

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <locale.h>
4 #define TAMANHO_PILHA 20
5
6 typedef struct{
7     int vetor[TAMANHO_PILHA]; //VETOR ♦ A PILHA com tamanho = a tamanho da pilha;
8     int topo;
9 }Pilha;
10
11 //prototipo da função empilha
12 void empilha(int valor, Pilha * pilha){
13     //pilha->topo significa: ponteiro "pilha" apontando para o CONTEUDO de um item
14     //de uma struct é equivalente a (*pilha).topo
15     if(pilha->topo < TAMANHO_PILHA){ //verificando se pilha não esta cheia
16         //dai pode empilhar
17         pilha->vetor[pilha->topo] = valor;
18         pilha->topo++;
19     }else{
20         printf("não ha mais espaço na pilha, \n");
21     }
22 }
23
24 void desempilha(int valor, Pilha *pilha){
25     if(pilha->topo > 0){
26         pilha->topo--; //desempilha
27         printf("Elemento retirado: %d.\n", pilha->vetor[pilha->topo]);
28     }else{
29         printf("A pilha esta vazia.\n"); //pilha vazia
30     }
31 }
32
33 int isCheia(Pilha *pilha){
34     if(pilha->topo >= TAMANHO_PILHA){
35         return 1; //identifica se a pilha esta cheia
36     }else{
37         return 0;
38     }
39 }
40
41 int isVazia(Pilha *pilha){
42     if(pilha->topo == 0){
43         return 1; //identifica se a pilha esta cheia
44     }else{
45         return 0;
46     }
47 }
48
49 void imprimePilha(Pilha *pilha){
50     int i;
51     for(i=(pilha->topo);i-->0){ //valor inicial de i na ultima posição da pilha e
52         //dai decrementa
53         printf("%02d\n", pilha->vetor[i]);
54     }
55 }
56
57 int main()
58 {
59     setlocale(LC_ALL, "Portuguese");
60     //DECLARA UMA PILHA
61     Pilha p;
62     p.topo = 0; //o topo da pilha deve começar em zero
63     char oper;
64     int num;
65
66     while (oper != 'x')
67     {
68         printf("\n\n-----MENU DA PILHA-----\n\n");
69         printf("Selecione a operacao desejada: \n\n");
70         printf("a: Adicione um elemento\n");
71         printf("r: Remover um elemento \n");
72         printf("s: Exibir os elementos \n");
73         printf("t: Exibir o numero de elementos \n");
74         printf("x: Encerrar programa\n");
75         scanf(" %c", &oper);
76         switch (oper)
77         {
78             {
79                 case 'a':

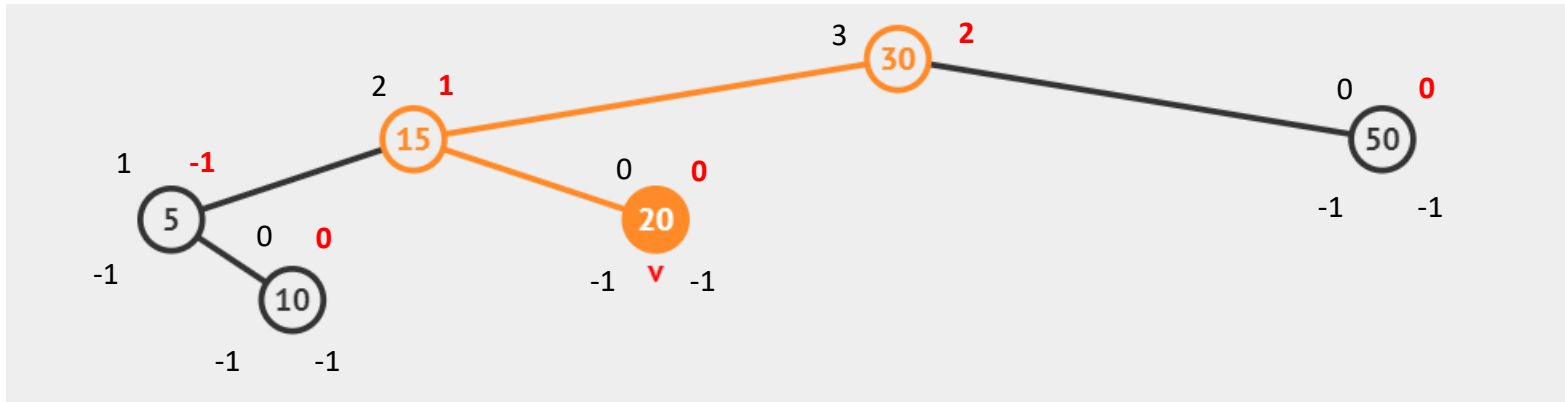
```

```
76         printf("\nDigite o valor que deseja inserir na pilha: \n");
77         scanf("%d",&num);
78         empilha(num,&p);
79         break;
80
81     case 'r':
82         desempilha(num,&p);
83         break;
84
85     case 's':
86         imprimePilha(&p);
87         break;
88
89     case 't':
90         printf("Topo da pilha: %d. \n", p.topo);
91         break;
92
93     default:
94         printf("Você digitou uma operacao invalida.\n\n");
95
96     }
97 }
98
99 return 0;
100 }
101
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define TAMANHO_FILA 20
4
5 typedef struct
6 {
7     int vetor[TAMANHO_FILA];
8     int fim;
9 } Fila;
10
11 void inserir(int valor, Fila *fila)
12 {
13     if (fila->fim < TAMANHO_FILA)
14     {
15         fila->vetor[fila->fim] = valor;
16         fila->fim++;
17     }
18     else
19     {
20         printf("Nao ha mais espaco na fila, \n");
21     }
22 }
23
24 void remover(Fila *fila)
25 {
26     if (fila->fim > 0)
27     {
28         printf("Elemento que sai da Fila: %d.\n", fila->vetor[0]);
29         for (int i = 0; i < (fila->fim - 1); i++)
30         {
31             fila->vetor[i] = fila->vetor[i + 1];
32         }
33         fila->fim--;
34     }
35     else
36     {
37         printf("A fila esta vazia. \n");
38     }
39 }
40
41 int isCheia(Fila *fila)
42 {
43     if (fila->fim >= TAMANHO_FILA)
44     {
45         return 1;
46     }
47     else
48     {
49         return 0;
50     }
51 }
52
53 int isVazia(Fila *fila)
54 {
55     if (fila->fim == 0)
56     {
57         return 1;
58     }
59     else
60     {
61         return 0;
62     }
63 }
64
65 void imprimeFila(Fila *fila)
66 {
67     int i;
68     for (i = 0; i < (fila->fim); i++)
69     {
70         printf("%02d \n", fila->vetor[i]);
71     }
72 }
73
74 int main(int argc, char *argv[])
75 {
76     char oper = 'a';
77     int num;
78     // DECLARA UMA FILA
```

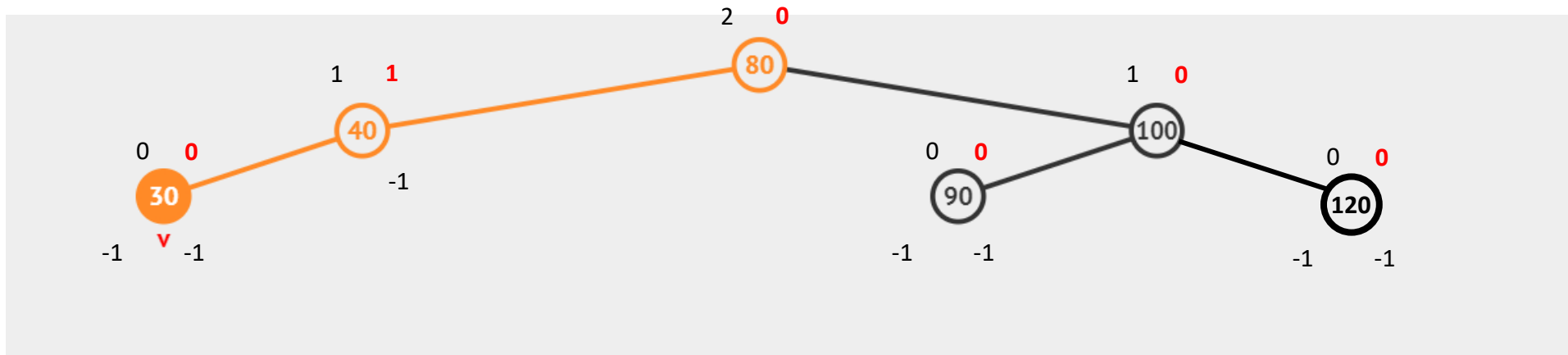
```
79     Fila f;  
80     f.fim = 0;  
81  
82     while (oper != 'x')  
83     {  
84         printf("\n\n-----MENU DA FILA-----\n");  
85         printf("Selecione a operacao desejada: \n\n");  
86         printf("a: Adicione um elemento\n");  
87         printf("r: Remover um elemento \n");  
88         printf("s: Exibir os elementos \n");  
89         printf("t: Exibir o numero de elementos \n");  
90         printf("x: Encerrar programa\n");  
91         scanf(" %c", &oper);  
92  
93         switch (oper)  
94         {  
95             case 'a':  
96                 printf("Digite o numero que você quer inserir: ");  
97                 scanf("%d", &num);  
98                 inserir(num, &f);  
99                 break;  
100  
101             case 'r':  
102                 remover(&f);  
103                 break;  
104  
105             case 's':  
106                 imprimeFila(&f);  
107                 break;  
108  
109             case 't':  
110                 printf("A fila tem %d elementos.\n", f.fim);  
111                 break;  
112             default:  
113                 printf("Você digitou uma operacao invalida.\n\n");  
114         }  
115     }  
116  
117     return 0;  
118 }
```

a)[30,15,50,5,10,20]



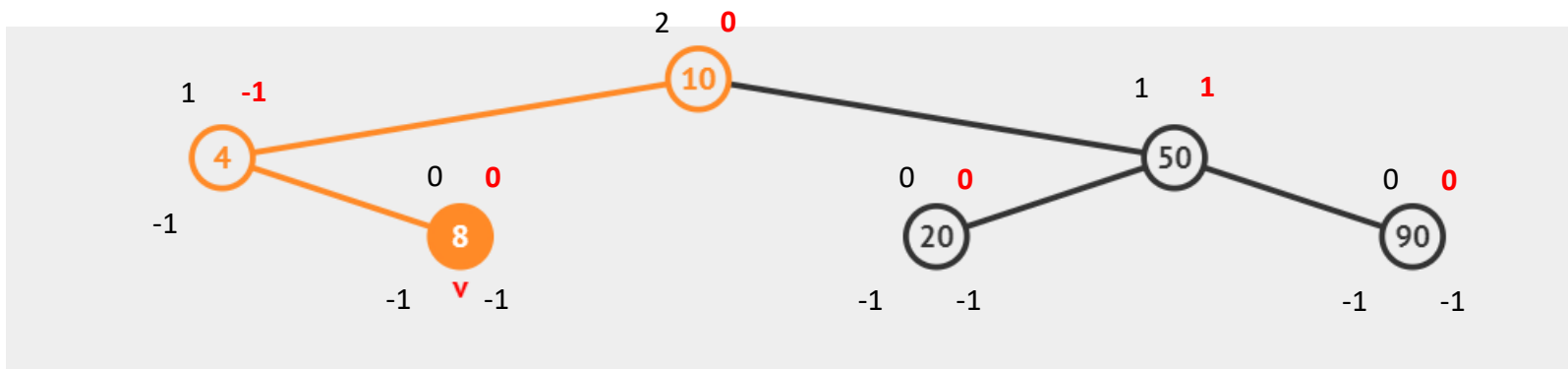
Não é AVL, pois esta desbalanceada no nó raiz (30).

b)[80,40,100,120,90,30]



É AVL, pois esta balanceada

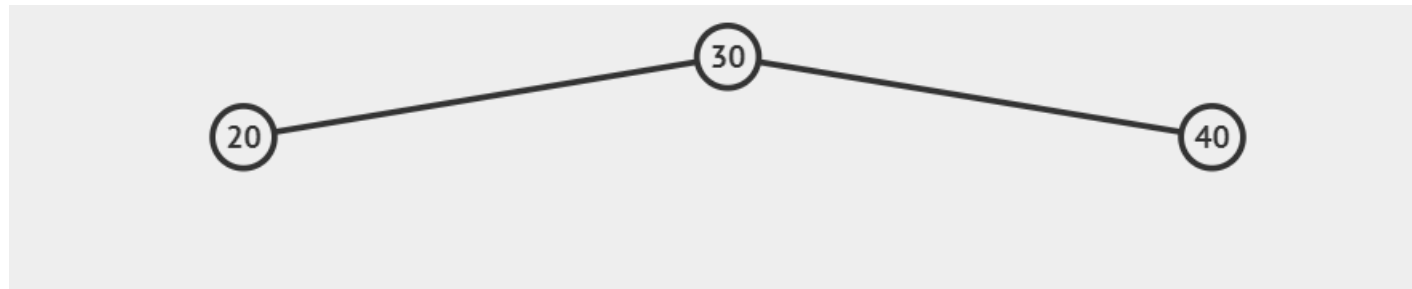
c)[10,50,4,90,20,8]



É AVL, pois esta balanceada

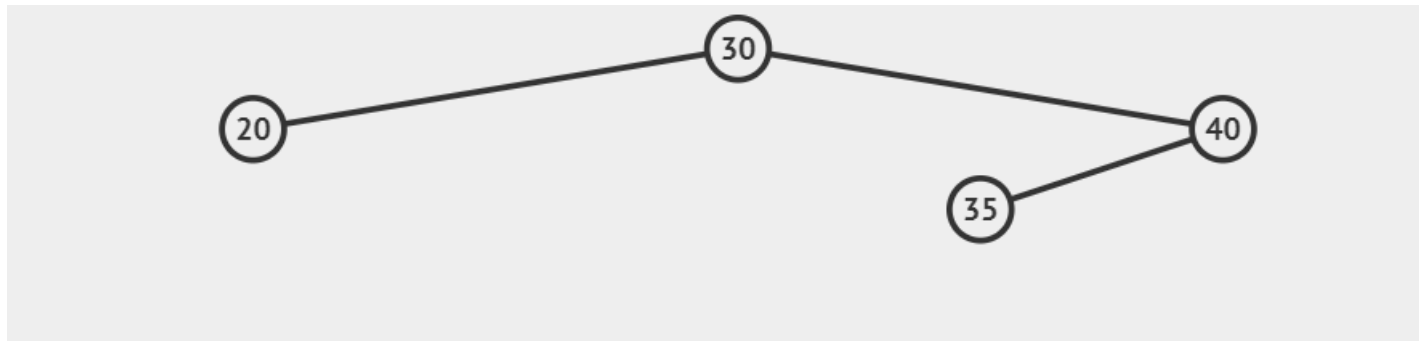
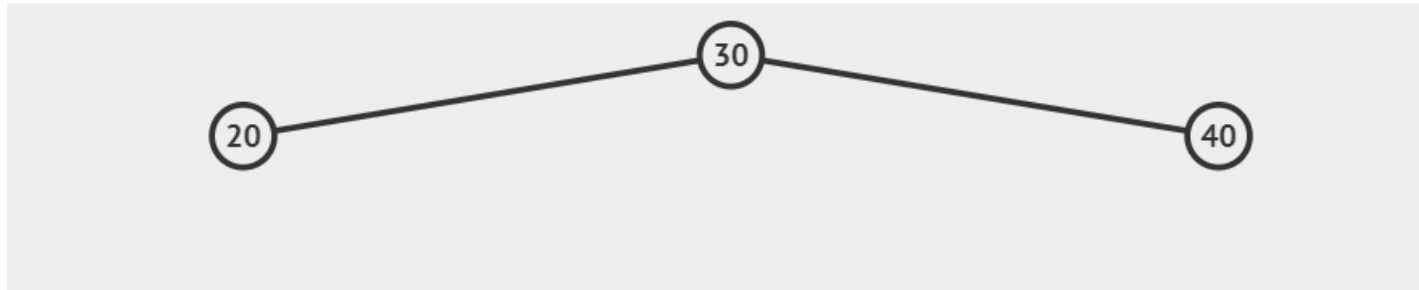
Rotação simples à Direita

a)[40,30,20]

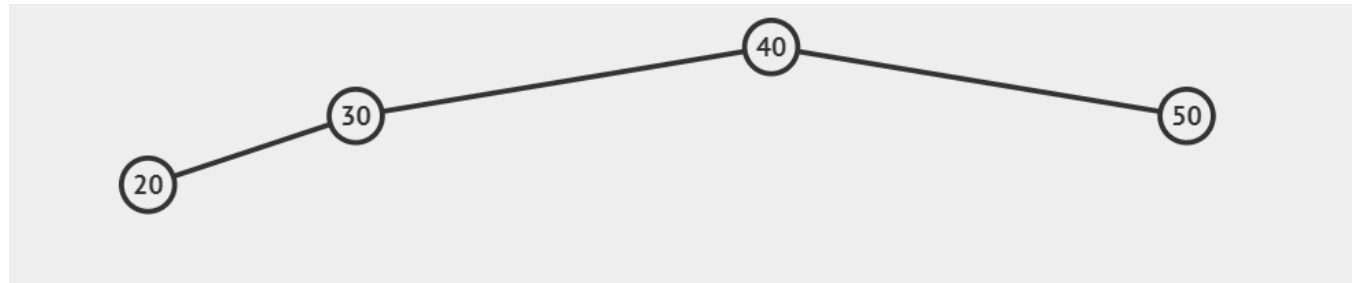
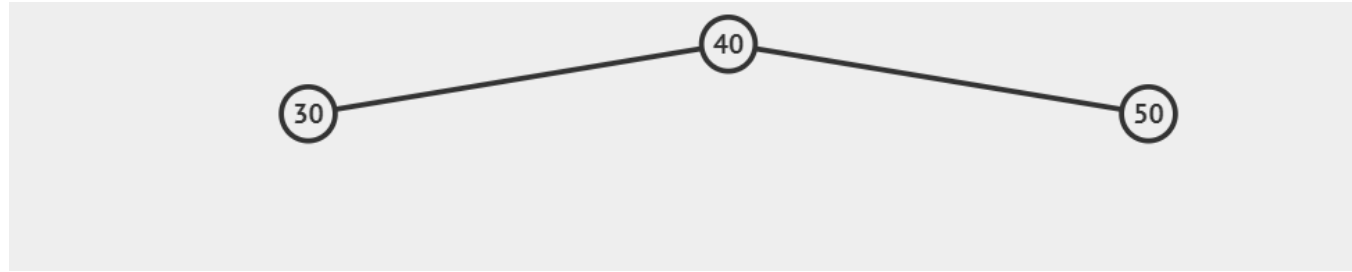


Rotação simples à Direita

b)[40,30,20,35]

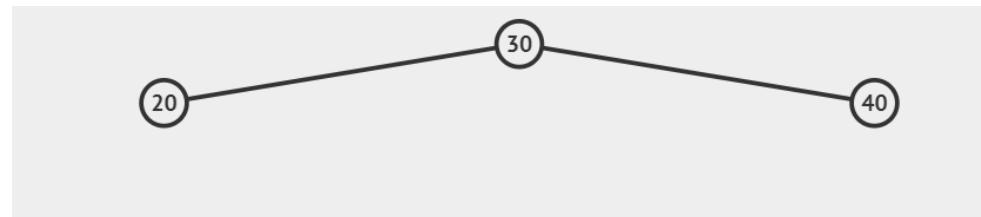
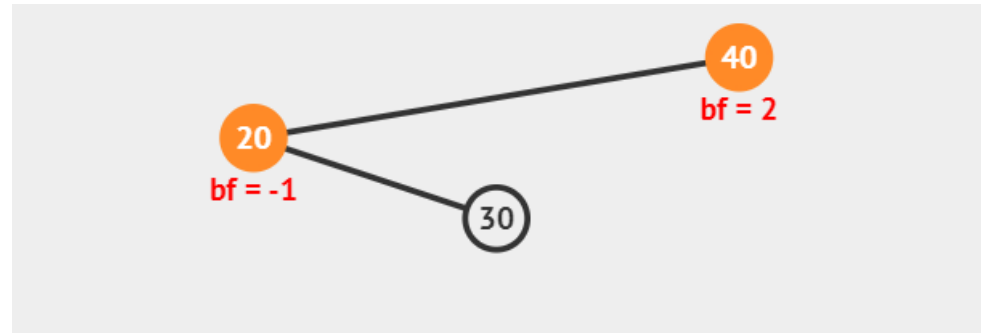


c)[40,50,30,20,35]



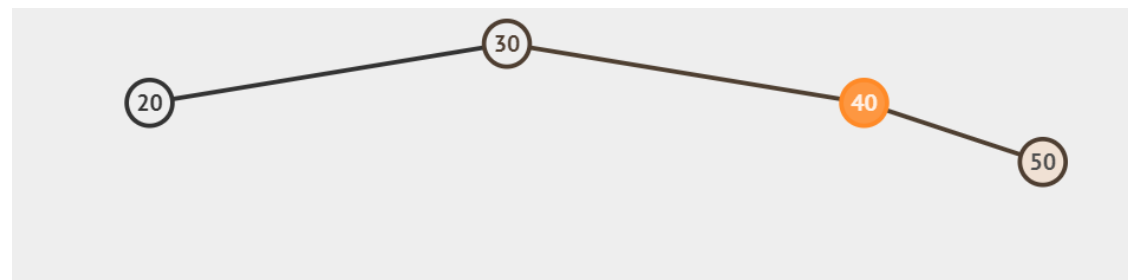
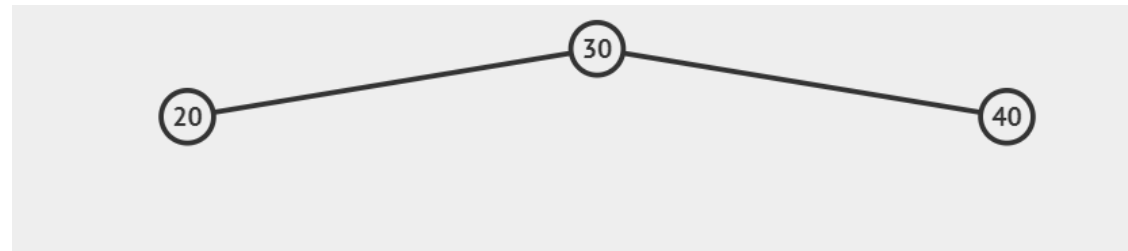
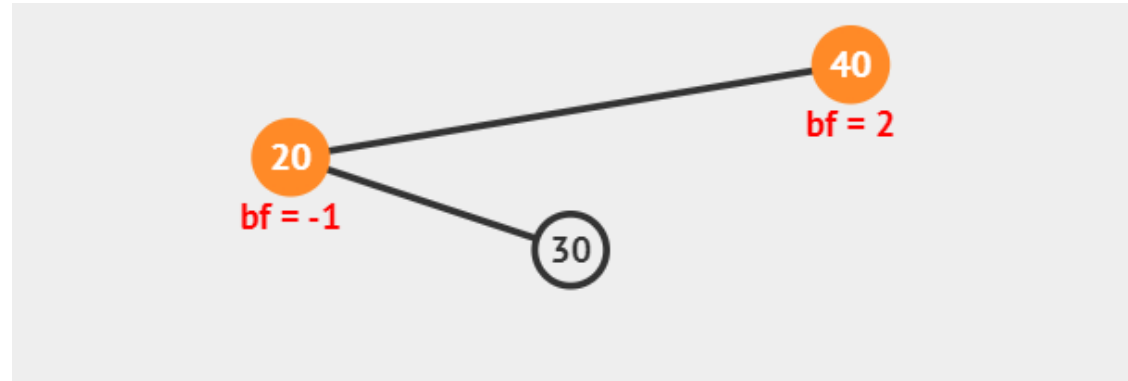
Rotação dupla a direita

[40,20,30]



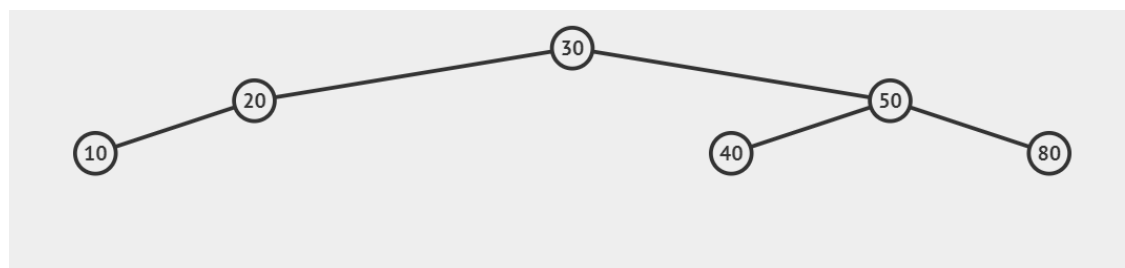
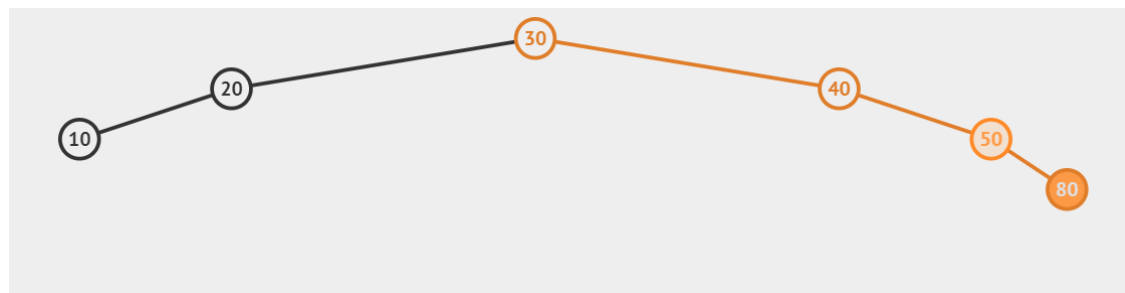
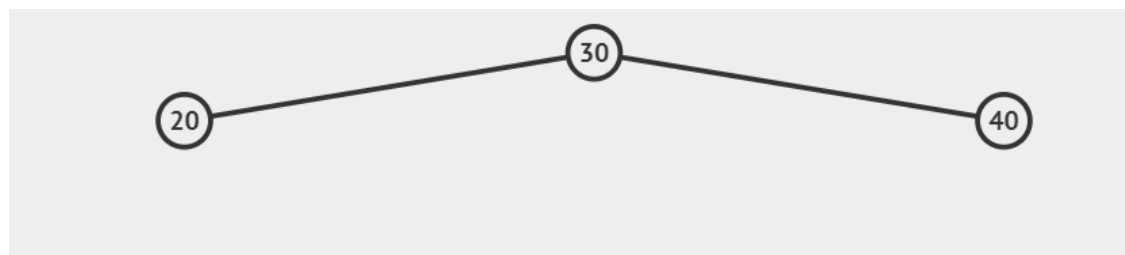
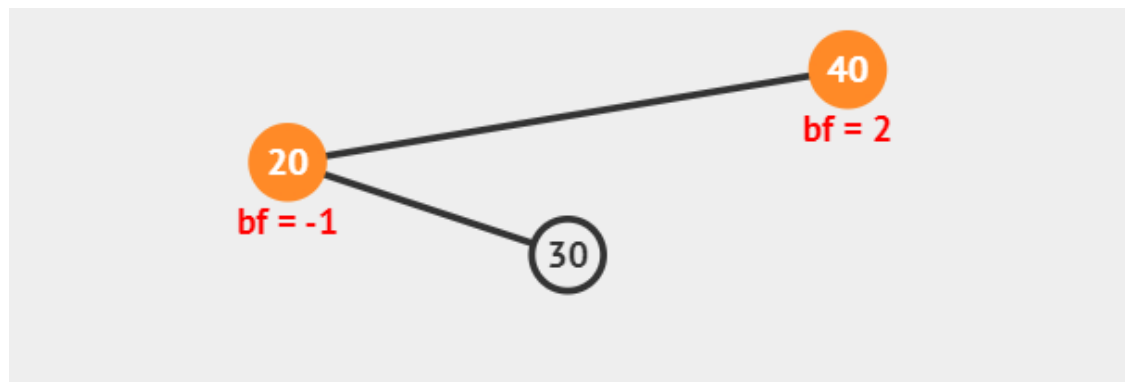
Rotação dupla a direita

b)[40,20,30,50]



Rotação dupla a direita

c)[40,20,30,10,50,80]



a)[30,15,50,5,10,20, Exclua o 15, 18]

Adicionando valores 30,15,50



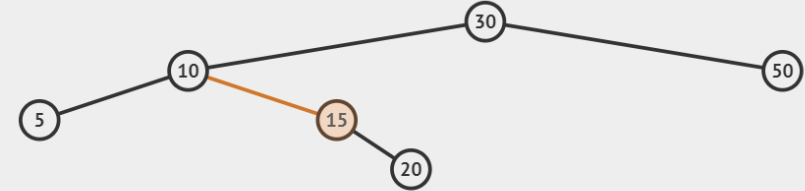
Adicionando valor 5



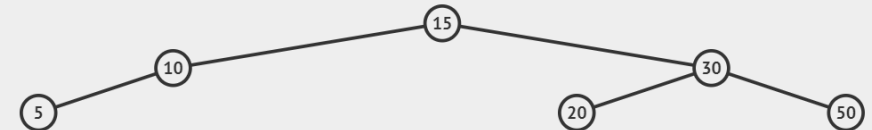
Adicionando valor 15



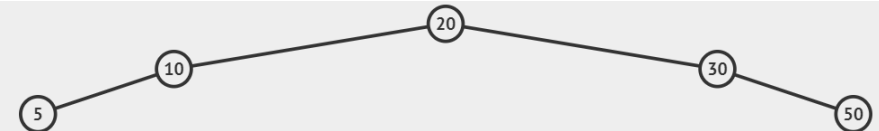
Adicionando valor 20



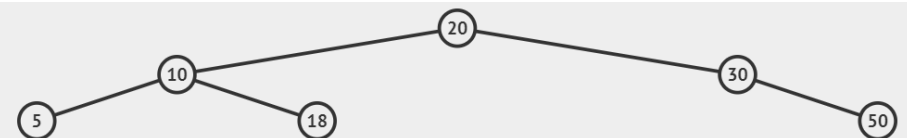
Rotação dupla a direita



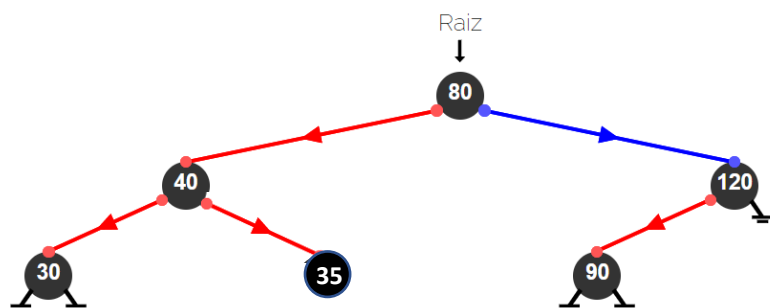
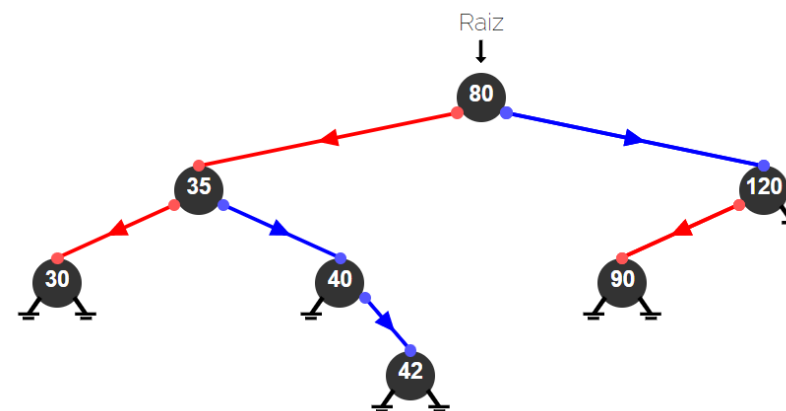
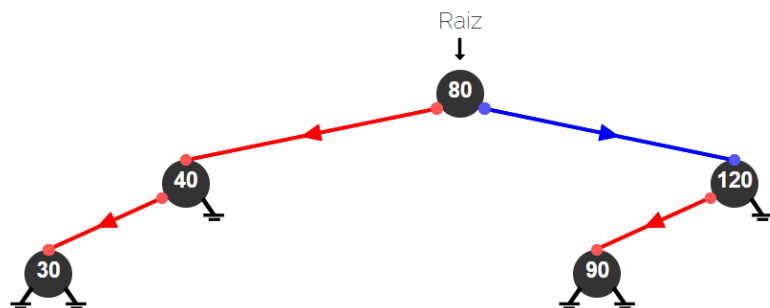
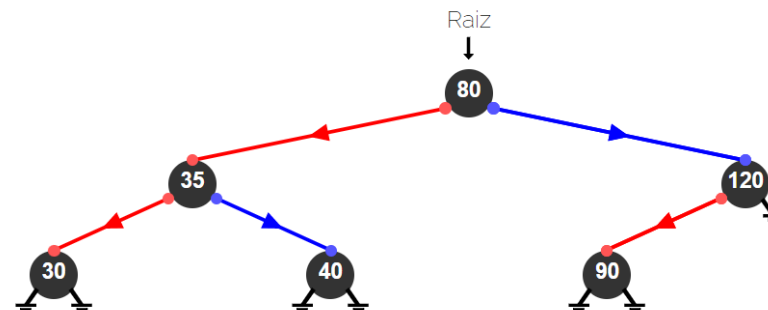
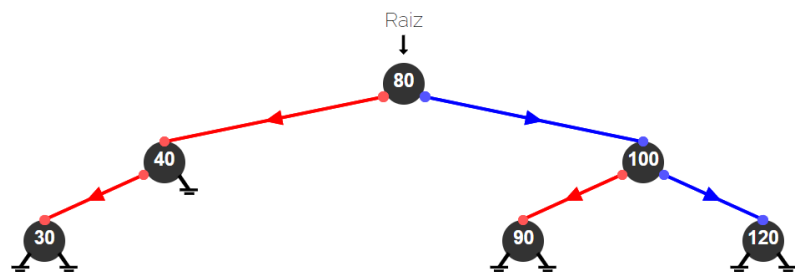
Remoção de nó com dois filhos



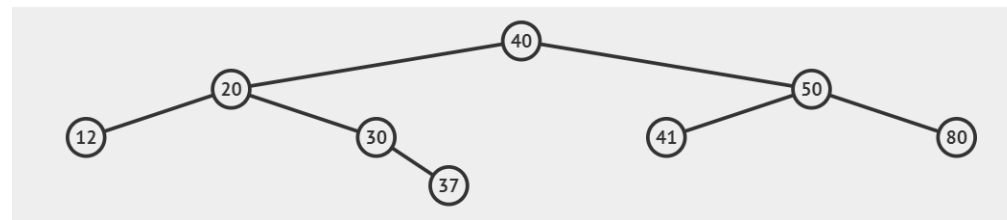
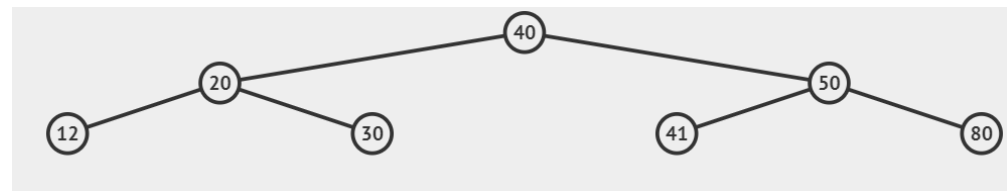
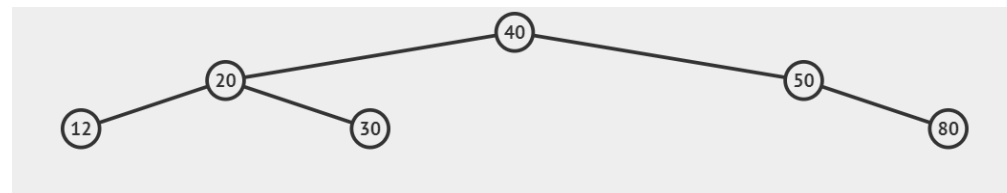
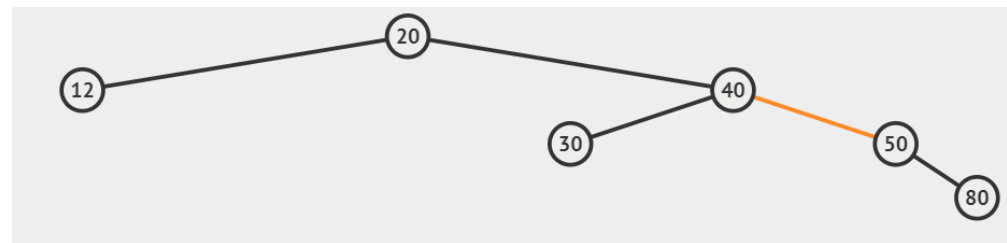
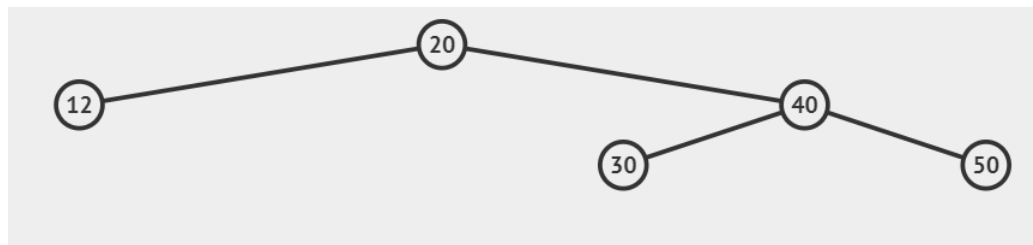
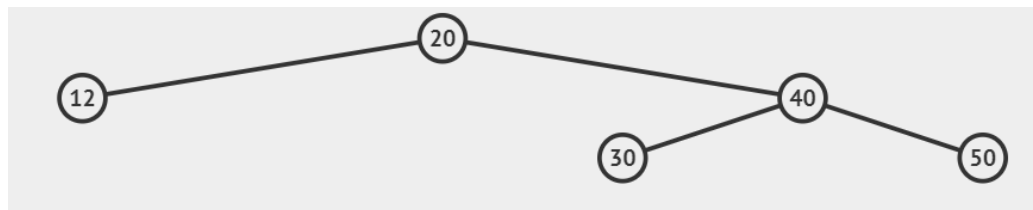
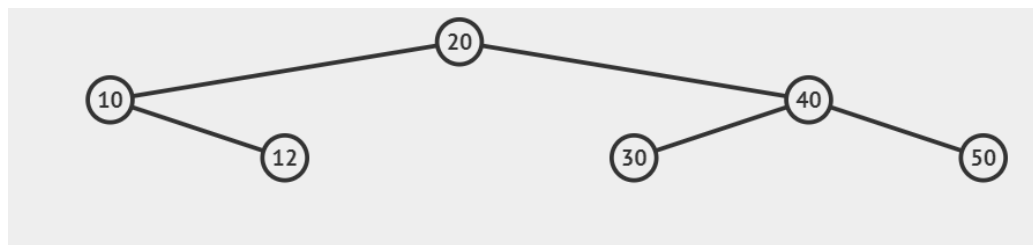
Adicionando valor 18



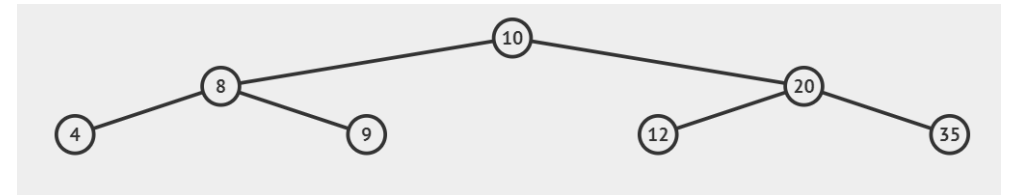
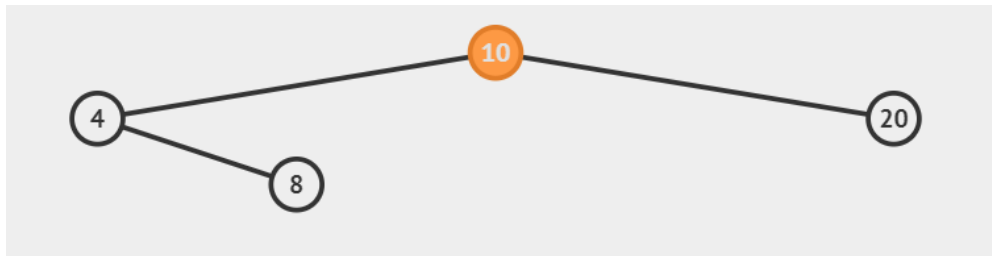
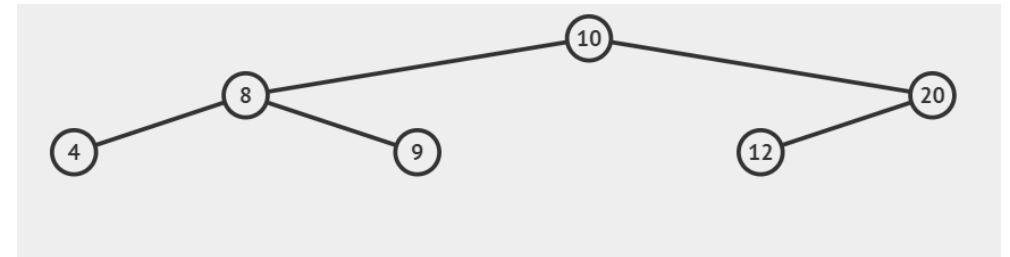
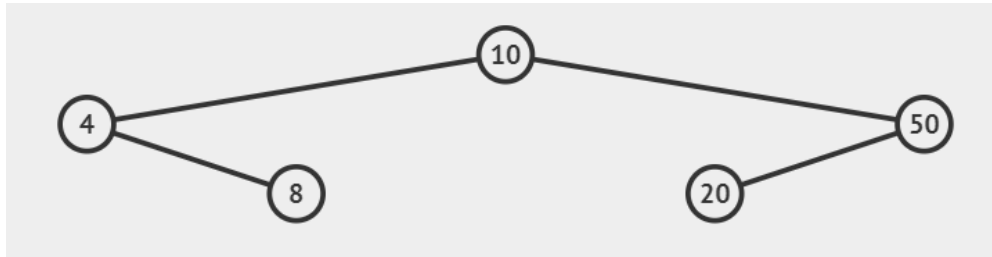
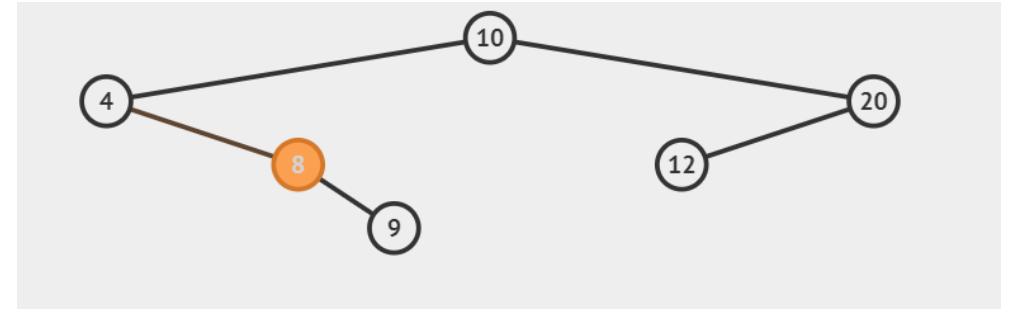
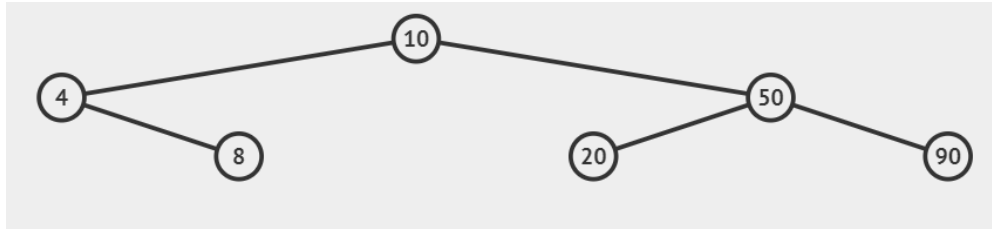
b)[80,40,100,120,90,30, Exclua o 100, 35, 42]



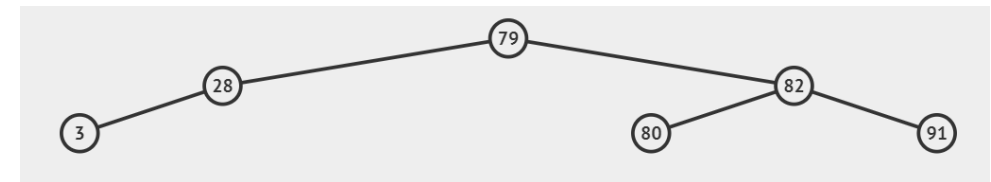
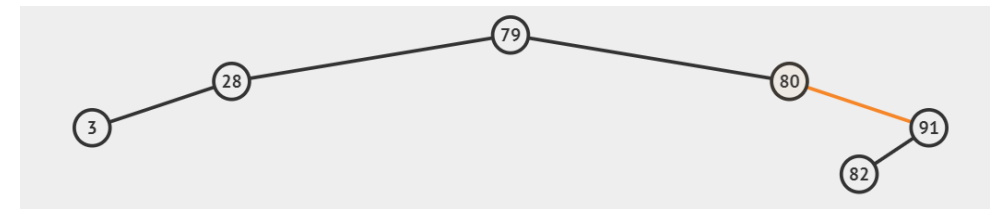
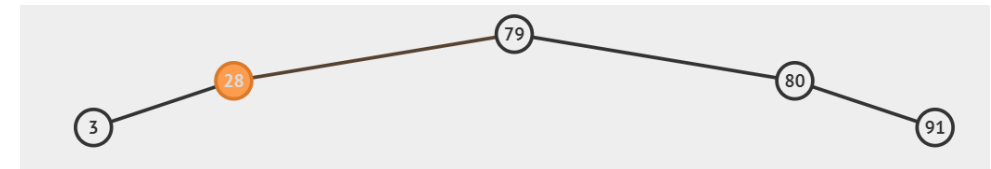
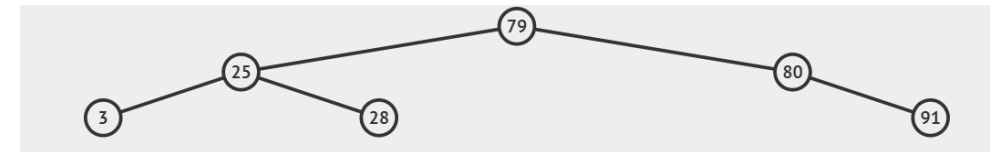
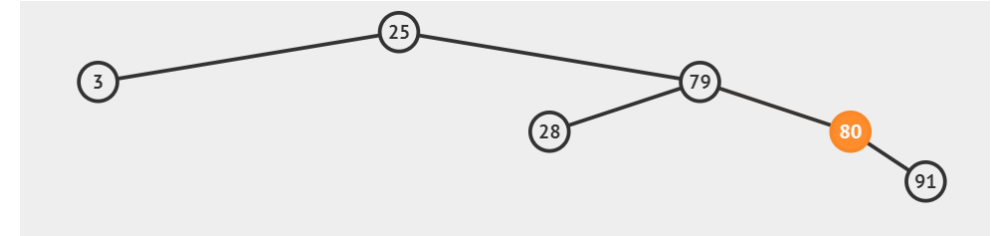
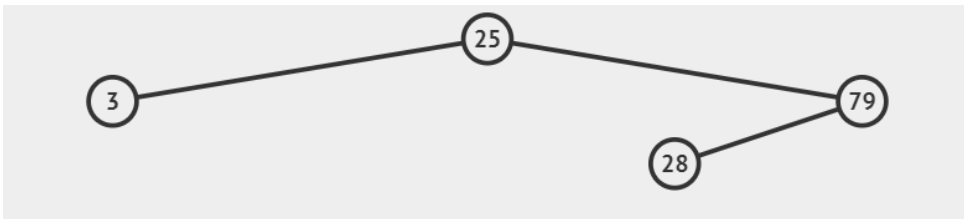
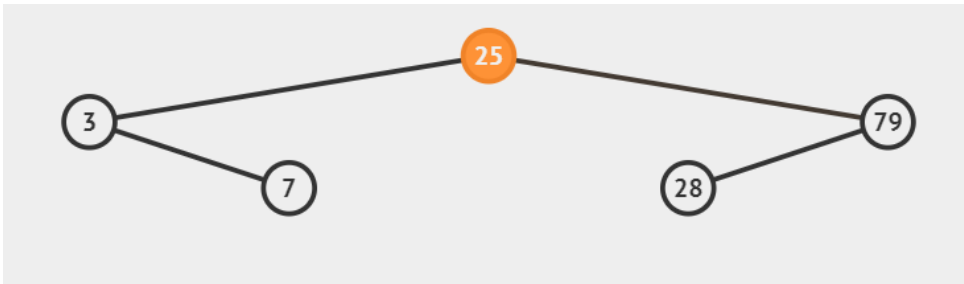
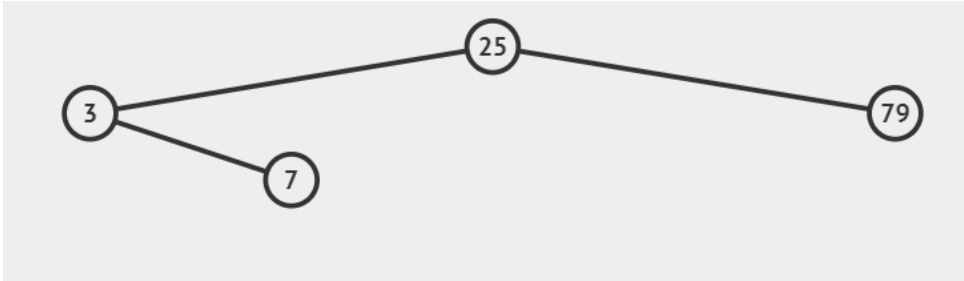
c)[20,10,40,30,50,12, Exclua o 10,80, 41, 37]



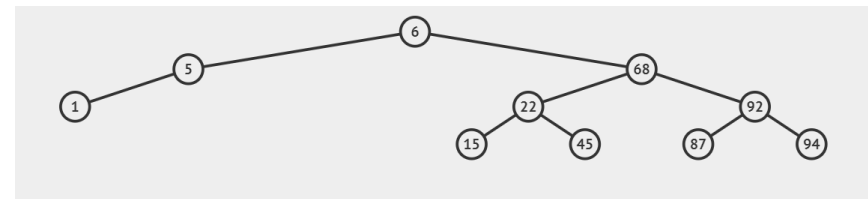
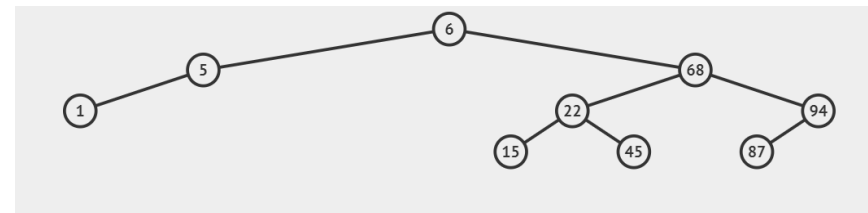
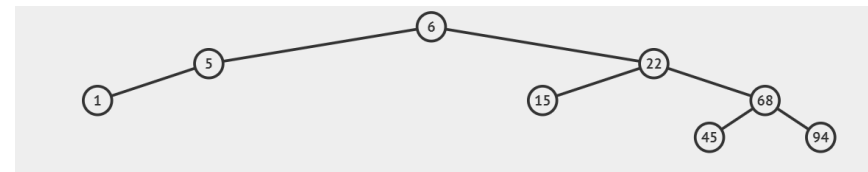
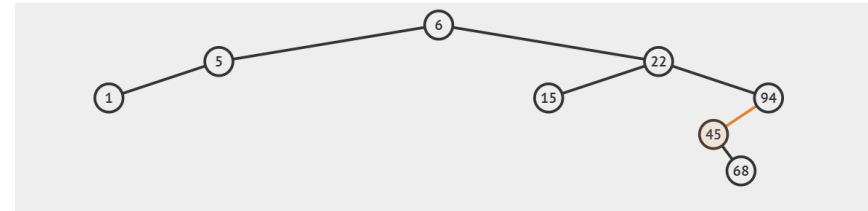
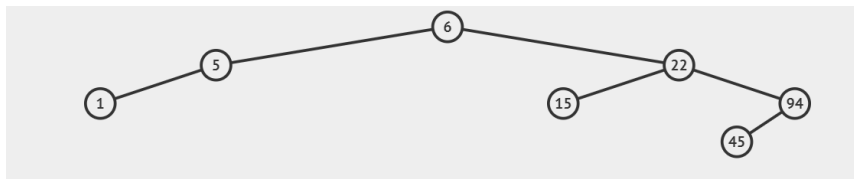
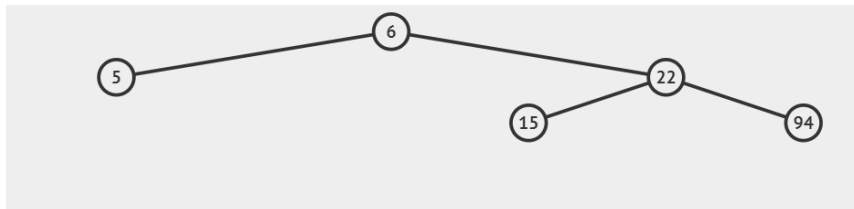
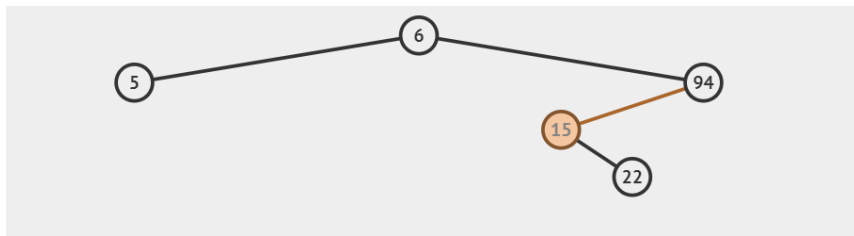
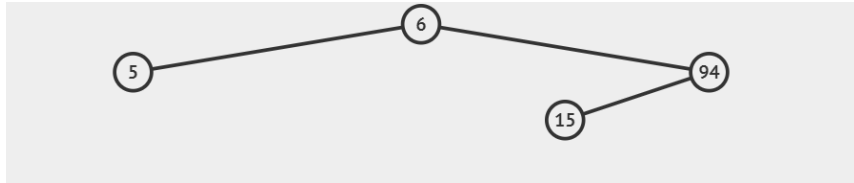
d) [10,50,4,90,20,8, Exclua o 90, Exclua o 50, 12, 9, 35]



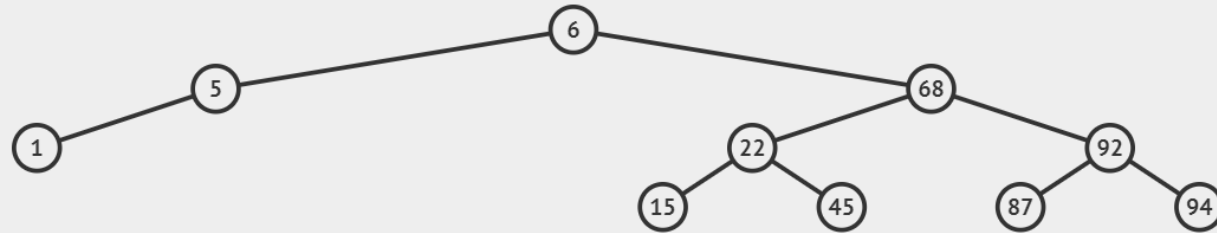
e) [79,25,3,7,28, Exclua o 7,80, 91, Exclua o 25, 82]



a)[06, 05, 94,15,22,01,45,68,87,92,15]



a)[06, 05, 94,15,22,01,45,68,87,92,15]



Insert 92

The tree is now balanced.

```
insert v
check balance factor of this and its children
case1: this.rotateRight
case2: this.left.rotateLeft, this.rotateRight
case3: this.rotateLeft
case4: this.right.rotateRight, this.rotateLeft
this is balanced
```



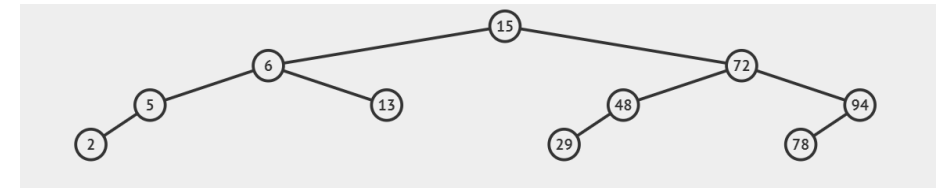
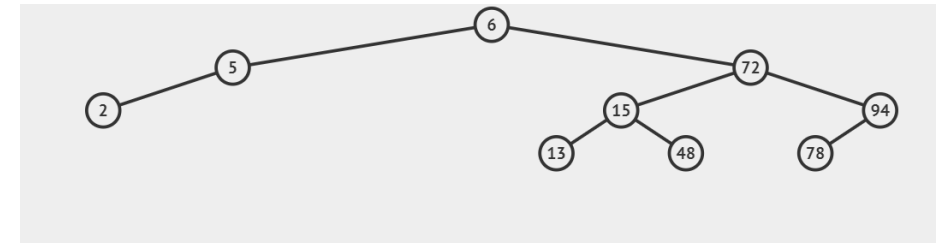
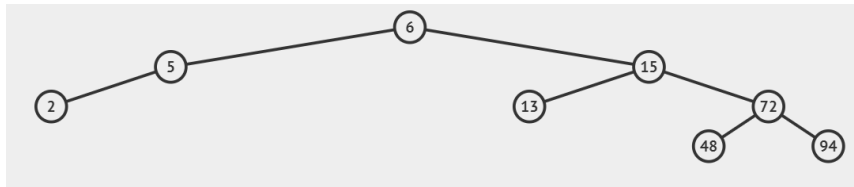
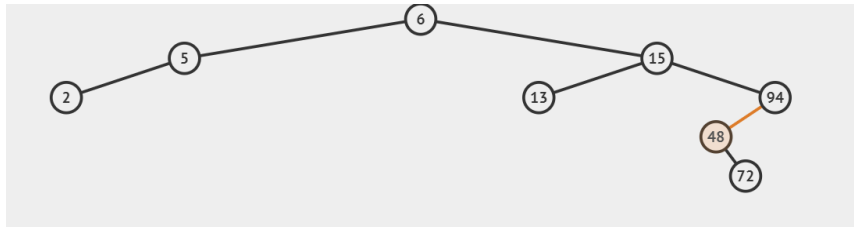
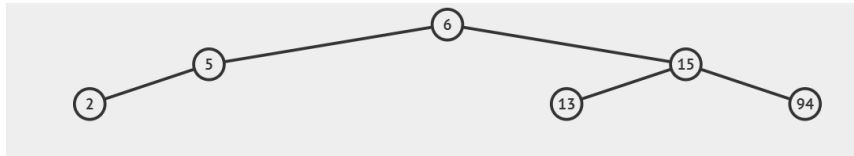
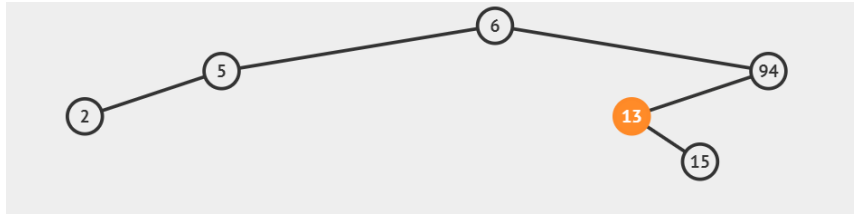
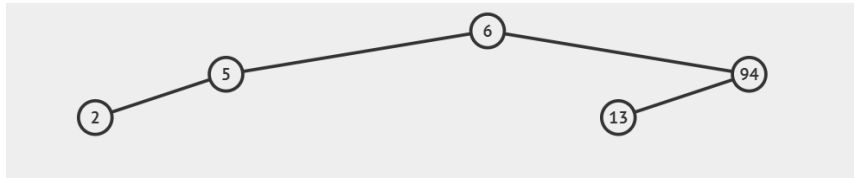
Create
Search
Insert
Remove
Predec-/Succ-essor
Tree Traversal

v = 15

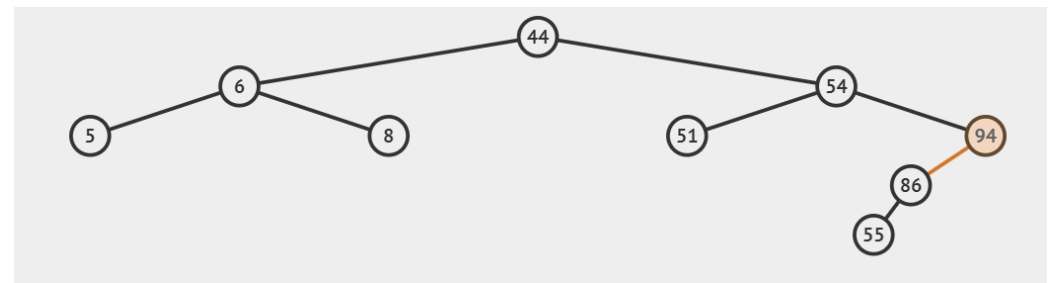
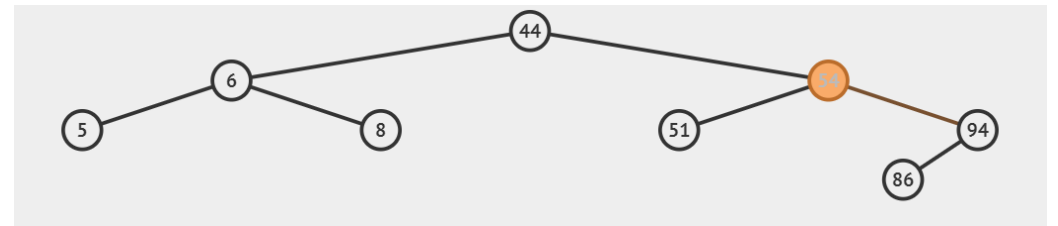
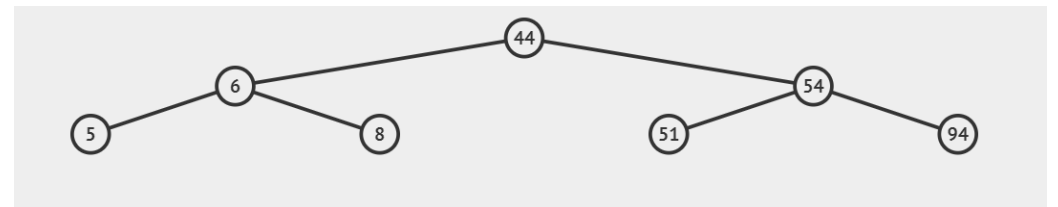
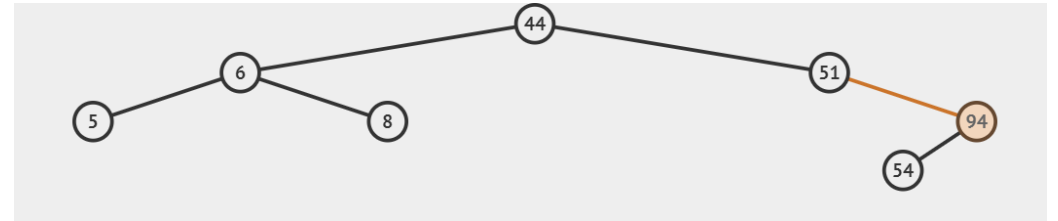
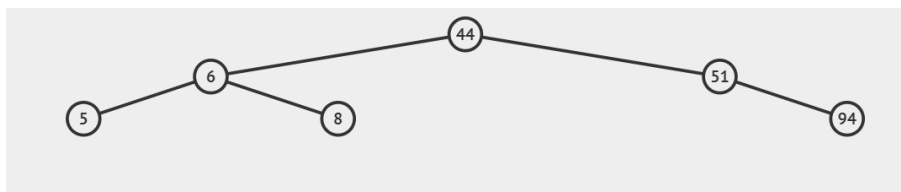
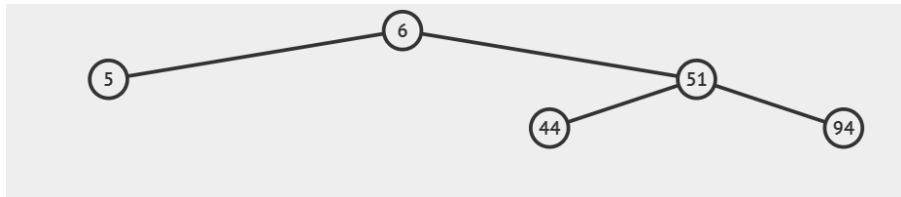
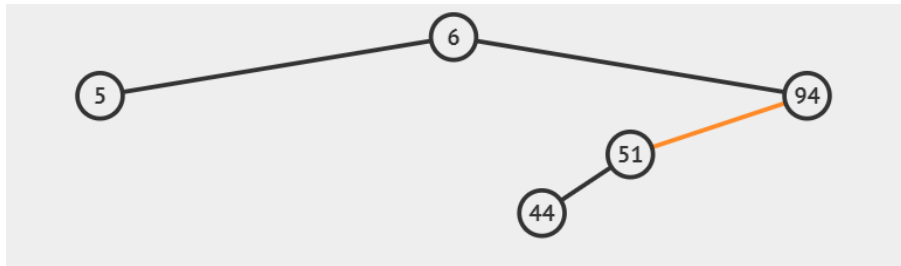
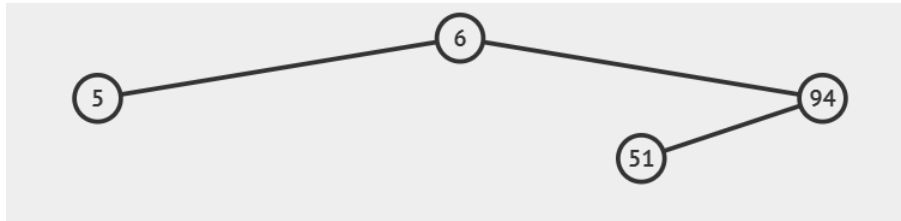
Go

No duplicate vertex allowed!

b)[06, 05, 94,02,13,15,48,72,78,29,51]



c) [06, 05, 94, 51, 44, 08, 54, 86, 55, 38, 92]



c) [06, 05, 94, 51, 44, 08, 54, 86, 55, 38, 92]

