Homework 3

Comp 3270

Haden Stuart / has0027

**Problem 1.)**

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

**Problem 2.)**



A = [10, 3, 9, 4, 8, 5, 7, 6], p=1, r=8, k=2

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

QuickSelect(A, 1, 3, 2)

A = [3, 4, 5, 6, 8, 9, 7, 10], p=1, r=3, k=2

After: A = [3, 4, 5, q, 6, 8, 9, 7, 10]

QuickSelect(A, 1, 2, 2)

A = [3, 4, 5, 6, 8, 9, 7, 10], p=1, r=2, k=2

After: A = [3, 4, q, 5, 6, 8, 9, 7, 10]

A[q] = 4



* The first base case checks if the starting and ending indexes are the same which would mean the array has only one element.
* If this is the case, then the algorithm will execute the first step: return A[p].
* T(first base case) = **7**
* The second base case checks if the k value is equal to the pivot meaning the kth smallest value would be found.
* If this is the case, then the algorithm will run step 1 but won’t return a value then it will continue executing steps 2, 3, 4, and 5.
* T(second base case) = 3 + (20n + 1) + 5 + 7 = **20n + 16 = O(n)**
* If it is not a base case, then the algorithm will run through each step, skipping the return value on step 1 and step 5, then it will either execute steps 6 and 7 or just step 8 depending on the value of k and pivotDistance.
* For the worst case: T(n) = 3 + (20n + 1) + 5 + 3 + 3 + T(n-1)

= **n[T(n-1) + 20n + 15] = O(n2)**

* For the best case: T(n) = 3 + (20n + 1) + 5 + 7 = **20n + 16 = O(n)**

**Problem 3.)**

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

A = [19, 6, 10, 7, 16, 17, 13, 14, 12, 9]

B = [6, 7, 9, 10, 12, 13, 14, 16, 17, 19]

C = [0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 3, 4, 4, 5, 6, 7, 7, 8, 9, 9]

**Problem 4.)**

* 1st pass:

3210

4321

2345

4567

5678

* 2nd pass:

3210

4321

2345

4567

5678

* 3rd pass:

3210

4321

2345

4567

5678

* 4th pass:

2345

3210

4321

4567

5678

**Problem 5.)**

**Bucket0:** (0, 0.066)

**Bucket1:** (0.066, 0.133)

**Bucket2:** (0.133, 0.200)

**Bucket3:** (0.200, 0.266)

**Bucket4:** (0.266, 0.333)

**Bucket5:** (0.333, 0.400)

**Bucket6:** (0.400, 0.466)

**Bucket7:** (0.466, 0.533)

**Bucket8:** (0.533, 0.600)

**Bucket9:** (0.600, 0.666)

**Bucket10:** (0.666, 0.733)

**Bucket11:** (0.733, 0.800)

**Bucket12:** (0.800, 0.866)

**Bucket13:** (0.866, 0.933)

**Bucket14:** (0.933, 1.000)

**Bucket0: [**0/n, 0 + 1/n**]**

**Bucket1: [**1/n, 1 + 1/n**]**

**Bucket(n-2): [**(n-2)/n, (n-1)/n**]**

**Bucket(n-1): [**(n-1)/n, n/n**]**

**Problem 6.)**

a. Performed arbitrarily. Make the second tree the child of the root of the first tree.

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

b. Performed by height. If trees have same height, make the 2nd tree the child of the root of the 1st tree.

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

c. Performed by size. If trees have the same size, make the second tree the child of the root of the first tree.

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

d. For the solution to part a, perform a find with path compression on the deepest node and show the array P after find finishes.

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

**Problem 7.)**

From the provided binomial trees we get:

* First tree = 1110 = 14
* Second tree = 1010 = 10
* Adding these we get 11000 = 24

So, the new binomial queue will have a size of 24 nodes.