Escreva uma sequência com 15 valores inteiros em ordem aleatória.

3 - Faça a remoção do elemento raiz e mostre como fica o heap após a remoção.

3.1 - Repita o item 3 mais 4 vezes (ou seja, serão feitas 5 remoções).

4 - Represente em um vetor como ficou o heap após as 5 remoções.

** Utilize o tipo de arquivo que preferir (slide, planilha, diagrama, etc), ou desenhe em seu caderno e envie a sequência de imagens em PDF.

Heap Máximo

1 - Construa um heap máximo com estes valores, inserindo-os um a um conforme aparecem na sua lista.

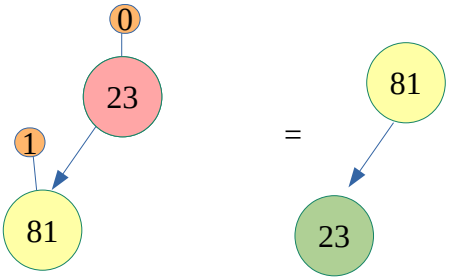
- Values (**23 - 81 - 58 - 16 - 91 - 95 - 37 - 5 - 65 - 70 - 7 - 96 - 15 - 92 – 39**)
- Sempre que o operador > aparecer irá ter balanceamento na árvore heap, do contrário a inserção estará no lugar correto.
- Se o operador < aparecer o vetor permanece com o elemento na posição de inserção.
- n= n - 1

0→n															
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Inserção dos números no vetor criado e representação da árvore e seu balanceamento:

1º número: 23

0→n	23														
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14



2º número: 81

81 > 23

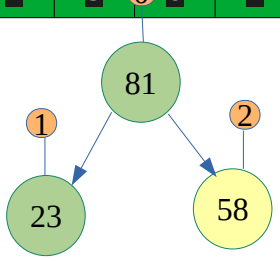
0→n	23	81													

indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

0→n	81	23													
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

3º número: 58

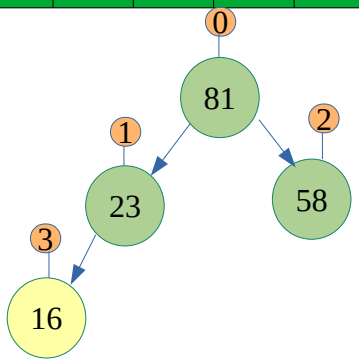
58 < 81



0→n	81	23	58												
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

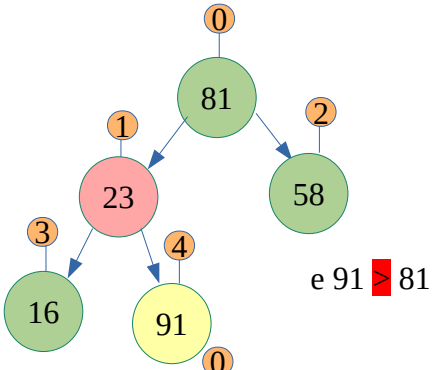
4º número: 16

16 <23

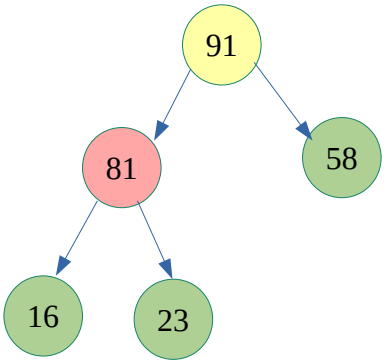


0→n	81	23	58	16											
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

5º número: 91

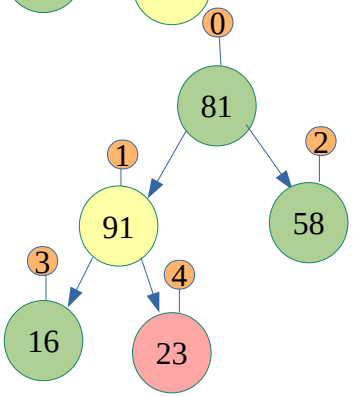


e 91 > 81



5º número: 91

91 > 23



0→n	81	23	58	16	91										

indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

1ª iteração

0→n	81	91	58	16	23										
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

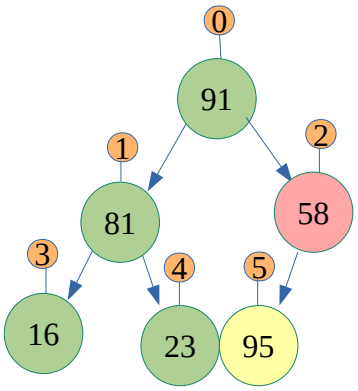
2ª iteração

0→n	91	81	58	16	23										
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

6º número: 95

0→n	91	81	58	16	23	95									
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

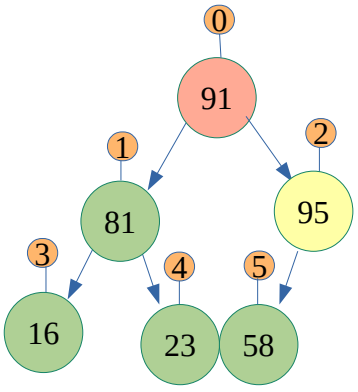
95 > 58



1ª iteração

0→n	91	81	58	16	23	95									
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

95 >91

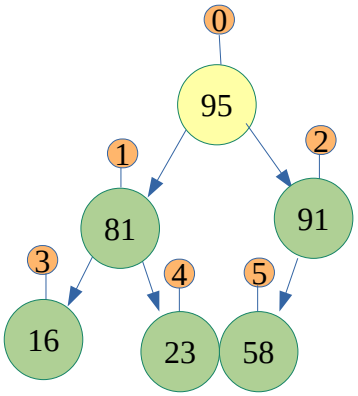


2ª iteração

0→n	91	81	95	16	23	58									
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

0→n	95	81	91	16	23	58									
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

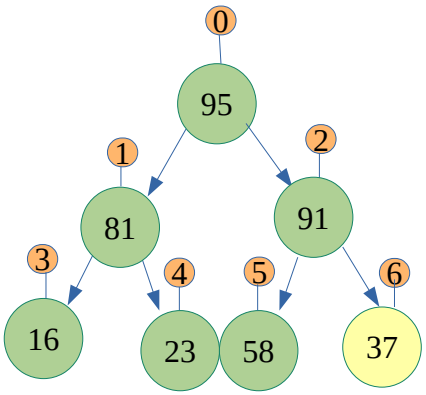
95 > 91



7º número: 37

0→n	95	81	91	16	23	58	37								
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

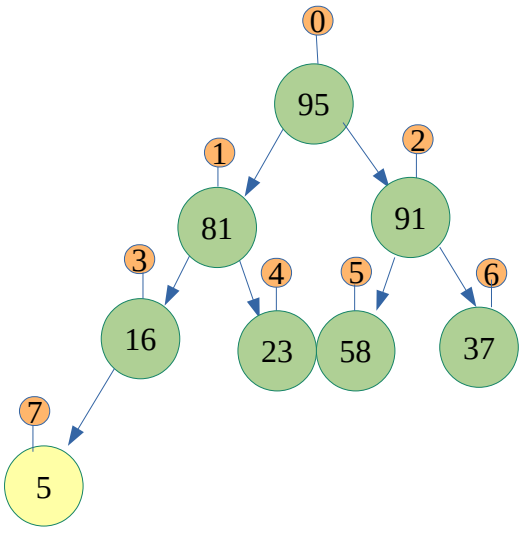
37 < 91



8º número: 5

0→n	95	81	91	16	23	58	37	5							
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

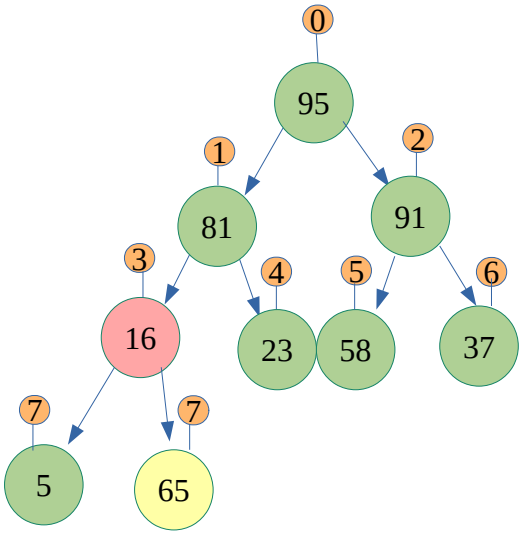
5 < 16



9º número: 65

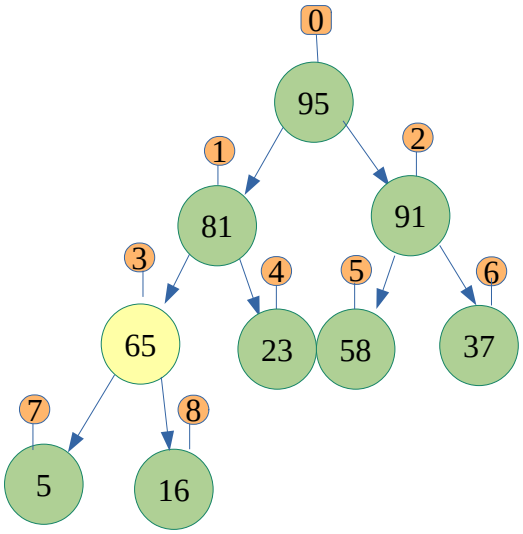
0→n	95	81	91	16	23	58	37	5	65						
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

65 > 16



1ª iteração

0→n	95	81	91	16	23	58	37	5	65						
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

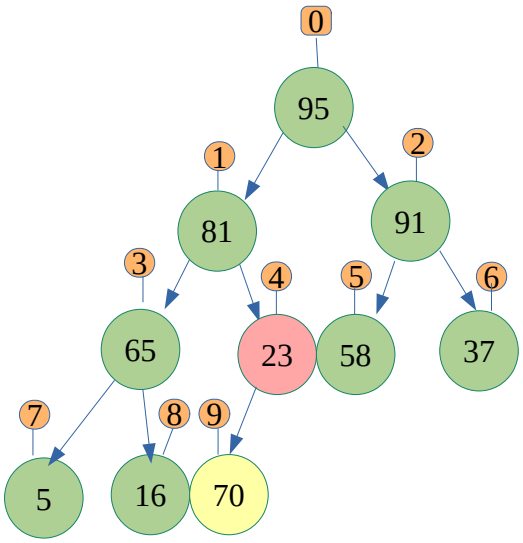


0→n	95	81	91	65	23	58	37	5	16						
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

10º número: 70

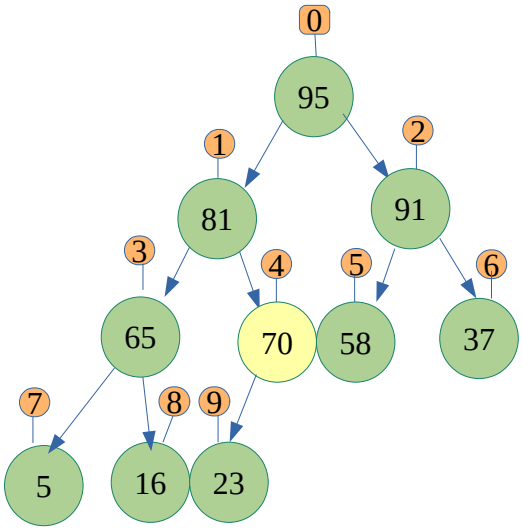
0→n	95	81	91	65	23	58	37	5	16	70					
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

70 > 23



1ª iteração

0→n	95	81	91	65	23	58	37	5	16	70					
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

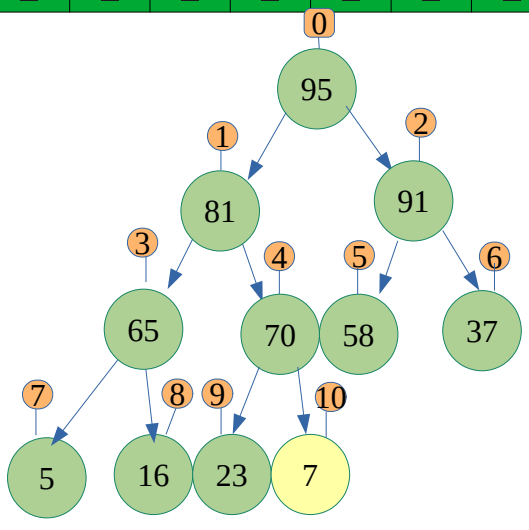


0→n	95	81	91	65	70	58	37	5	16	23					
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

11º número: 7

0→n	95	81	91	65	70	58	37	5	16	23	7				
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

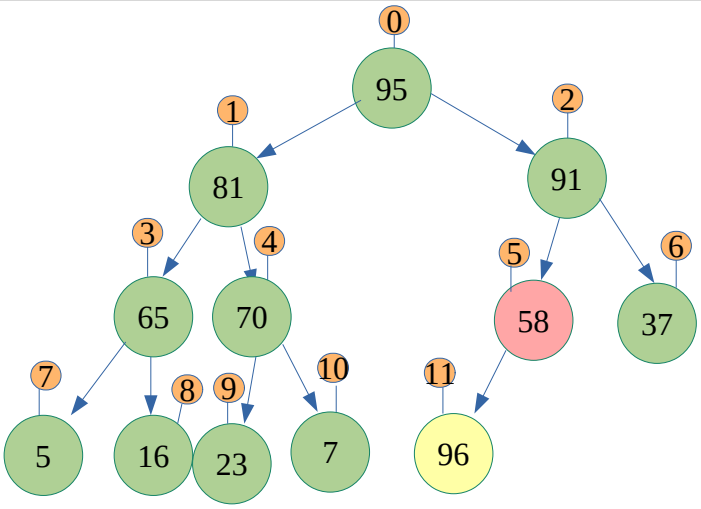
7 < 70



12º número: 96

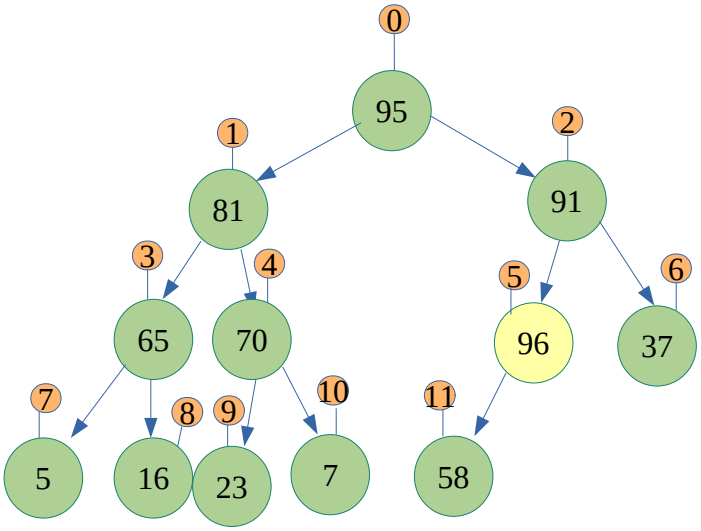
0→n	95	81	91	65	70	58	37	5	16	23	7	96			
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

96>58



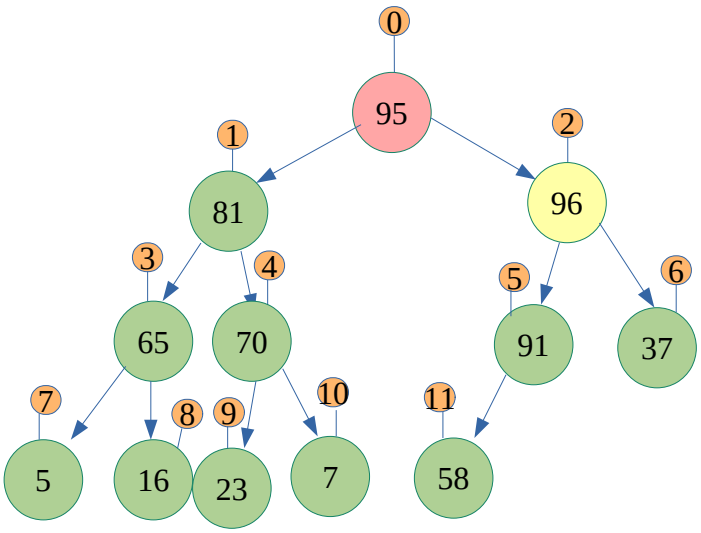
0→n	95	81	91	65	70	58	37	5	16	23	7	96			
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

96> 91

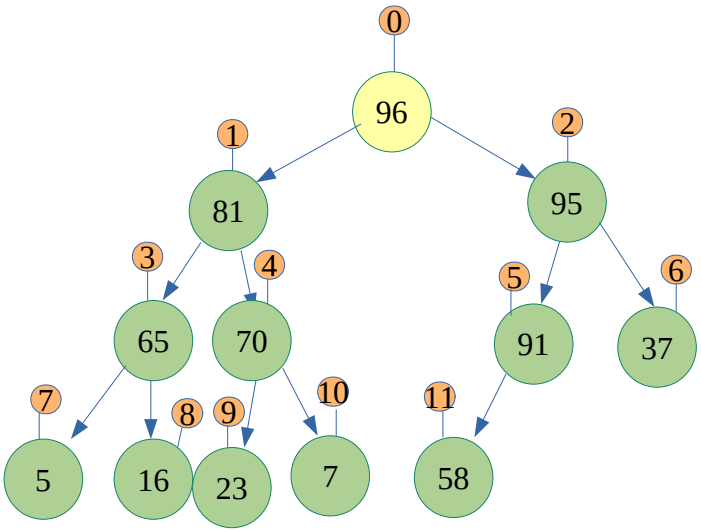


0→n	95	81	91	65	70	96	37	5	16	23	7	58			
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

96> 95



0→n	95	81	96	65	70	91	37	5	16	23	7	58			
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

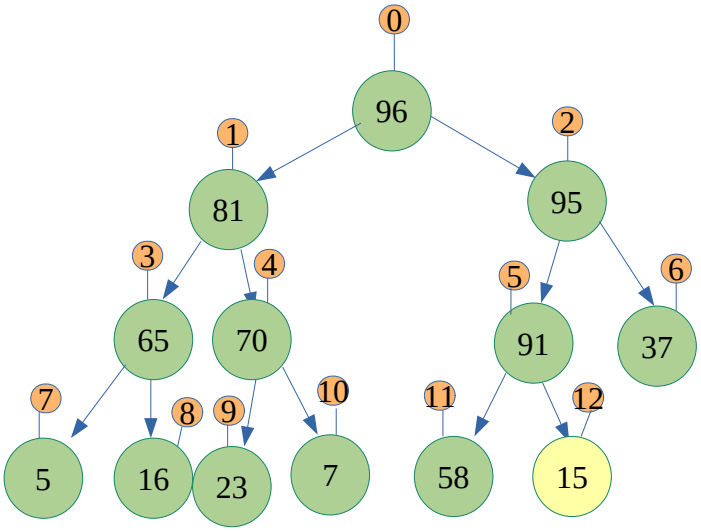


0→n	96	81	95	65	70	91	37	5	16	23	7	58			
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

13º número: 15

0→n	96	81	95	65	70	91	37	5	16	23	7	58	15		
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

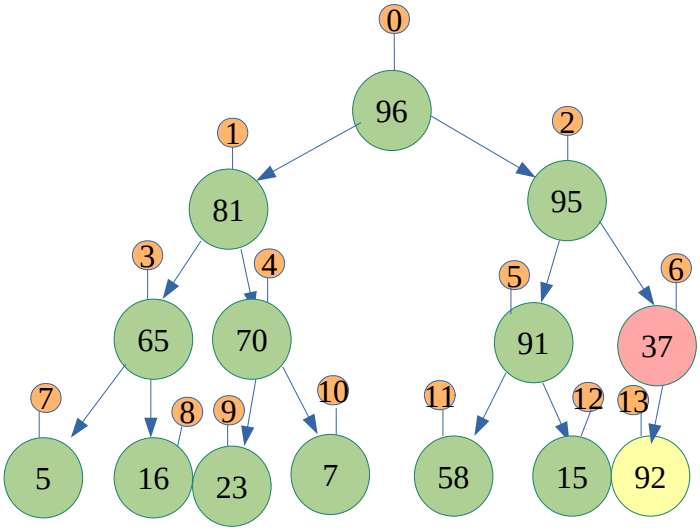
15<91



14º número: 92

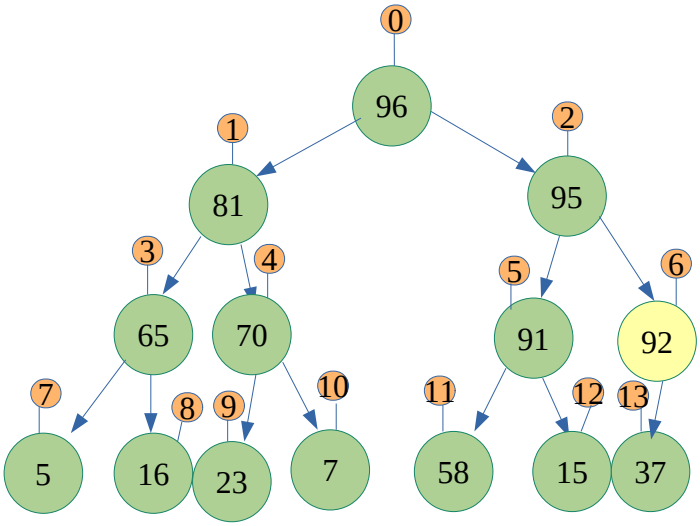
0→n	96	81	95	65	70	91	37	5	16	23	7	58	15	92	
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

92 > 37



1ª iteração

0→n	96	81	95	65	70	91	37	5	16	23	7	58	15	92	
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

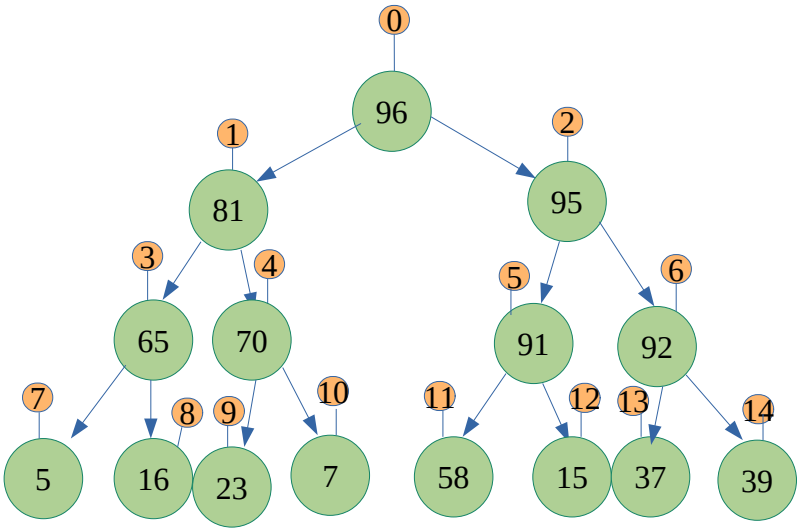


0→n	96	81	95	65	70	91	92	5	16	23	7	58	15	37	
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

15º número: 39

0→n	96	81	95	65	70	91	92	5	16	23	7	58	15	37	39
indx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

39<92



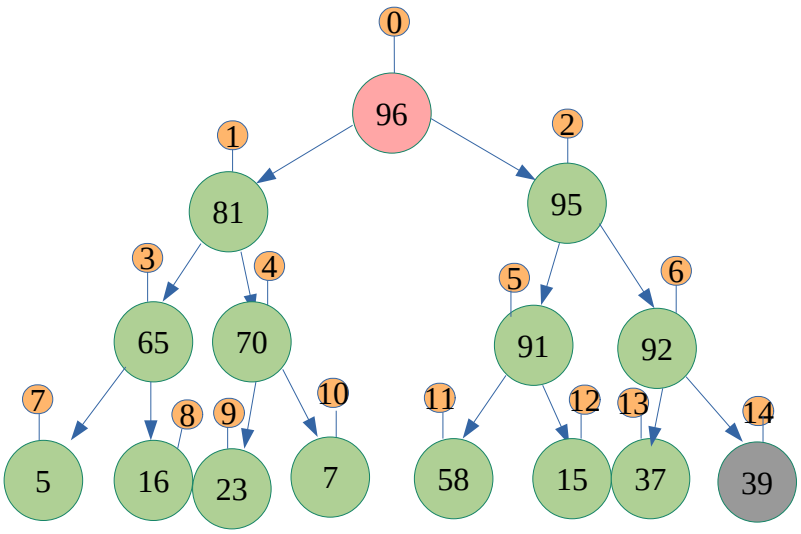
2 - Represente os elementos do heap como um vetor.

vetor final:

96	81	95	65	70	91	92	5	16	23	7	58	15	37	39

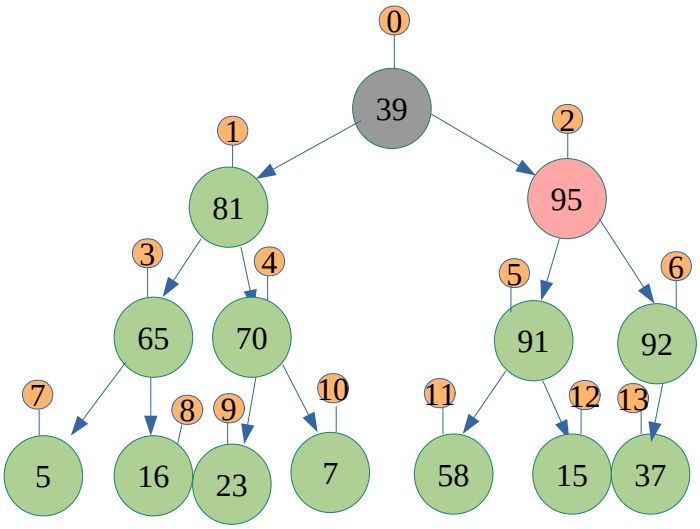
3 - Faça a remoção do elemento raiz e mostre como fica o heap após a remoção.

Deletar raiz(chave 96);



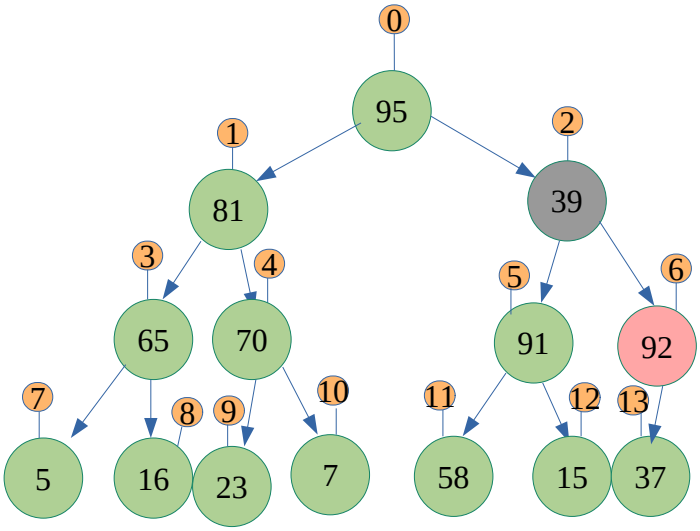
primeira iteração (troca raiz pelo último elemento)

95 > 39

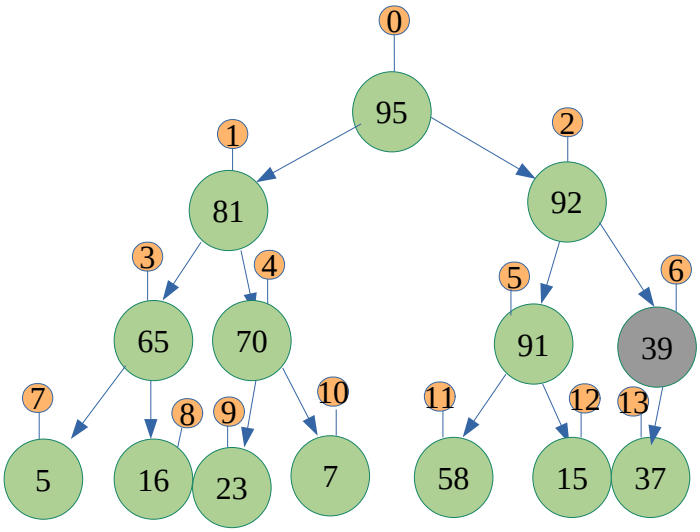


2ª iteração, balanceamento da raiz com a folha de maior índice na linha seguinte;

92 > 39

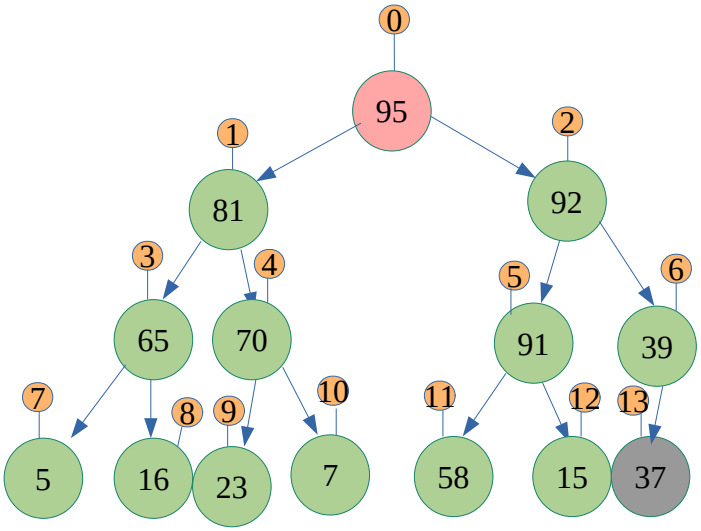


3ª iteração, se ainda necessário, continue fazendo o balanceamento nas demais linhas;



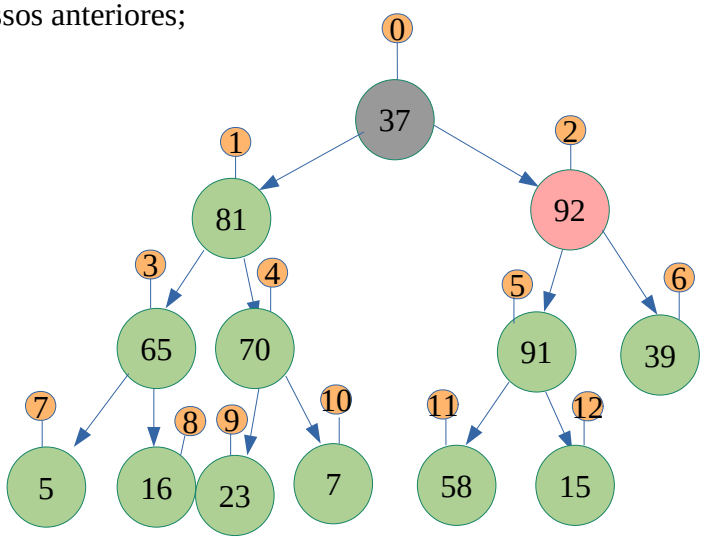
3.1 - Repita o item 3 mais 4 vezes (ou seja, serão feitas 5 remoções).

Deletar raiz(chave 95);

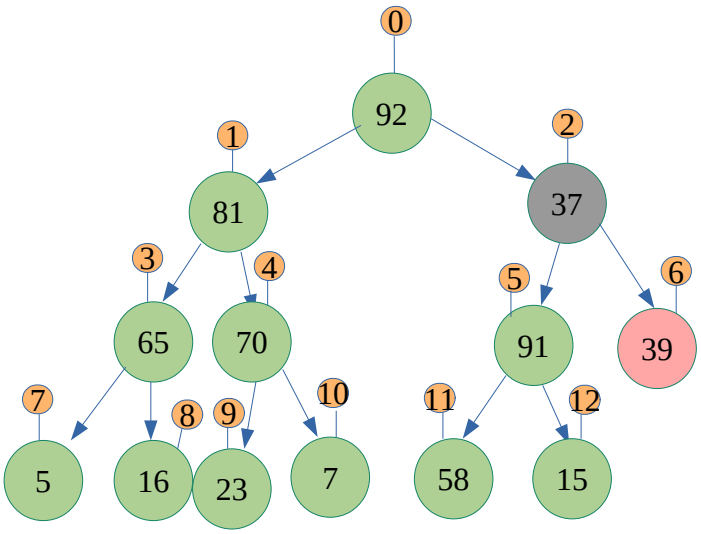


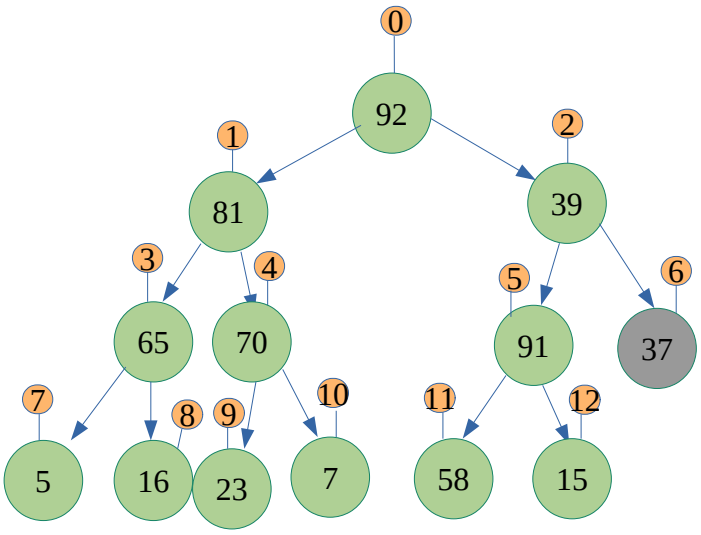
Repete os passos anteriores;

92 > 37

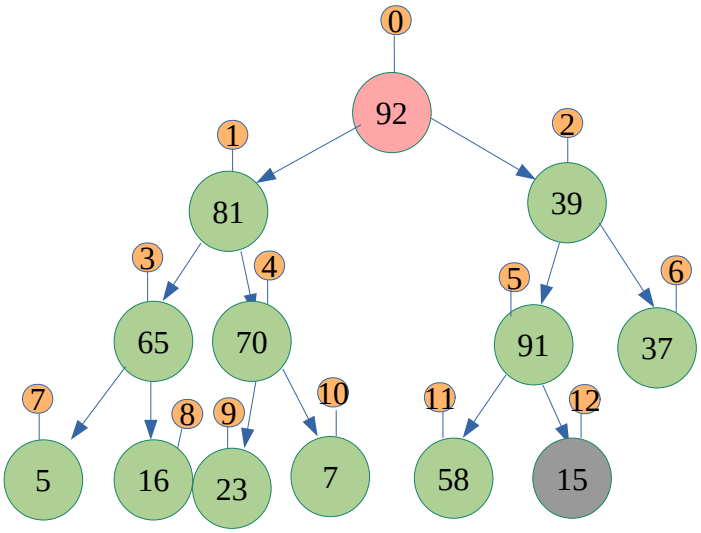


39 > 37

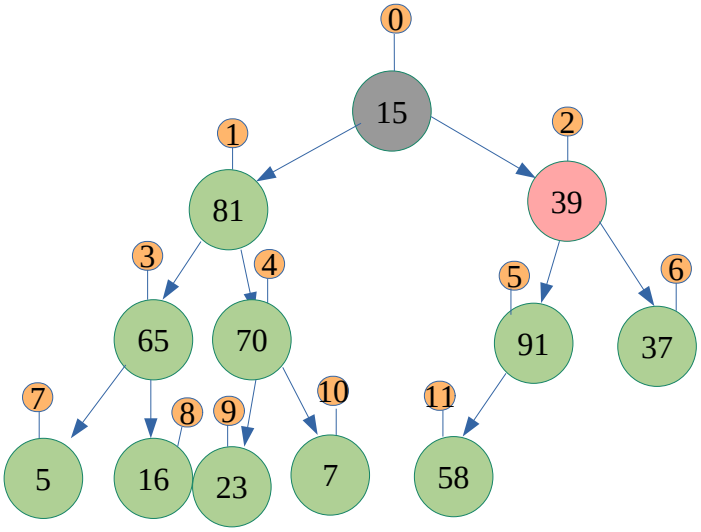




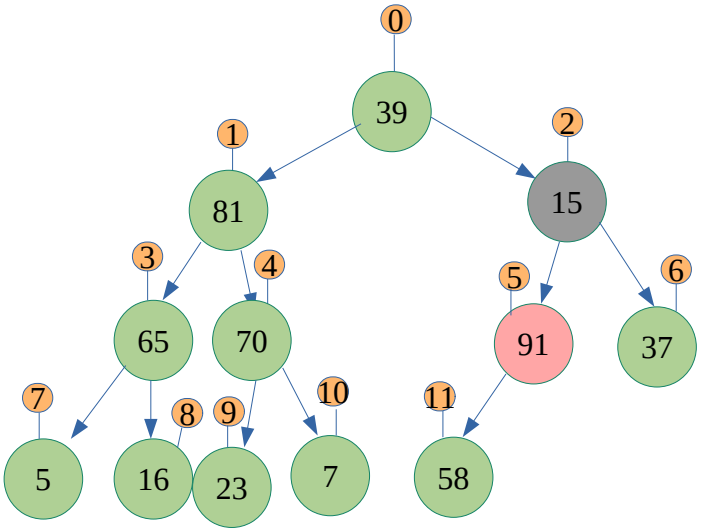
Deletar raiz (chave 92)



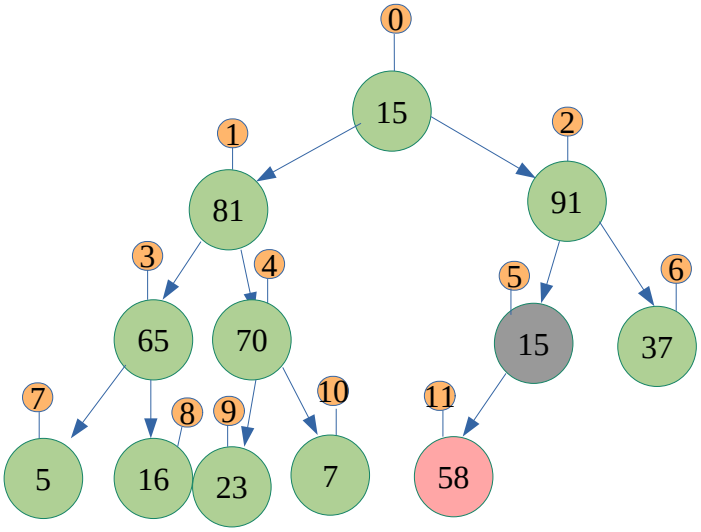
39 > 15

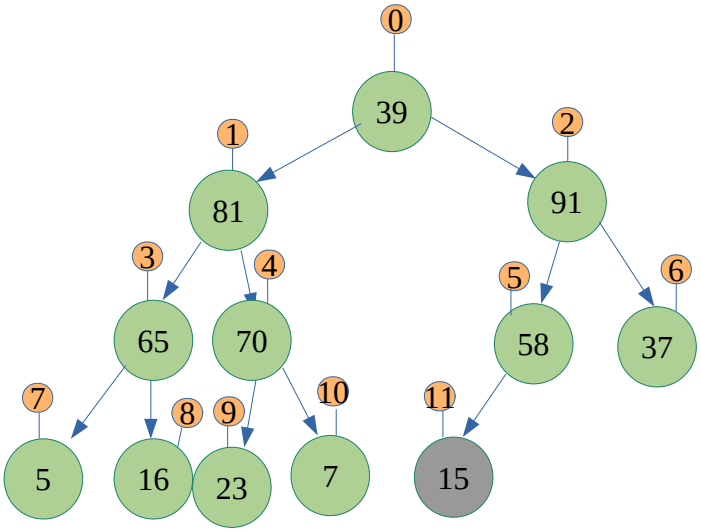


91 > 15

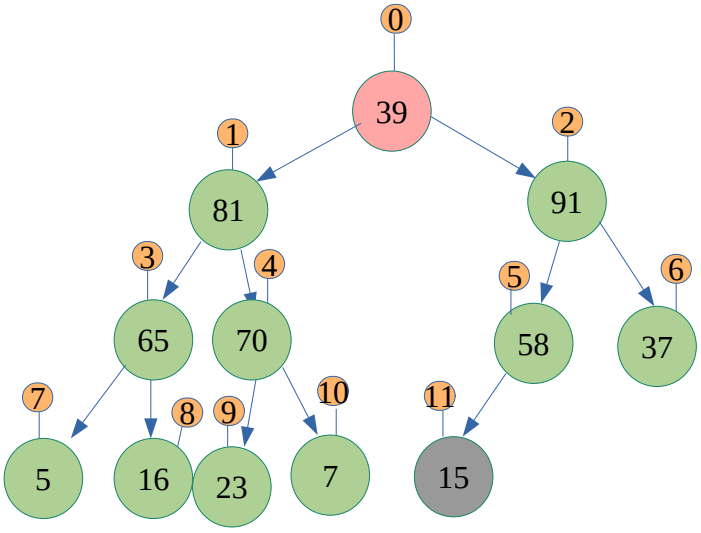


58 > 15

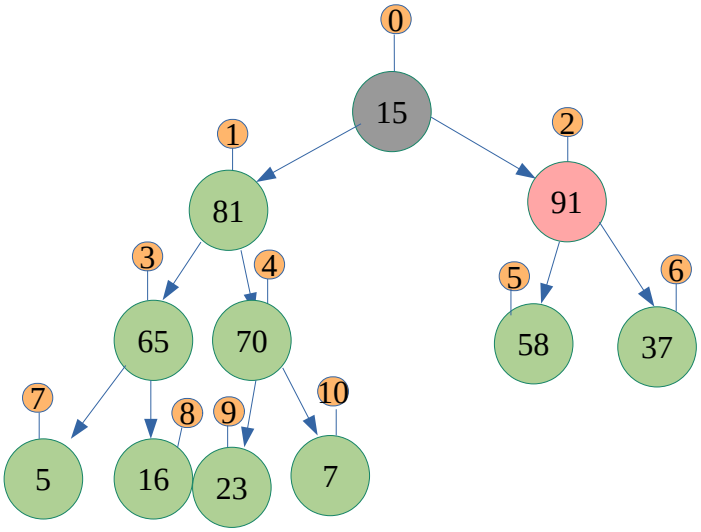




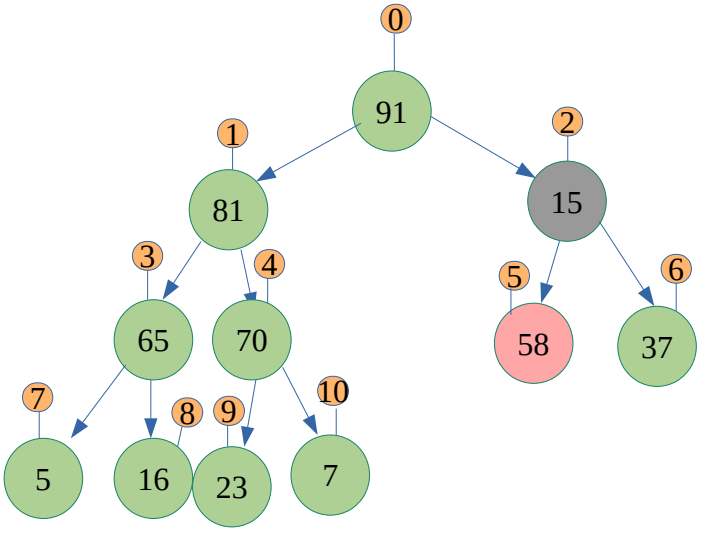
Deletar raiz (chave 39) substitui 39[index 0] ↔ 15[ultimo_index]


0
39
1
81
2
91
3
65
4
70
5
58
6
37
7
5
8
16
9
23
10
7
11
15

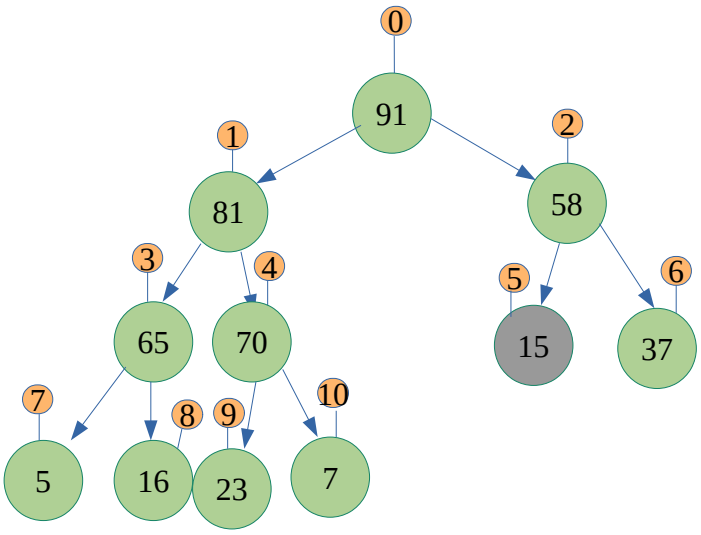
91 > 15


0
15
1
81
2
91
3
65
4
70
5
58
6
37
7
5
8
16
9
23
10
7

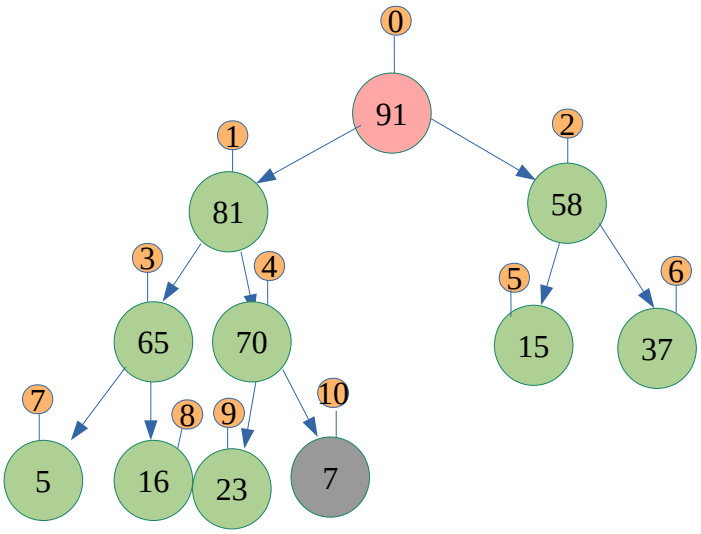
58 > 15


0
91
1
81
2
15
3
65
4
70
5
58
6
37
7
5
8
16
9
23
10
7

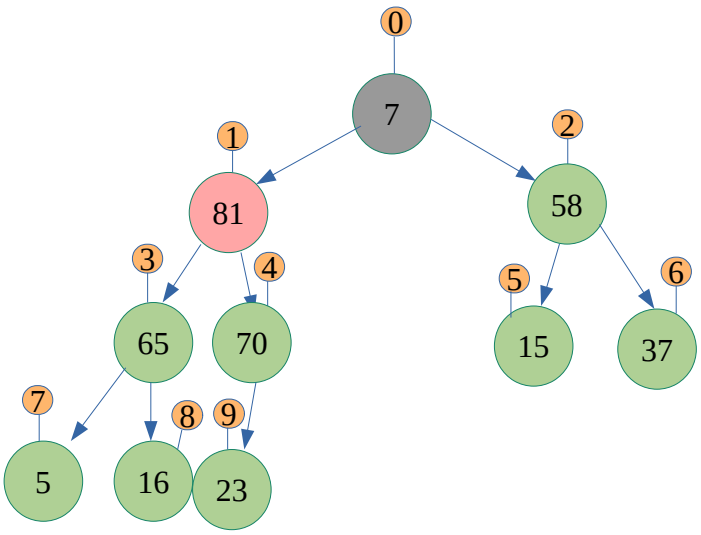
58 > 15



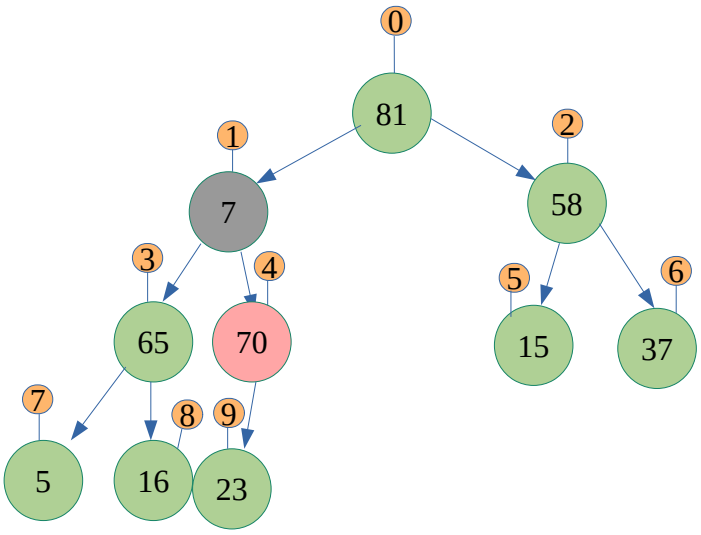
Deletar raiz (chave 91) substitui 91[index 0] ↔ 7[ultimo_index]



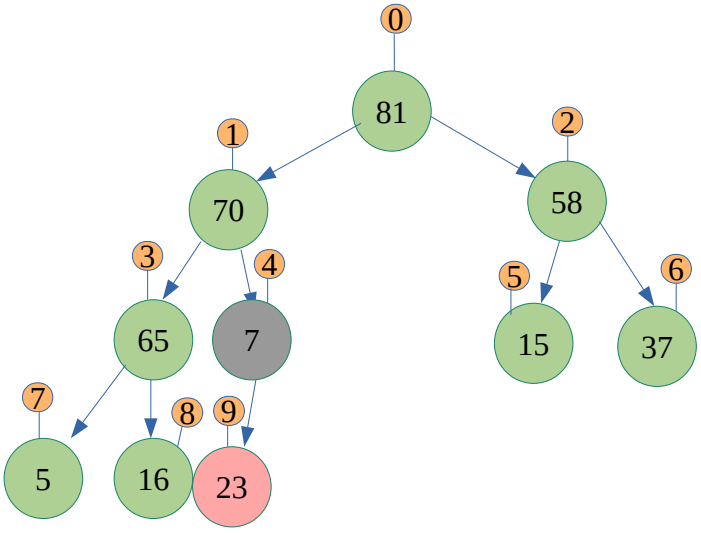
81 > 7

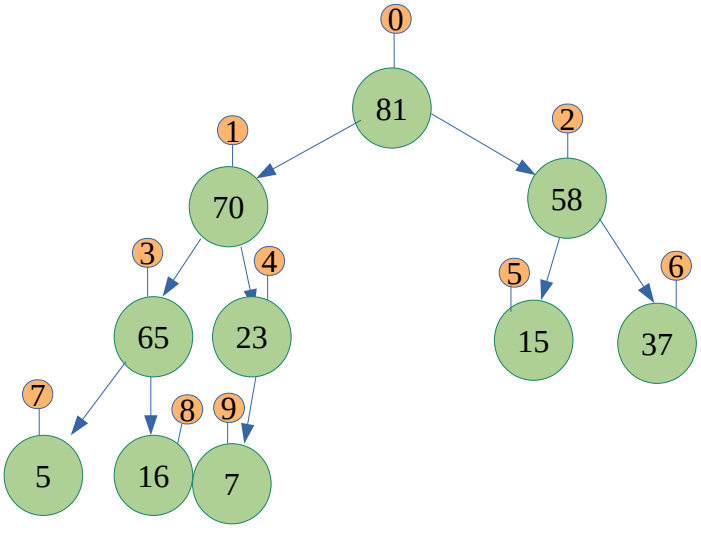


70 ≥ 7


0
81
1
7
2
58
3
65
4
70
5
15
6
37
7
5
8
16
9
23

23 ≥ 7


0
81
1
70
2
58
3
65
4
7
5
15
6
37
7
5
8
16
9
23


0
81
1
70
2
58
3
65
4
23
5
15
6
37
7
5
8
16
9
7

4 - Represente em um vetor como ficou o heap após as 5 remoções.

vetor final

81	70	58	65	23	15	37	5	16	7

Observações finais:

Foi analisado que existe um custo maior na remoção de uma raiz comparado a inserção ordenada. Visto que em todos os casos é necessario fazer um rebalanciamento na árvore, isso somente se executarmos uma recursão para deletar toda árvore à partir da raiz comparado a inserção de todos os elementos ordenados.