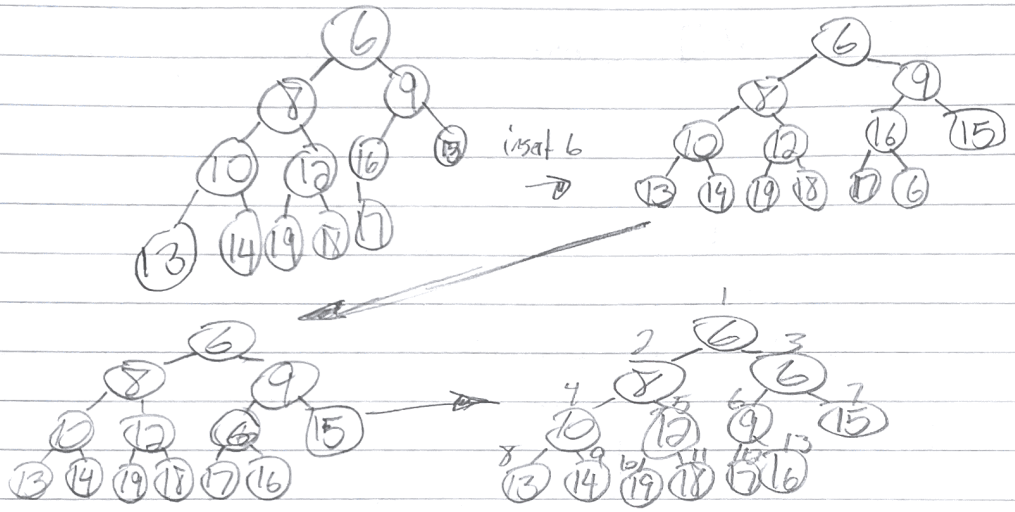
COMP 3270 Homework 3

1.) Given Array: A=[6,8,9,10,12,16,15,13,14,19,18,17]
Show array after using algorithm Min-Heap-Insert(A,6)

- We know that for a Max-Heap, the largest # will be the root node.
- Therefore a Min-Heap will have the smallest # up top @ root node



A=[6,8,6,10,12,9,15,13,14,19,18,17,16]

2.) Efficient $O(n \lg n)$ algorithm for taking collection of objects & converting them to a set

- Looking for $O(n \lg n)$ time
  - If we sort the collection first we can use mergesort $O(\lg n)$ time & get all duplicates next to each other.
  - Take list & scan for duplicates & remove them $O(n)$
  - Combining 2 steps results in $O(n \lg n)$ run-time

array start finish

5.) Partition(A, p, r)

```
x = A[r]                    last element
i = p-1                     One before first element
for j = p to r-1            From start to one before last
    if A[j] ≤ x             If jth element less than equal to last
        then i = i+1        Iterate i
        swap A[i] and A[j]  Switch ith & jth element
swap A[i+1] and A[r]
return i+1
```

- The goal of our partition is to get a pivot value that will have an array in 2 parts. One side greater than & one side less than
- The last element of the array is our pivot value

A=[0,3,9,4,8,5,7,6], p=1, r=8, k=2

pivot=6

partition Array: A=[3,4,5,6,10,9,8,7]

```
Quickselect(A, p, r, k)
  if p=r then return A[p]
  else
  q = Partition(A, p, r)
  pivotDistance = q-p+1          - How far pivot is from front
  if k=pivotDistance then        - if pivot is kth smallest
     return A[q]
  else if k < pivotDistance      - if k is on left side of pivot
       return Quickselect(A, p, q-1, k)
  else
       return Quickselect(A, q+1, r, k-pivotDistance)   - k is on right
                                                        - Adjust k for new array
```

5.) 1st iteration: Quickselect(A, 1, 8, 2)

p r k

pivot A: [3,4,5,6,10,9,8,7]

q=4, pD=4 → 2<4

2nd: Quickselect(A, 1, 3, 2) A=[3,4,5]

p r k

pivot A: [3,4,5]

q=3, pD=3 → 2<3

3rd: Quickselect(A, 1, 2, 2) A=[3,4]

pivot A: [3,4]

q=2, pD=2 → 2=2 → Meets 2nd Base Case

Return A[q]=4

b.) T(first base case) = 7

6.) Counting Sort for A=[9,6,10,7,16,17,13,14,12,9]

input range from 0-19

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
C	0	0	0	0	0	0	1	1	0	1	1	0	1	1	1	0	1	1		1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
C	0	0	0	0	0	0	1	2	2	3	4	4	5	6	7	7	8	9	9	10

	0	1	2	3	4	5	6	7	8	9
B	6	7	9	10	12	13	14	16	17	19

IX.) Radix Sort applied on A = [4567, 3210, 2345, 4321, 5678]

4567	3210	3210	3210	2345
3210 →	4321 →	4321 →	4321 →	3210
2345	2345	2345	2345	4321
4321	4567	4567	4567	4567
5678	5678	5678	5678	5678

X.) Bucket Sort: length(A) = 15 range of inputs is from 0 to 14.

Bucket #:	Value
0	0 → 1/15
1	1/15 → 2/15
2	2/15 → 3/15
3	3/15 → 4/15
4	4/15 → 5/15
5	5/15 → 6/15
6	6/15 → 7/15
7	7/15 → 8/15
8	8/15 → 9/15
9	9/15 → 10/15
10	10/15 → 11/15
11	11/15 → 12/15
12	12/15 → 13/15
13	13/15 → 14/15
14	14/15 → 15/15

- We will use Radix Sort since our base for the number system can be represented by $b = O(\lg n)$, which the textbook states "that is often the case"

11.) Disjoint Set

a.)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	1	14	3	3	3	1	1	8	3	3	3	3	1	14	16	16	19	20	1

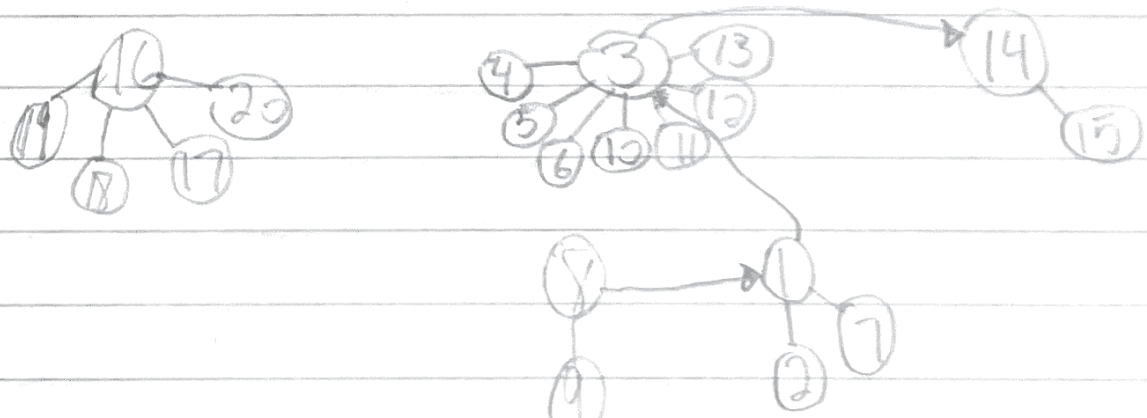
b.) Union by Height

- Make the shallower tree the subtree of the deeper tree
- When we have trees of same height, Union first tree to second
- Root cells will hold negative value of tree depth

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
			3	3	3				3	3	3	3		15	-1	16	16	16	16

c.) Union by Size

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

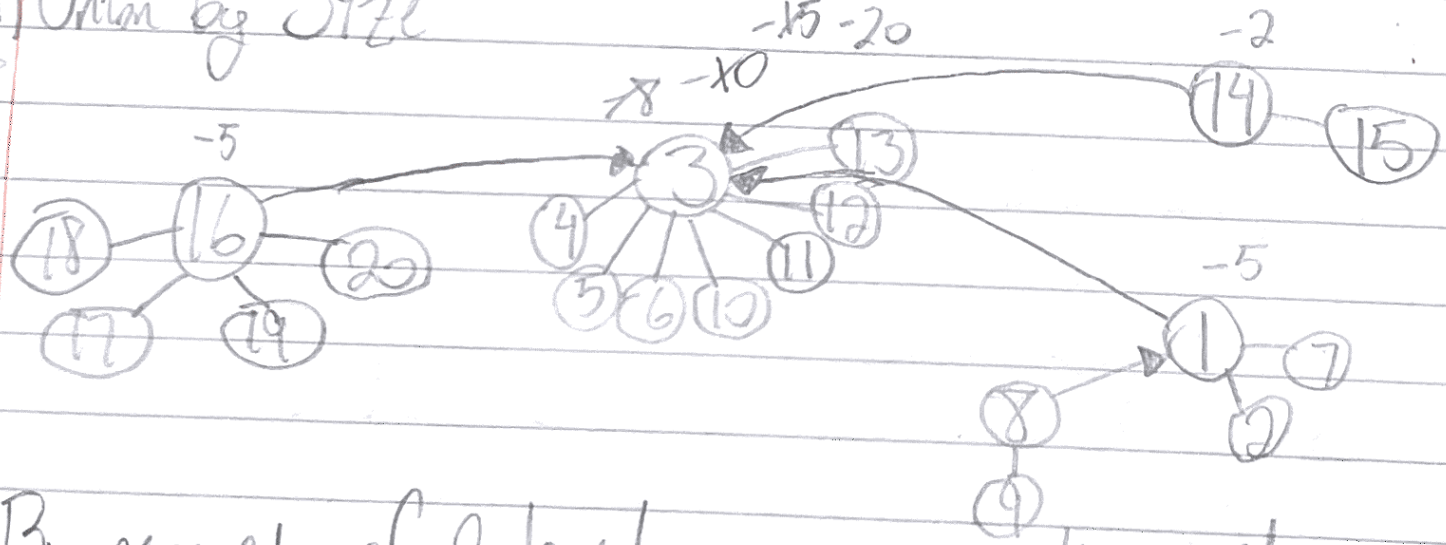


- By given rules, if given trees with same height, then 2nd tree will become child of the root of the 1st tree.

Resulting Array:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-4	1	14	3	3	3	1	1	8	3	3	3	3	1	14	1	16	16	16	16

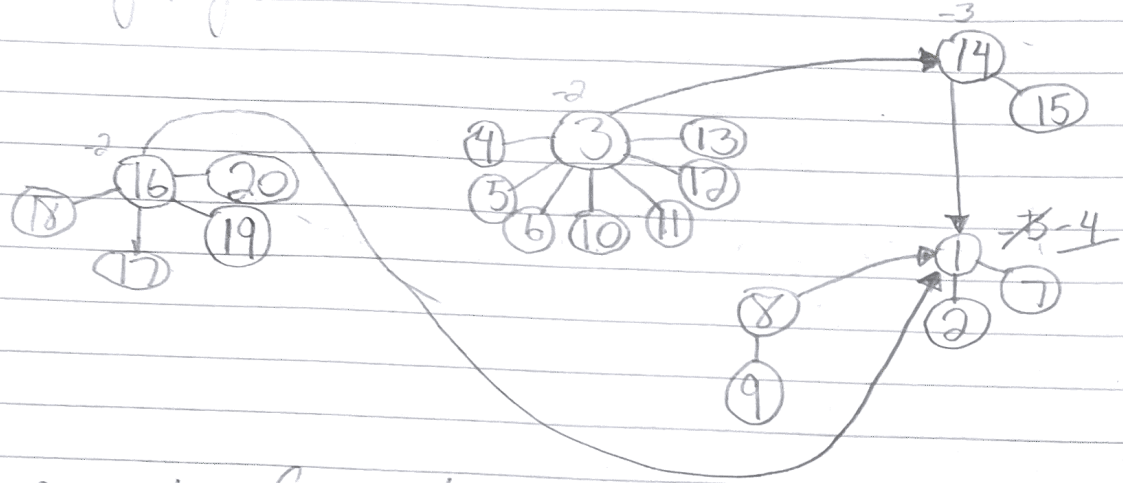
c) Union by Size



- By given rules, if 2 trees have same size, then 2nd tree becomes child of the root of 1st tree.

Resulting Array:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	1	-2	3	3	3	1	1	8	3	3	3	3	3	14	3	16	16	16	16

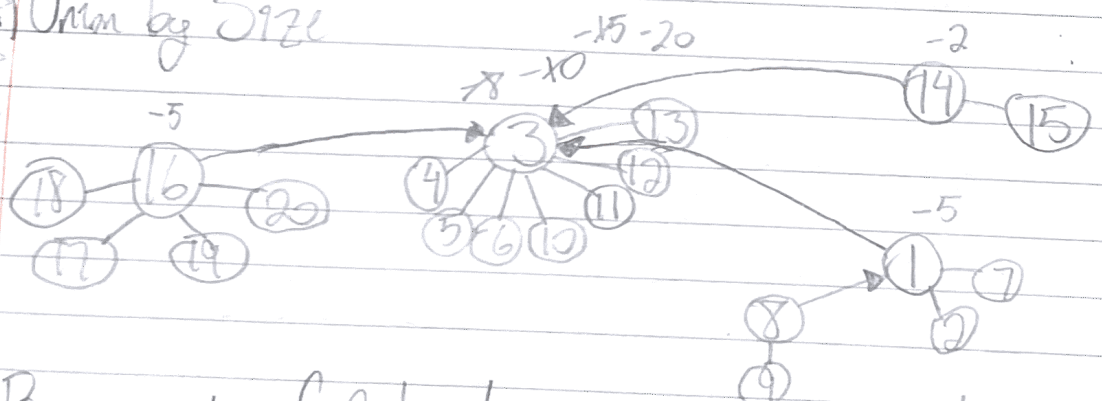


- By given rules, if given trees with same height, then 2nd tree will become child of the root of the 1st tree.

Resulting Array:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-4	1	14	3	3	3	1	1	8	3	3	3	3	1	14	1	16	16	16	16

c) Union by Size

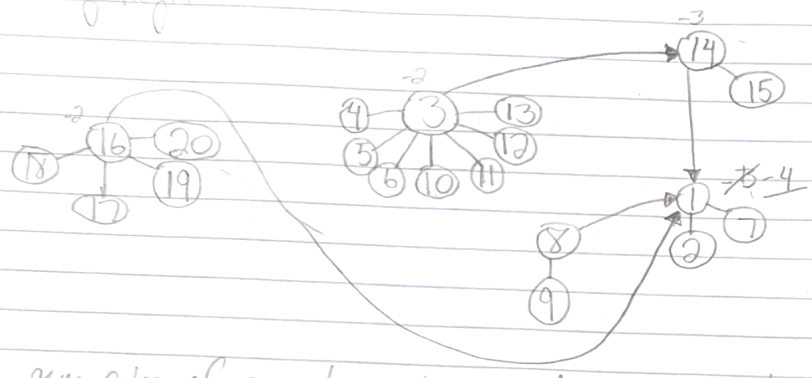


- By given rules, if 2 trees have same size, then 2nd tree becomes child of the root of 1st tree.

Resulting Array:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	1	-2	3	3	3	1	1	8	3	3	3	3	3	14	3	16	16	16	16

b) Union by Height

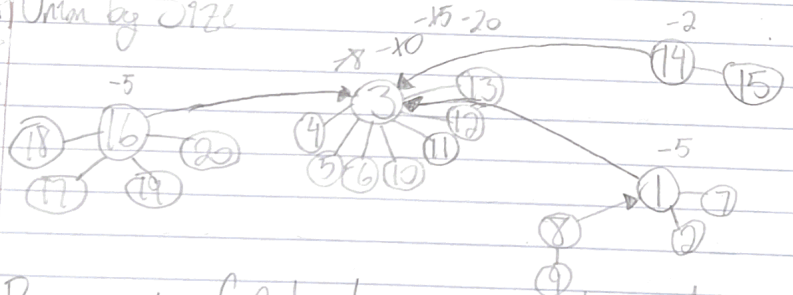


By given rules, if given trees with same height, then 2nd tree will become child of the root of the 1st tree.

Resulting Array:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-4	1	14	3	3	3	1	1	8	3	3	3	3	1	14	1	16	16	16	16

c) Union by Size

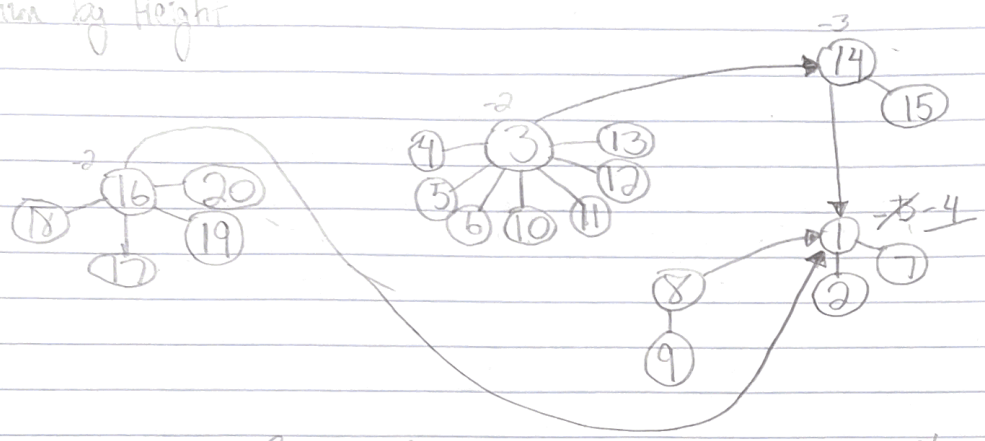


By given rules, if 2 trees have same size, then 2nd tree becomes child of the root of 1st tree.

Resulting Array:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	1	-2	3	3	3	1	1	8	3	3	3	3	3	14	3	16	16	16	16

b) Union by Height

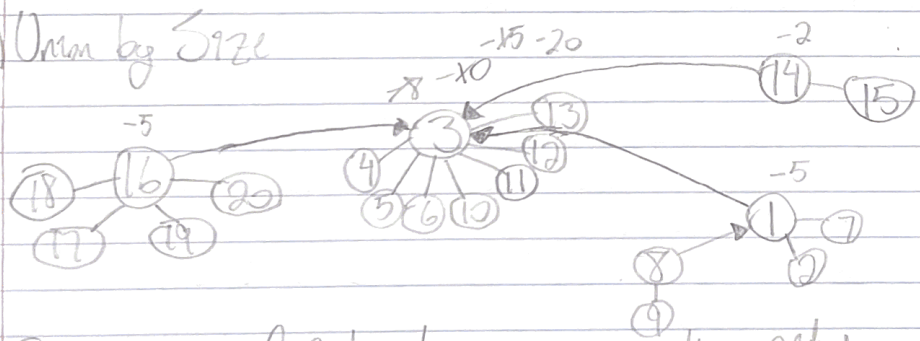


- By given rules, if given trees with same height, then 2nd tree will become child of the root of the 1st tree.

Resulting Array:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-4	1	14	3	3	3	1	1	8	3	3	3	3	1	14	1	16	16	16	16

c) Union by Size

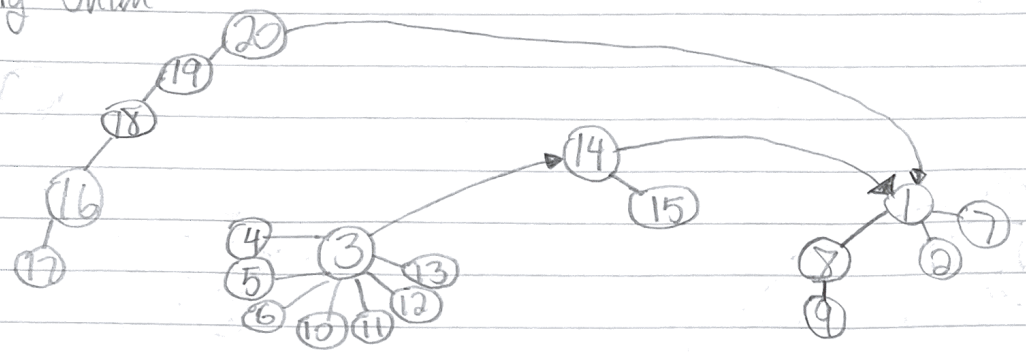


- By given rules, if 2 trees have same size, then 2nd tree becomes child of the root of 1st tree.

Resulting Array:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	1	-2	3	3	3	1	1	8	3	3	3	3	3	14	3	16	16	16	16

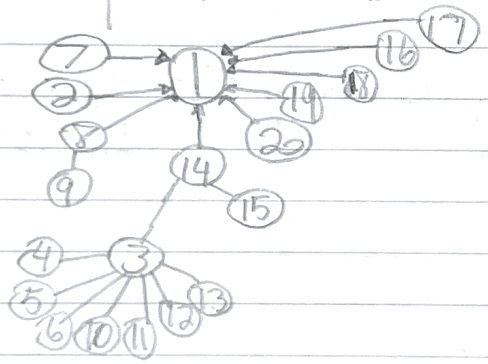
a.) Arbitrary Union



Resulting Array:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	1	14	3	3	3	1	1	8	3	3	3	3	1	14	18	16	19	20	1

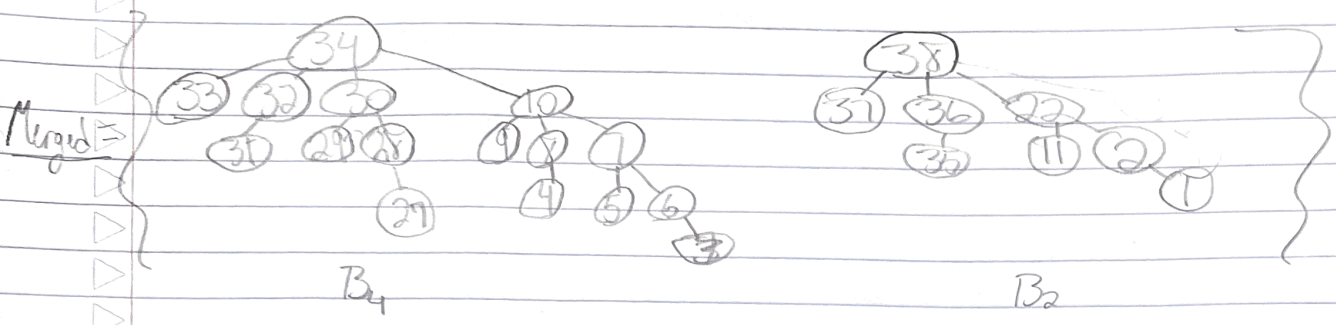
b.) Path Compression on Deepest Node From (a)



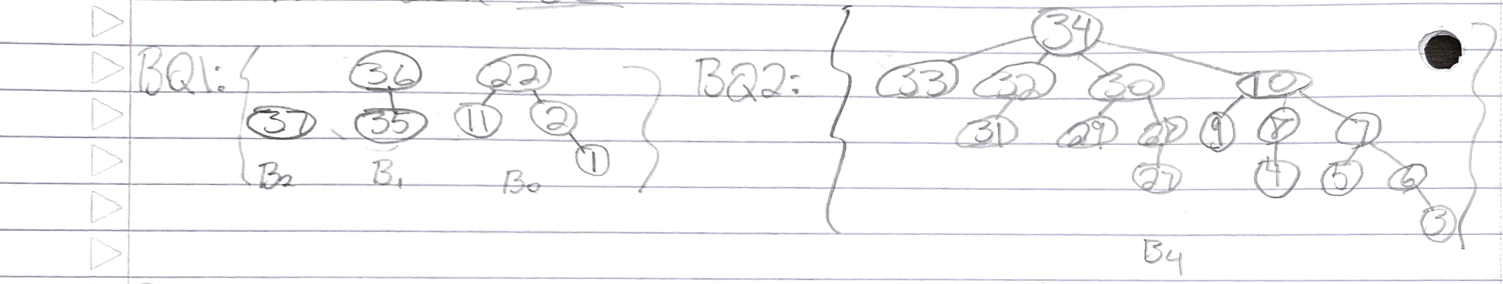
Resulting Array:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	1	14	3	3	3	1	1	8	3	3	3	3	1	14	1	1	1	1	1

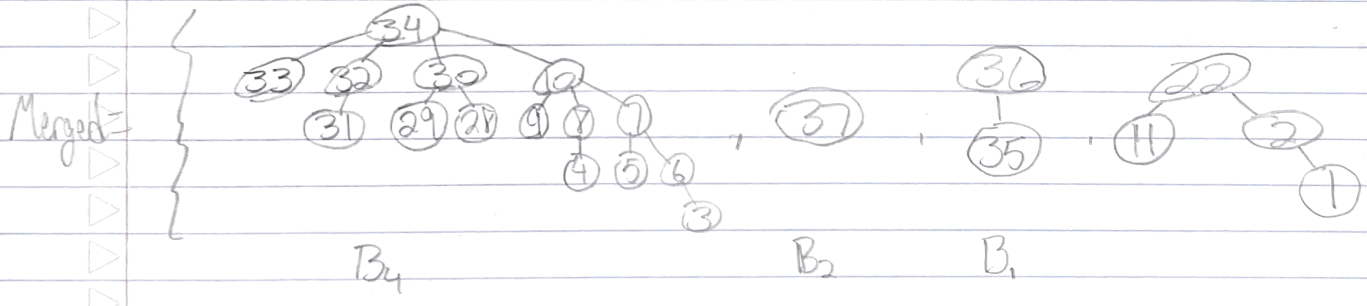
12.) BQ1 + BQ2 = 1011 + 1001 = 10100 → {B_4 B_2}



Extract Max: return 38



BQ1 + BQ2 = 00111 + 10111 = 10111 → {B_4 B_2 B_1 B_0}



Worst case: $T(n) = \begin{cases} 7 & ; n+1 \\ T(n-1)+20n+15 & ; n>1 \end{cases}$

Best case: $T(n) = \begin{cases} 7 & ; n+1 \\ T(\frac{n}{2})+20n+15 & ; n>1 \end{cases}$