# COP 3502 10/18/21

## Announcements

1. FRIDAY Recitation starts after 2 pm is cancelled. RP program still exists. You may elect to do it. You definitely can do it during the scheduled recitation time.
2. TURN IN STUDY GROUPS THIS SUNDAY. If you're in a TA one, they'll give me the attendance, etc.

## ~~MERGE~~ SORT
## MERGE

Works because of Merge

Input: 2 sorted lists

Output: a merged sorted list with each item from either input list

n List 1: 2, 6, 8, 9, 15, 17 (6 items)

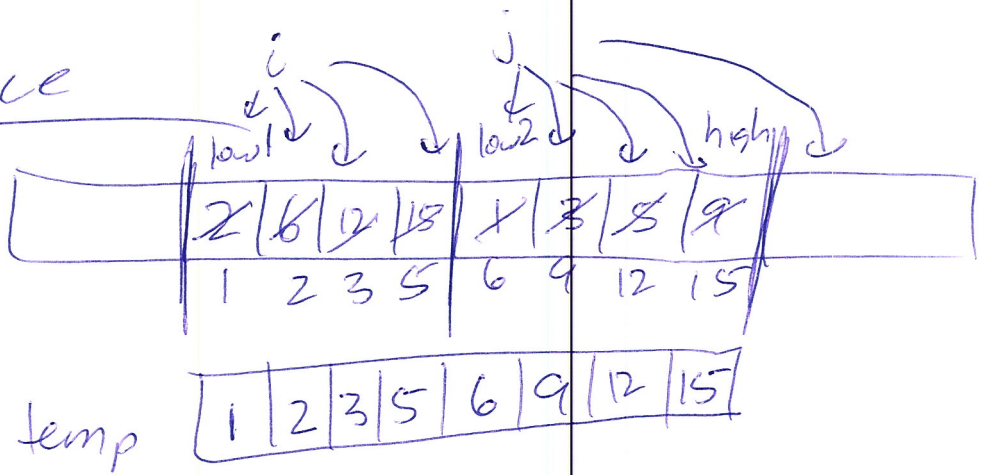
m List 2: 3, 4, 5, 11, 20, 22, 45, 57 (8 items)

Output: 2, 3, 4, 5, 6, 8, 9, 11, 15, 17, 20, 22, 45, 57

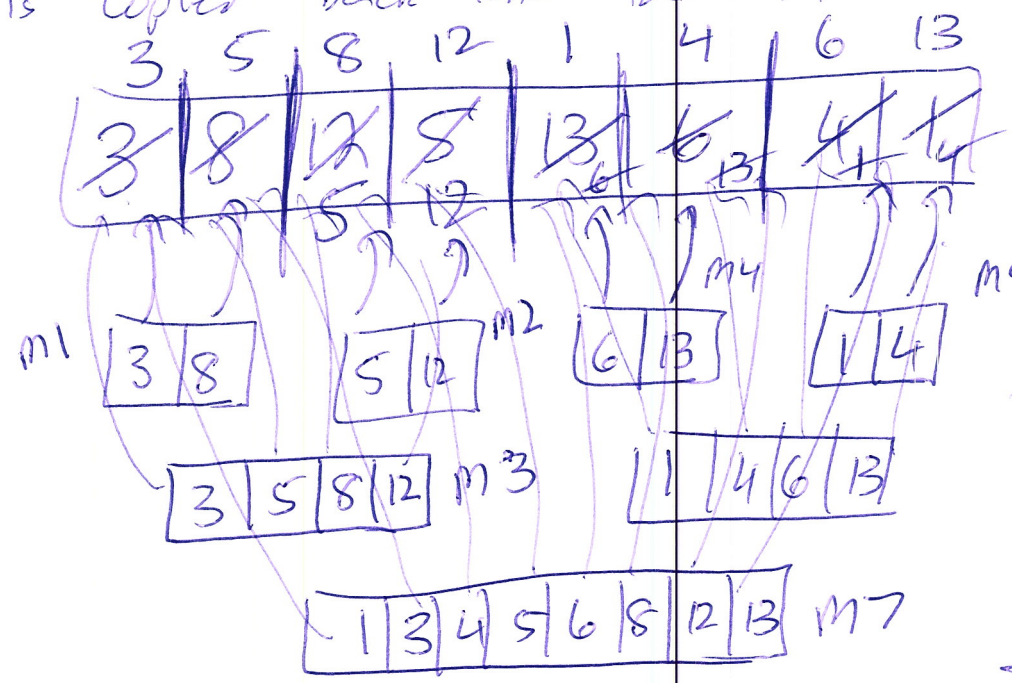
basically iterate through both lists, always take the smallest value + update that index.

Runtime: $O(n+m)$

Practice




2 arrays are side by side in 1 array and indicated by starting/ending indexes + result is copied back into the main array.




~~MS(3,3)~~
~~MS(2,2)~~
MS(2,3)
~~MS(1,1)~~
~~MS(0,0)~~
~~MS(0,1)~~
~~MS(0,3)~~
MS(0,7)

~~MS(7,7)~~
~~MS(6,6)~~
MS(6,7)
~~MS(5,5)~~
~~MS(4,4)~~
~~MS(4,5)~~
~~MS(4,7)~~
MS(0,7)

# Merge Sort

## Merge

Input: 2 sorted list

Output: 1 sorted list w/ each item from the 2 old lists.

LIST 1: 3, 6, 9, 11, 15, 45, 47, 53, 56

LIST 2: 2, 5, 7, 8, 12, 13, 22, 25, 27

OUTPUT: 2, 3, 5, 6, 7, 8, 9, 11, ...

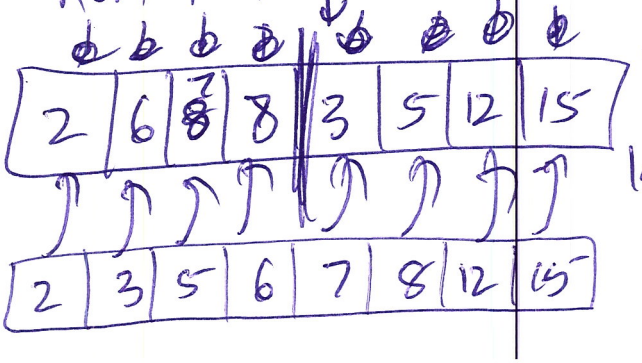
ALG: start 2 ptrs (integer indexes) into both lists at 0, repeatedly copy smaller value into the next slot in the out put array. (Once you run out of a list just copy the other list over.)

- Array out of Bounds
- Test w/ array sizes 1, 2, etc.

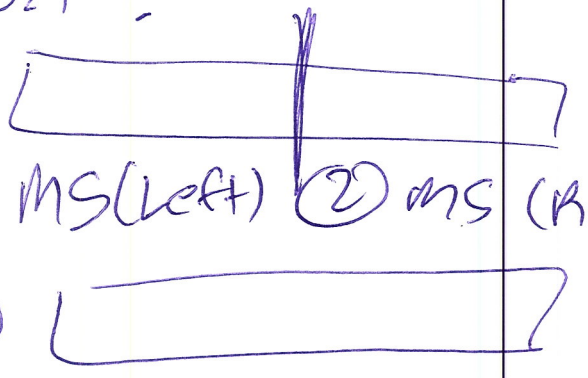
n = size of list 1

m = size of list 2

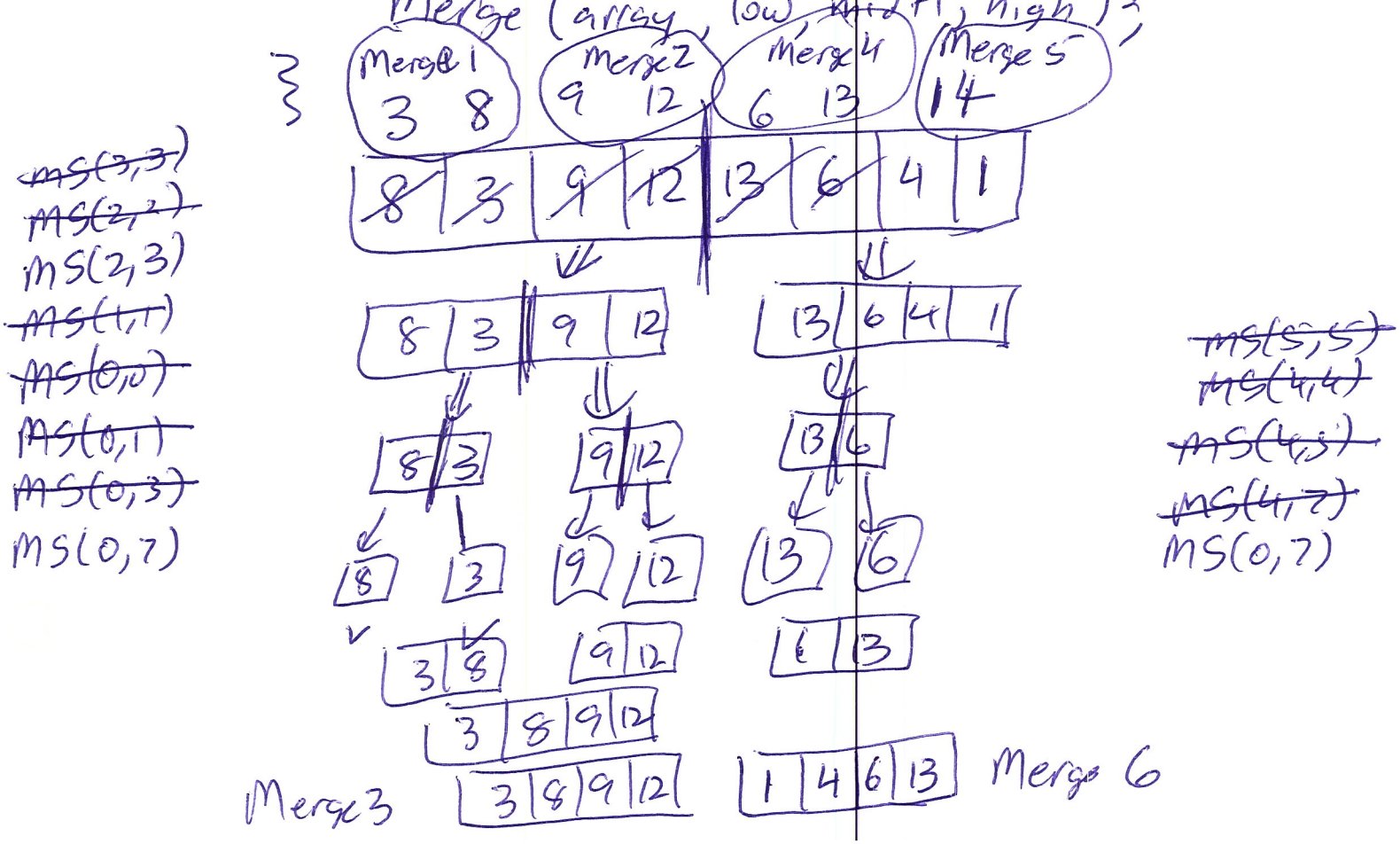
Merge Run time = $O(n+m)$



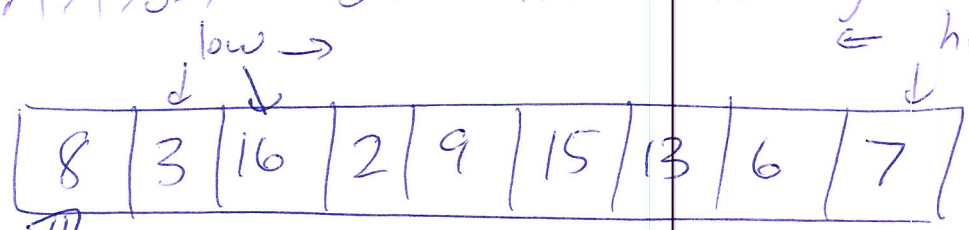
How can we use this tool to sort?




```
Void MergeSort(int * array, int low, int high){
    if (low >= high) return;
    int mid = (low+high)/2;
    MergeSort(array, low, mid);
    MergeSort(array, mid+1, high);
    Merge(array, low, mid+1, high);
}
```



# Partition of an array




Everything < PartElem ⟹ left
Everything > PartElem ⟹ right

Goal: after we're done, everything left PartElem is less than it, everything right of PartElem is greater than it.

```
while (low < high) {

    while (low < end && a[low] < a[part]) low++;
    while (high > part && a[high] >= a[part] high--;
    if (low < high) swap(a, low, high);
}
```

8, 3, 7, 2, 9, 15, 13, 6, 16
8, 3, 7, 2, 6, 15, 13, 9, 16

swap(a, part, high)

6, 3, 7, 2, 8, 15, 13, 9, 16

SORT SORT ✓