A picture containing text

Description automatically generated

**CS435**

*Prof. Emdad Khan*

*September 2019*

*Lab#9*

***Group 1***

*Group members:*

*Asad Ali Kanwal*

*Aser Ahmad Ibrahim Ahmad*

*Jean Wilbert Volcy*

*Zayed Hassan*

1. Problem 1

A BST with the described sequence has the all elements as left branches of their roots, and all elements as right branches of their roots. Also, element is its root. The tree should look as the shape below:

Since

And since any search will lead to going through exactly one branch of the two above (if the searched value is < the search will be limited to the left branch, and otherwise limited to the right branch), then the search and insertions will be similar to applying a search/insertion to a linked list, which has a worst case running time of .

Since the length of any branch above is equal to , then worst case running time is , which is also .

1. Problem 2: Insertion for red-black tree for values: 3, 2, 1, 4, 5, 6.
2. Insert 3, 2:
3. Insert 1:

***X***

***P***

***G***

Red-red violation due to an outer grandchild.

* Change color of P & G:

***X***

***P***

***G***

* Rotate P & G, lifting X:
* Parent node 2 has both children red. Change colors of 1 & 3:

1. Insert 4:

* All black-red rules check out. Continue.

1. Insert 5:

***X***

***P***

***G***

Red-red violation due to an outer grandchild:

* Change color of P & G:

***X***

***P***

***G***

* Rotate G & P, lifting X:
* All children of 4 are red. Flip colors of 4, 3, 5:

1. Insert 6:

* All black-red rules check out. Insertion ended.

1. Problem 3: array-based Heap Sort of the array: A = [1, 4, 3, 9, 12, 2, 4]:

*Phase I: Heapification:*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 3 | 9 | 12 | 2 | 4 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

No upheaping needed.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 3 | 9 | 12 | 2 | 4 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

Swap A[1] = 4 with A[(1-1)/2] = A[0] = 4 and take next element:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | 1 | 3 | 9 | 12 | 2 | 4 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

No upheaping needed. Take next element:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | 1 | 3 | 9 | 12 | 2 | 4 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

Swap A[3] = 9 with A[(3-1)/2] = A[1] = 1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | 9 | 3 | 1 | 12 | 2 | 4 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

Swap A[1] = 9 with A[(1-1)/2] = A[0] = 4 and take next element:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 9 | 4 | 3 | 1 | 12 | 2 | 4 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

Swap A[4] with A[(4-1)/2] = A[1] = 4:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 9 | 12 | 3 | 1 | 4 | 2 | 4 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

Swap A[1] = 12 with A[(1-1)/2] = A[0] = 9 and take next element:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 12 | 9 | 3 | 1 | 4 | 2 | 4 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

No upheaping needed. Take next element:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 12 | 9 | 3 | 1 | 4 | 2 | 4 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

Swap A[6] = 4 with A[(6-1)/2)] = A[2] = 3:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 12 | 9 | 4 | 1 | 4 | 2 | 3 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

No more upheaping is needed.

*Phase II: In-place sorting:*

Take out A[0] = 12 and replace it with A[n-1] = A[6] = 3:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 9 | 4 | 1 | 4 | 2 | \_\_ |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

Downheaping A[0] = 3:

Swap A[0] = 3 with the largest child, A[(2\*0)+1] = A[1] = 9:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 9 | 3 | 4 | 1 | 4 | 2 | \_\_ |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

Swap A[1] = 3 with the largest child, A[(2\*1)+2] = A[4] = 4:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 9 | 4 | 4 | 1 | 3 | 2 | \_\_ |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

No more downheaping is possible. Let temp = A[0] = 9 and let A[0] = 2:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 4 | 4 | 1 | 3 | \_\_ | 12 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

Swap A[0] = 2 with the largest child, A[(2\*0)+1] = A[1] = 4:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | 2 | 4 | 1 | 3 | \_\_ | 12 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

Swap A[1] = 2 with the largest child, A[(2\*1)+2] = A[4] = 3:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | 3 | 4 | 1 | 2 | \_\_ | 12 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

No more downheaping possible. Let A[5] = temp = 9, let temp = A[0] = 4, let A[0] = A[4] = 2:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 1 | \_\_ | 9 | 12 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

Swap A[0] = 2 with its largest child, A[(0\*2)+2] = A[2] = 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | 3 | 2 | 1 | \_\_ | 9 | 12 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

No more downheaping possible. Let A[4] = temp = 4, temp = A[0] = 4, A[0] = A[3] = 1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 2 | \_\_ | 4 | 9 | 12 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

Swap A[0] with its largest child, A[0\*2+1] = A[1] = 3:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 1 | 2 | \_\_ | 4 | 9 | 12 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

No more downheaping is possible. Let A[3] = temp = 4, temp = A[0] = 3, A[0] = A[2] = 2:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | \_\_ | 4 | 4 | 9 | 12 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

No more downheaping is possible. Let A[2] = temp = 3, temp = A[0] = 2, A[0] = A[1] = 1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | \_\_ | 3 | 4 | 4 | 9 | 12 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

No more downheaping is possible. Let A[1] = temp = 2. Since only one element is left, add it to the sorted array

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 4 | 9 | 12 |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* |

Sorted array is: [1, 2, 3, 4, 4, 9, 12].

1. Problem 4:

**Algorithm** subsetSum (S, k)

***Input:*** Set S of positive integers, positive integer k

***Output:*** true if there is a subset of S whose sum is k; false, otherwise.

sum = 0

***for*** (i = 0 to S.size – 1) ***do***

***for*** (j = 0 to i) ***do***

***if*** (sum + S[j] ≤ k) ***then*** sum += S[j]

***if*** (sum = k) ***then*** return true

return false