it takes to pare down the array to one element, so when $n/2^i = 1$,

$$\Rightarrow n = 2^i$$

$$\lg n = i \leftarrow \text{number of times executed}$$

$$T(n) = \Theta(\lg n)$$

7) Input : S of n integers , X

Output : exists 2 elements that sum to x?

1. Create another array whose elements are X − original element value. For example, if S = [1, 2, 3, 4] and X = 6, the new array would be [5, 4, 3, 2]
2. Combine the two arrays and sort them with merge sort.
3. Loop through the larger array, comparing each value to the i−1 value. If a match is found, the function returns true, else false.

In the above example, the combined, sorted array would be : [1, 2, 2, 3, 3, 4, 4, 5]
The duplicates imply that the original array contains both values needed to sum to X. A special case is needed for value X/2, and I might replace it with a sentinel, since a set must not contain duplicates (so it wouldn't be a candidate for making a sum)

$T(n)$: Step 1 : $\Theta(n)$
Step 2: $\Theta(n \lg n)$
Step 3: $2\Theta(n)$

$= 3\Theta(n) + \Theta(n \lg n)$

after a certain time, $n \lg n$ will dwarf $3\Theta(n)$, so $T(n)$ can be just $\underline{\Theta(n \lg n)}$

7.2

1) $T(n) = T(n-1) + \Theta(n)$

$$u_k = u_{k-1} + f(k)$$

$$u_k - u_{k-1} = f(k)$$

$$\sum_{i=1}^{k} u_i - u_{i-1} = \sum f(k)$$

$$\begin{aligned} &\cancel{u_1} - u_0 \\ + &\cancel{u_2} - \cancel{u_1} \\ + &\cancel{u_3} - \cancel{u_2} \\ &\vdots \\ &u_k - \cancel{u_{k-1}} \end{aligned} = u_k - u_0 = \sum f(k)$$

$$u_k = u_0 + \sum f(k)$$

$$T(n) = T(0) + \sum_{i=1}^{n} n = (n+n+n \dots + n) = n^2$$

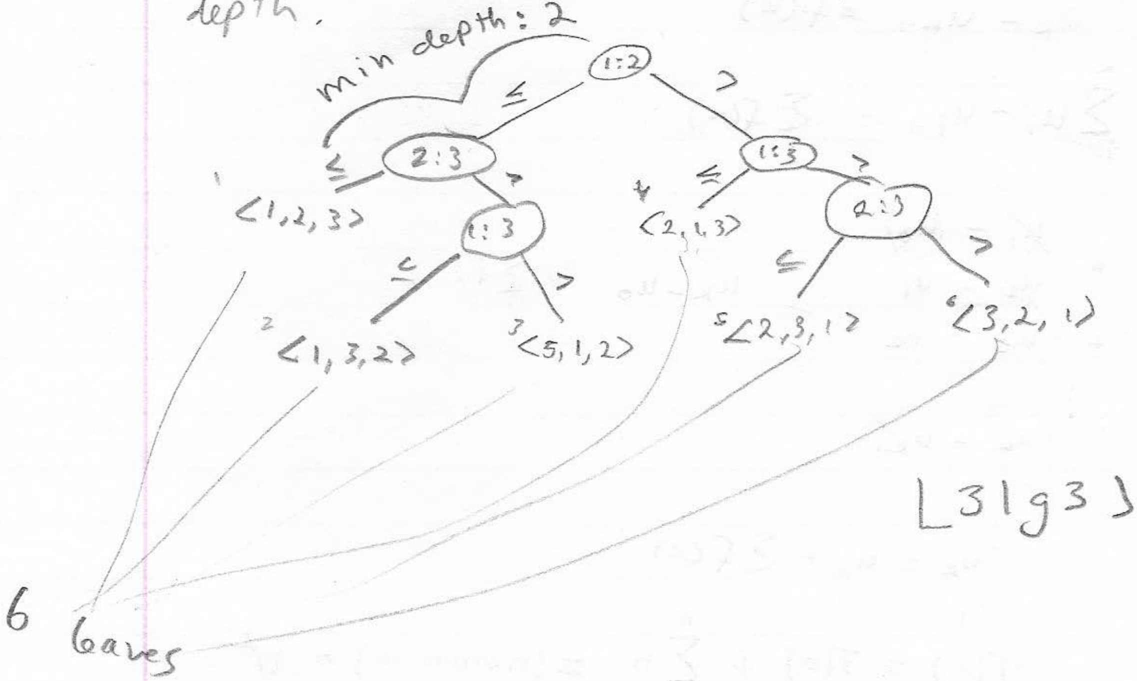
7.3

1) For the worst case to occur, you would have to have the worst partition every time. Since all partitions have an equal chance of happening, you are more likely to have a mix of good and bad partitions.

8.1

1) $\lfloor n \lg(n) \rfloor$ — there are $n!$ leaves, so the height is $\lg(n!)$, which is equal to $n \lg n$. The lower bound of $n \lg n$ will be the smallest depth.

permutations of n



$\lfloor 3 \lg 3 \rfloor = 2$

3) $\lg n!/2 = \lg n! - 1 = n \lg n - 1$ ← half

$\frac{1}{n}$: $\lg \frac{n!}{n} = \lg n! - \lg n = n \lg n - \lg n$

$n(\lg n - \frac{1}{n} \lg n)$ ← cannot be reduced to linear function

$\frac{1}{2^n}$: $\lg \frac{n!}{2^n} = \lg n! - \lg 2^n = n \lg n - n$

$= n(\lg n - 1)$