Ronald Rounsifer
Professor Cao
CIS 263 03
4/2/2018

Homework 6

1) Iterative merge sort

```
def merge_sort_iterative(numbers):
    for x in range(0, len(numbers)):
            x *= 2
            for y in range(0,len(numbers), x):
                    temp1 = [y:y+(x/2)]
                    temp2 = [y+(x/2):x-y]
                    low = 0
                    mid = 0
                    while low < len(temp1) AND mid < len(temp2):
                            if temp1[low] < temp2[mid]:
                                    mid += 1
                            elif temp1[low] > temp2[mid]:
                                    temp1[low], temp2[mid] = temp2[mid], temp1[low]
                                    low += 1
                    numbers[y:y+(x/2)], numbers[y+(x/2):x-y] = temp1, temp2
    return numbers
```

This merge sort will have a runtime complexity of O(NlogN) because the outer loop window sizes grow at a power of 2 so X has logN iterations. Even though there are multiple windows that are covered by the inner loop all of the windows for a given X cover the input array meaning that the inner loop is O(N). By combining these two loops we get O(NlogN).

Ronald Rounsifer
Professor Cao
CIS 263 03
4/2/2018

2) Simultaneous min / max

```
def recursive_min_max(numbers):
    sorted = merge_sort(numbers)
    min = sorted[0]
    max = sorted[:-1]
    return min, max


def merge_sort(numbers):
    if len(numbers) < 2:
            return numbers

    middle = len(numbers)/2
    left = numbers[:middle]
    right = numbers[middle:]
    return merge(left, right)


def merge(left, right)
    if not(left) or not(right):
            return left or right
    merged = []
    x, y = 0, 0
    while len(merged) < len(left) + len(right):
            if left[x] < right[y]:
                    merged.append(left[x])
                    x += 1
            elif left[x] > right[y]:
                    merged.append(right[y])
                    y += 1
            if x == len(left) or y == len(right):
                    merged.extend(left[x:] or right[y:])
                    break
    return merged
```

Ronald Rounsifer
Professor Cao
CIS 263 03
4/2/2018

3a) Two numbers that sum up the inputted k $O(N^2)$

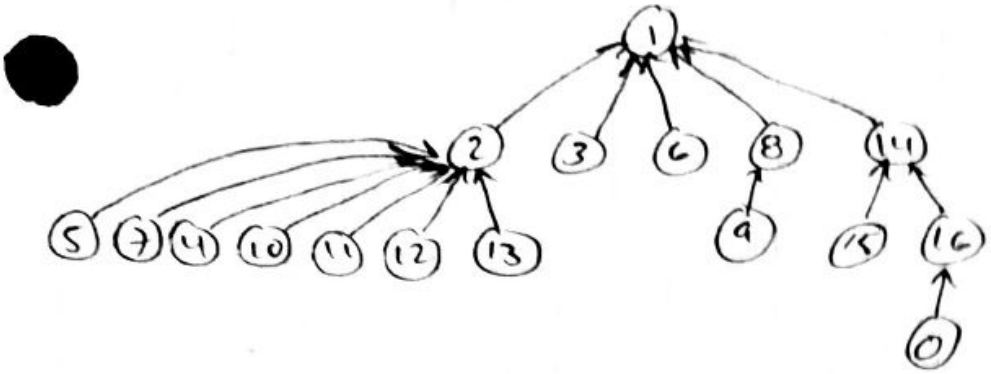
```
def sum_equals_k(numbers, k):
        for (i = 0; i < len(numbers); i++)
                for(j = 0; j < len(numbers); j++)
                        if numbers[i] + numbers[j] == k:
                                return yes
        return no
```

3b) Two numbers that sum up the inputted k O(NlogN)

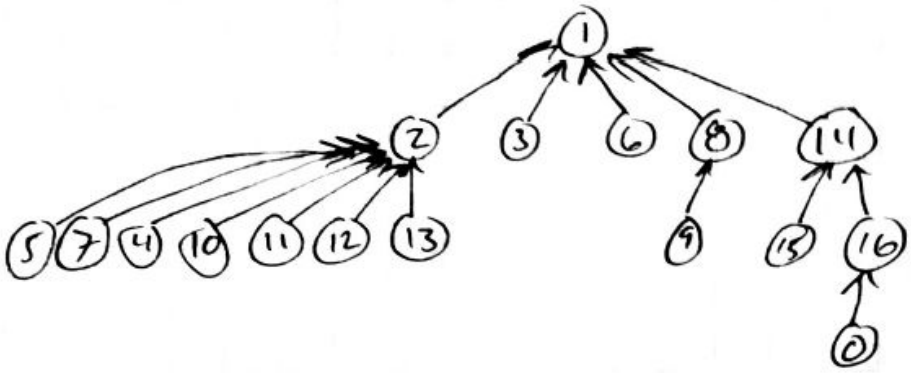
```
def sums_equals_k_sorted(numbers, k):
   numbers = merge_sort(numbers) # function from Question 2
   left = 0
   right = len(numbers)

   while  left < right:
        if numbers[left] + numbers[right]:
                return yes
        elif numbers[left] + numbers[right] < k:
                left += 1
        else:
                right -= 1
   return no
```

Ronald Rounsifer
Professor Cao
CIS 263 03
4/2/2018

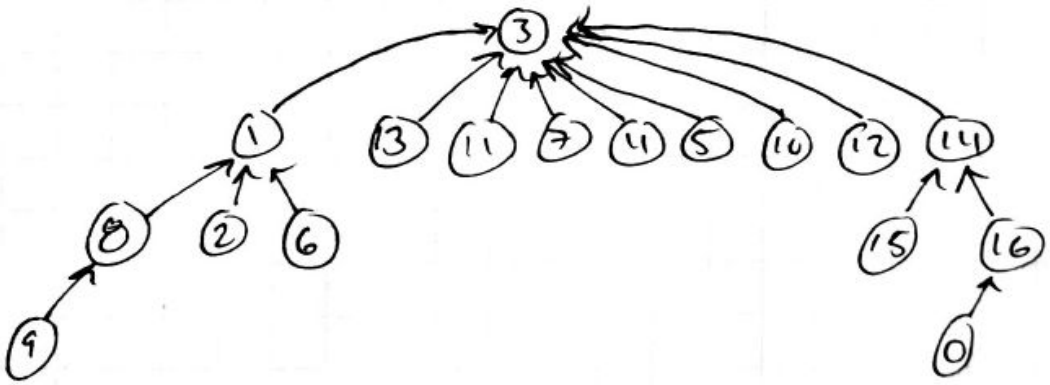
4) a)



b)

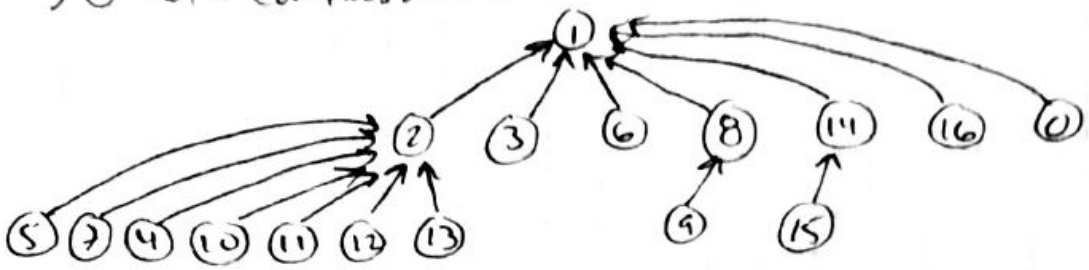


c)

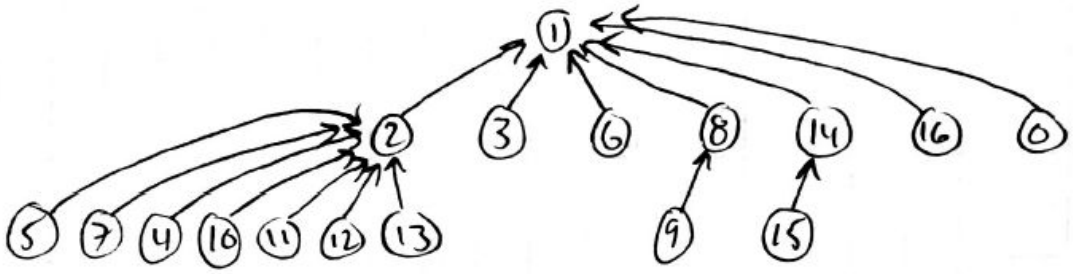


Ronald Rounsifer
Professor Cao
CIS 263 03
4/2/2018

5) (a) Path Compression of 4a



(b) Path compression of 4b



(c) Path compression of 4c