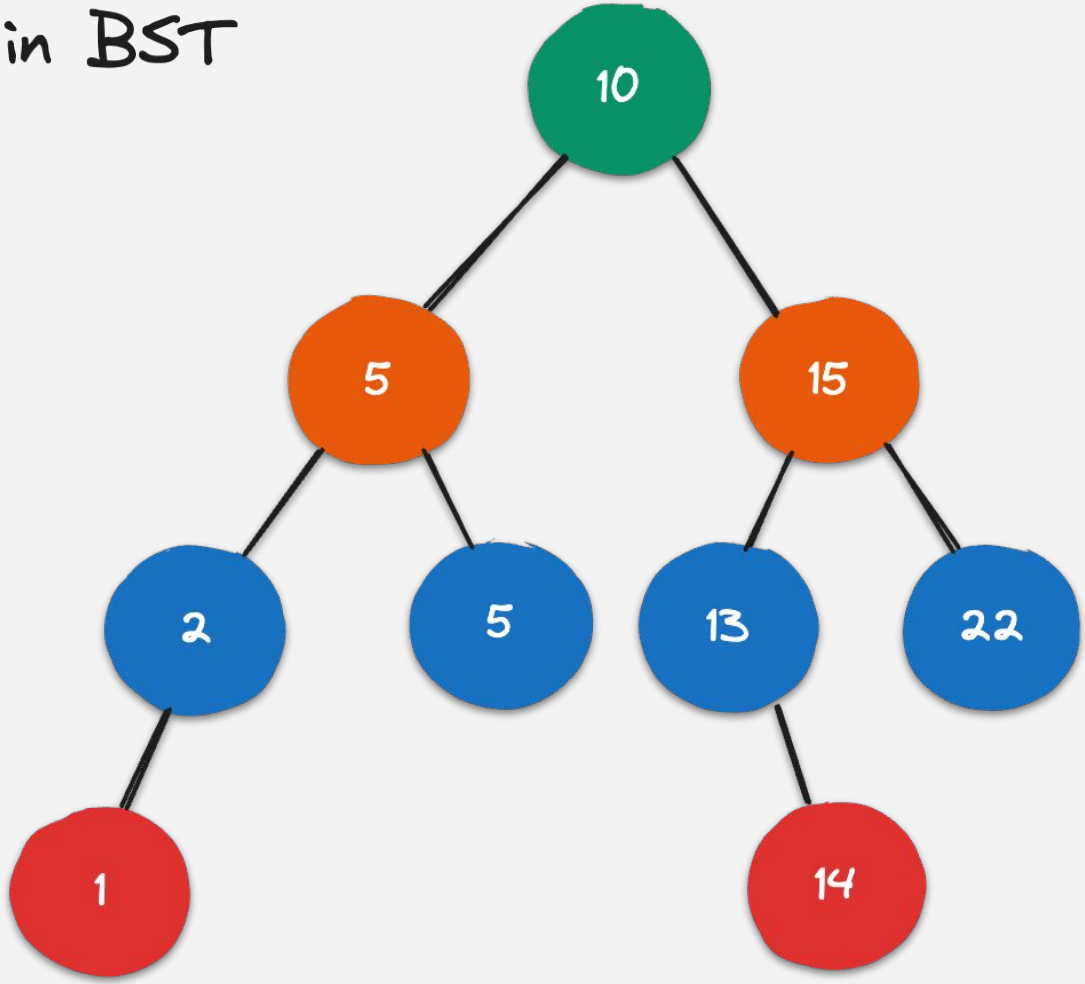
Challenge 01

Find closest value in BST

target = 12



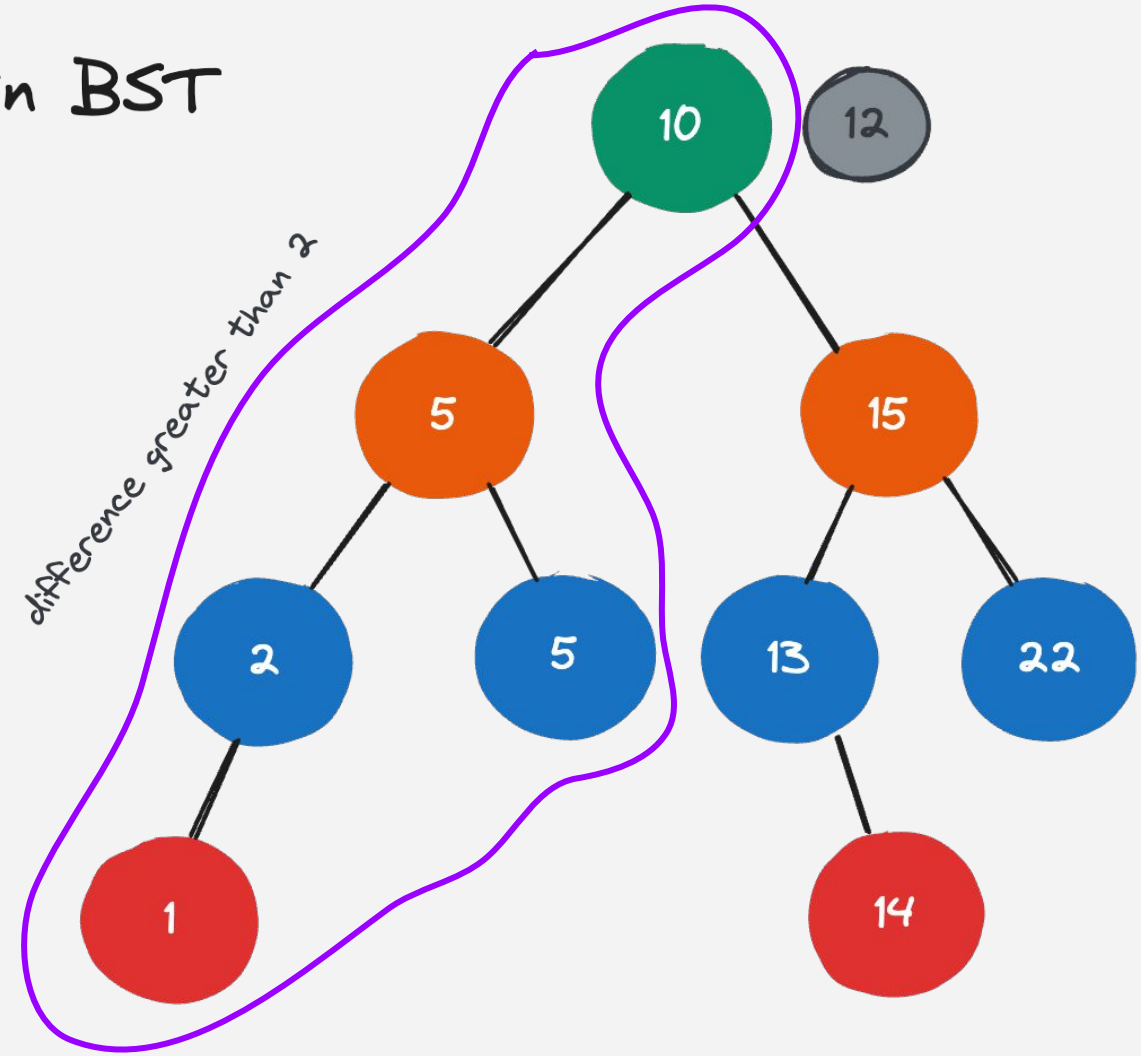
Find closest value in BST

target = 12

Step = 0

closest = root value (10)

| 10 - 12 | = 2



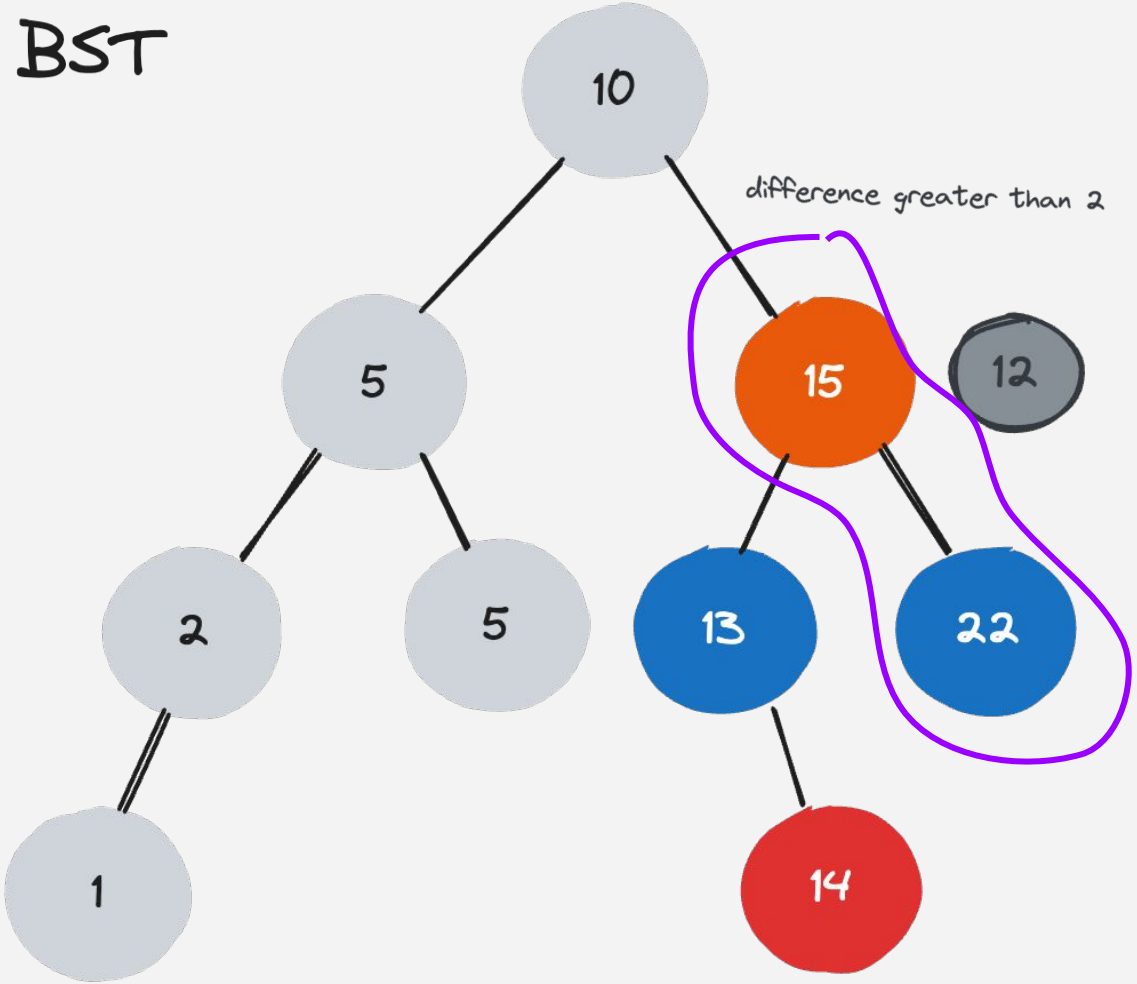
Find closest value in BST

target = 12

Step = 1

closest = 10

| 10 - 12 | = 2 < | 15 - 12 | = 3



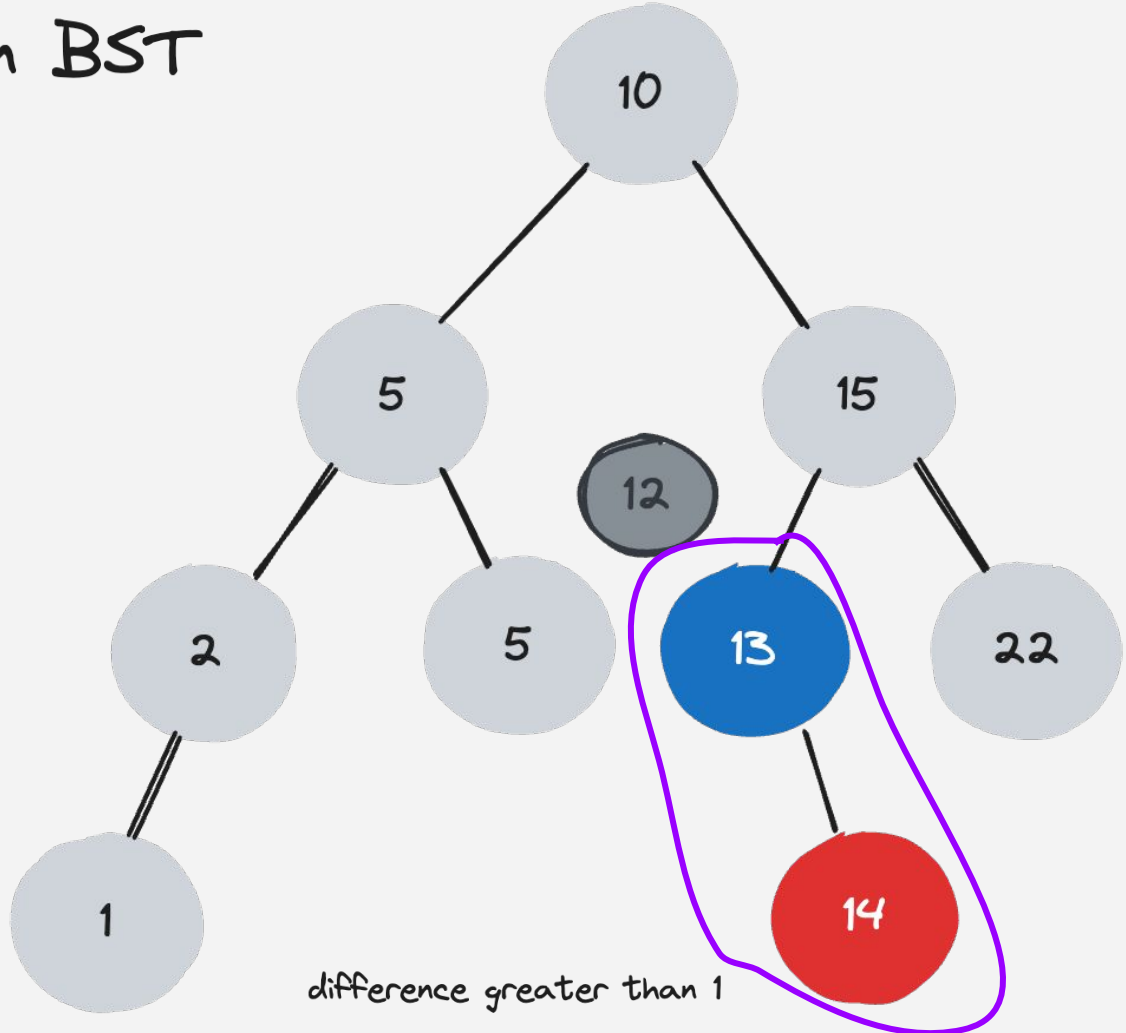
Find closest value in BST

target = 12

Step = 2

~~closest = 10~~ closest = 13

| 10 - 12 | = 2 > | 13 - 12 | = 1



Find closest value in BST

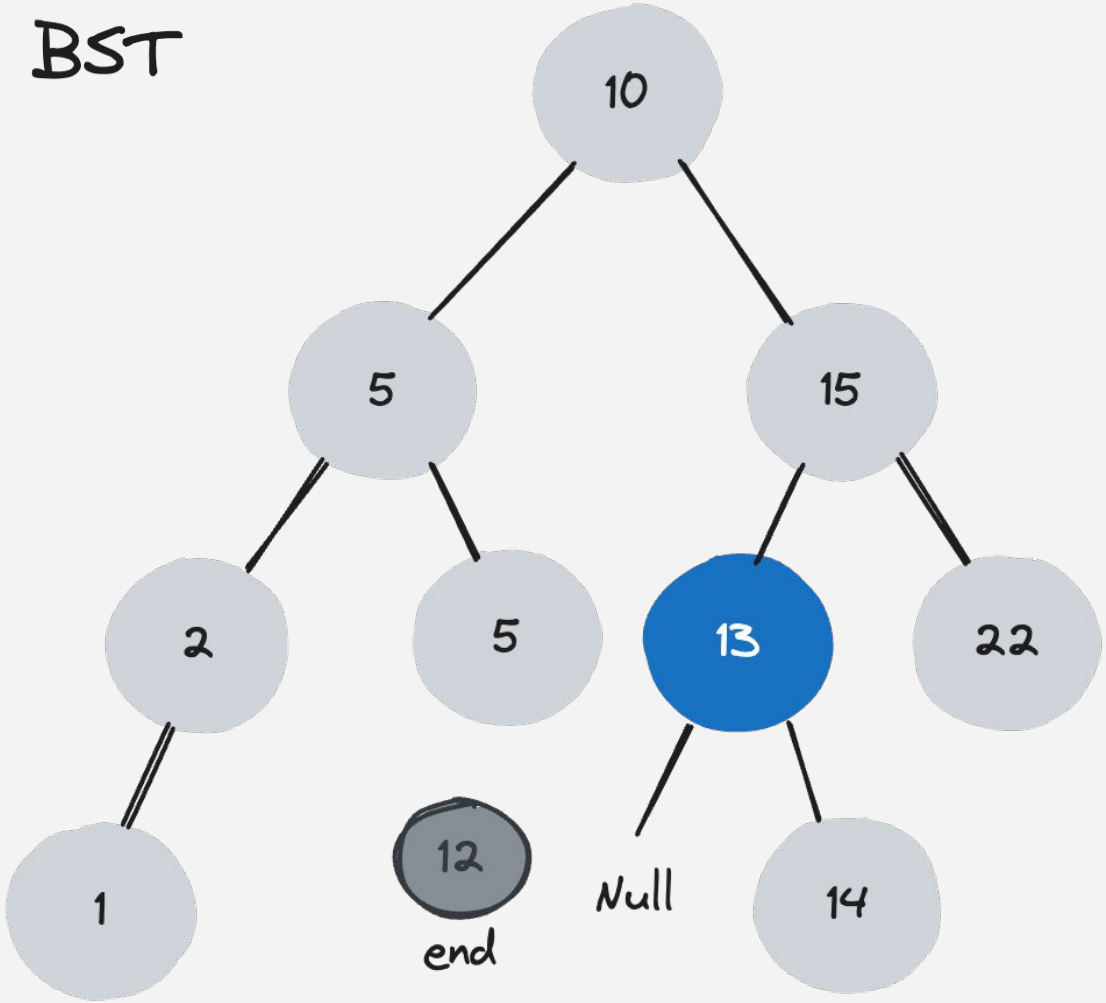
target = 12

Step = 3

return closest = 13

Avg:
Time = O(log N)
Space = O(1)

Worst:
Time = O(N)
Space = O(1)



Challenge 2

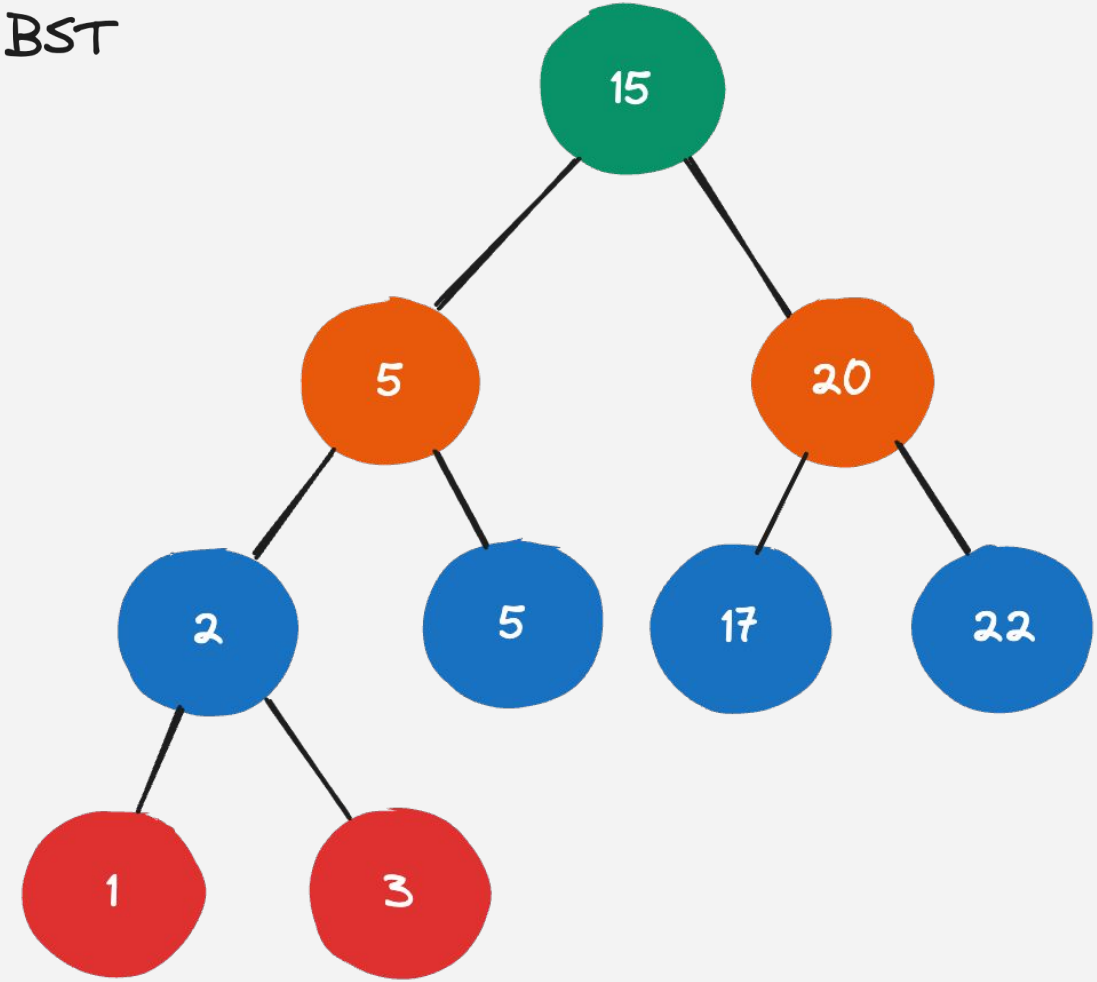
Find Kth Largest Value in BST

k = 3

output = 17

Idea 01: in-order traversal

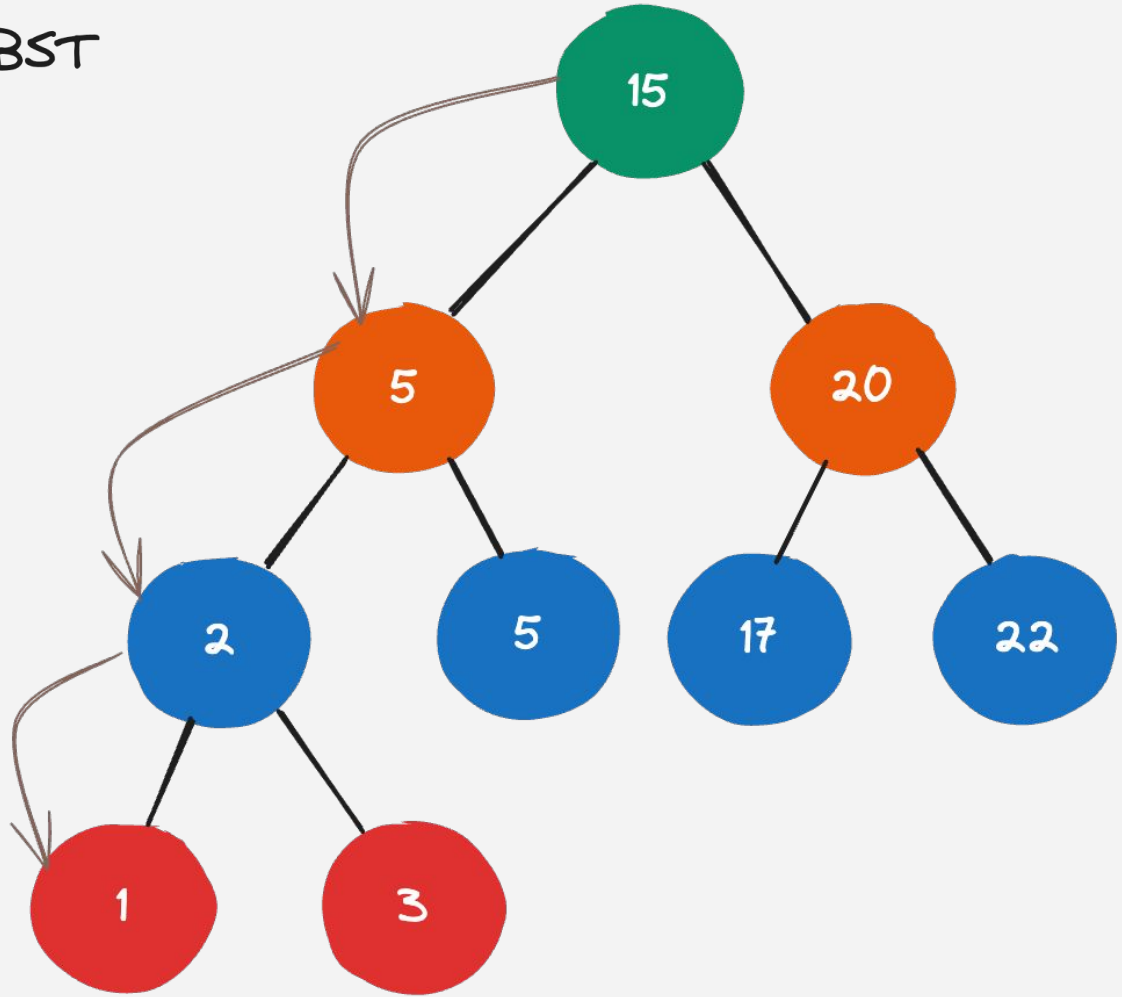
Left
Visit
Right



Find Kth Largest Value in BST

k = 3

output = 17



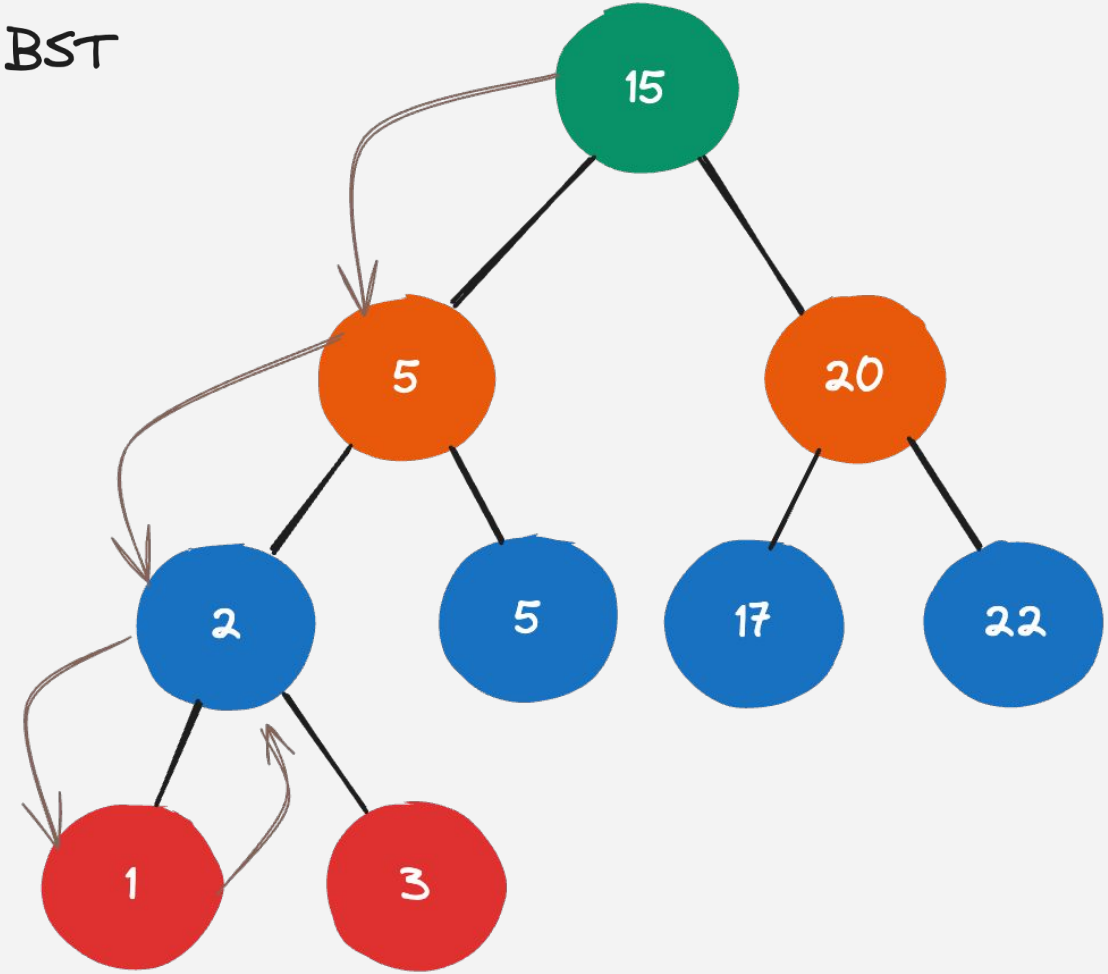
Idea 01: in-order traversal

Left
Visit
Right

Find Kth Largest Value in BST

k = 3

output = 17



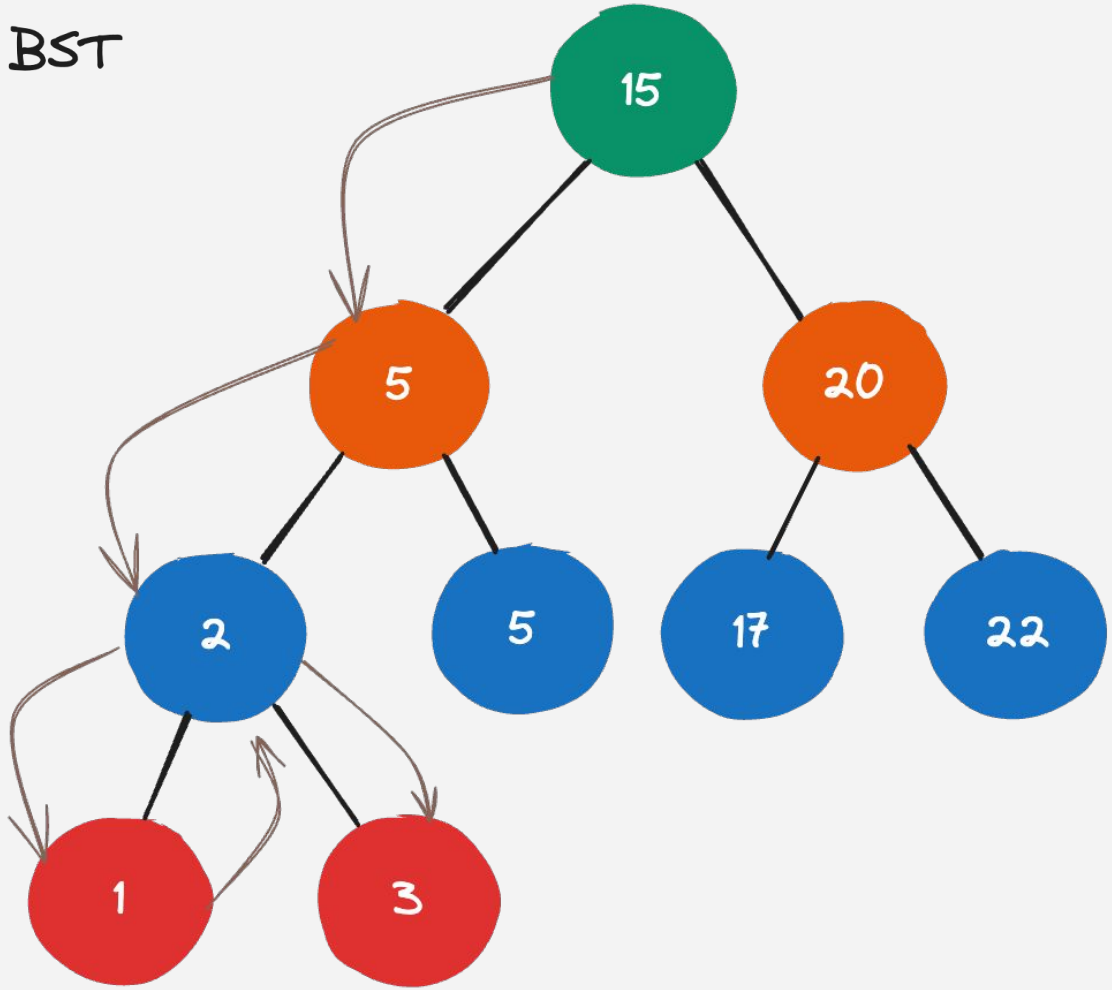
Idea 01: in-order traversal

Left
Visit
Right

Find Kth Largest Value in BST

k = 3

output = 17



Idea 01: in-order traversal

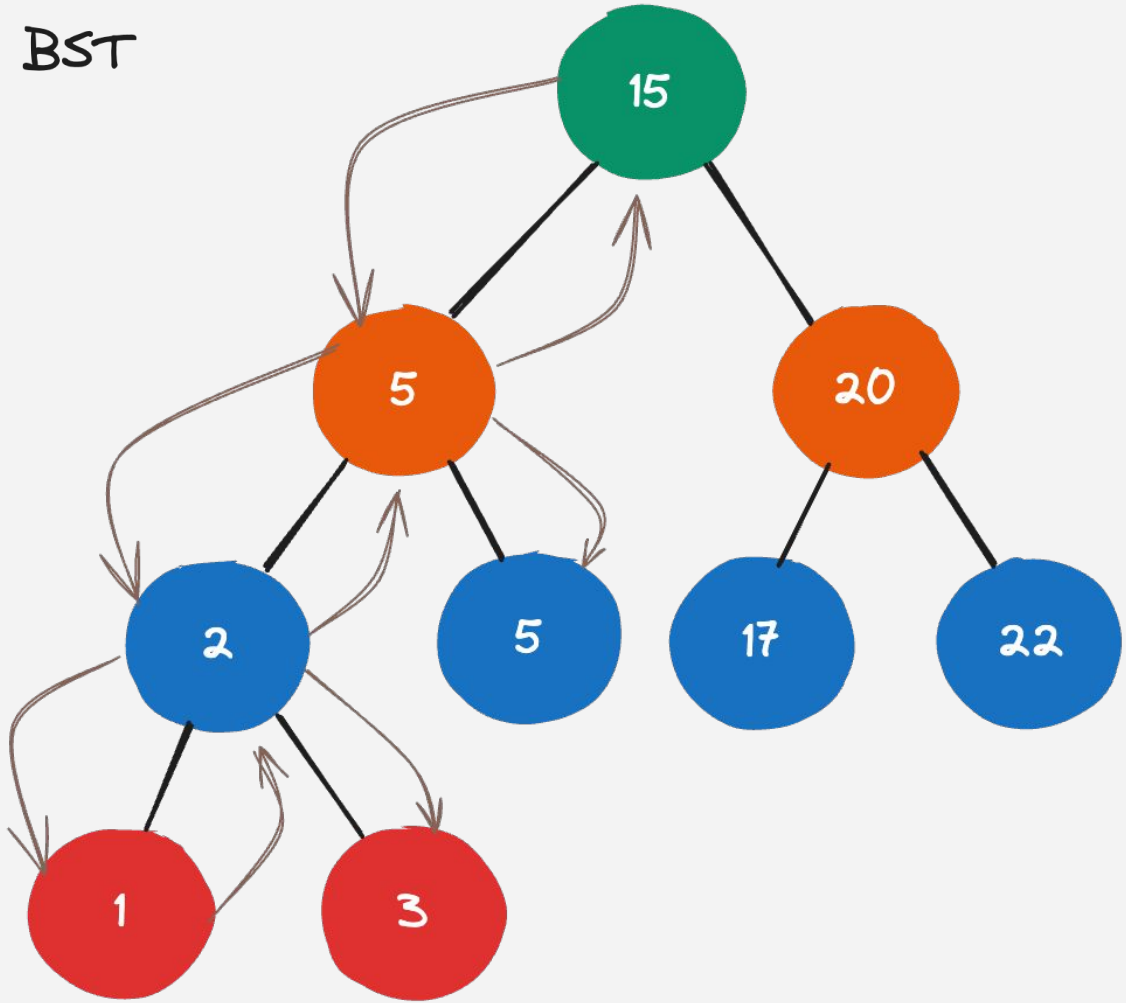
Left
Visit
Right

1 2 3

Find Kth Largest Value in BST

k = 3

output = 17



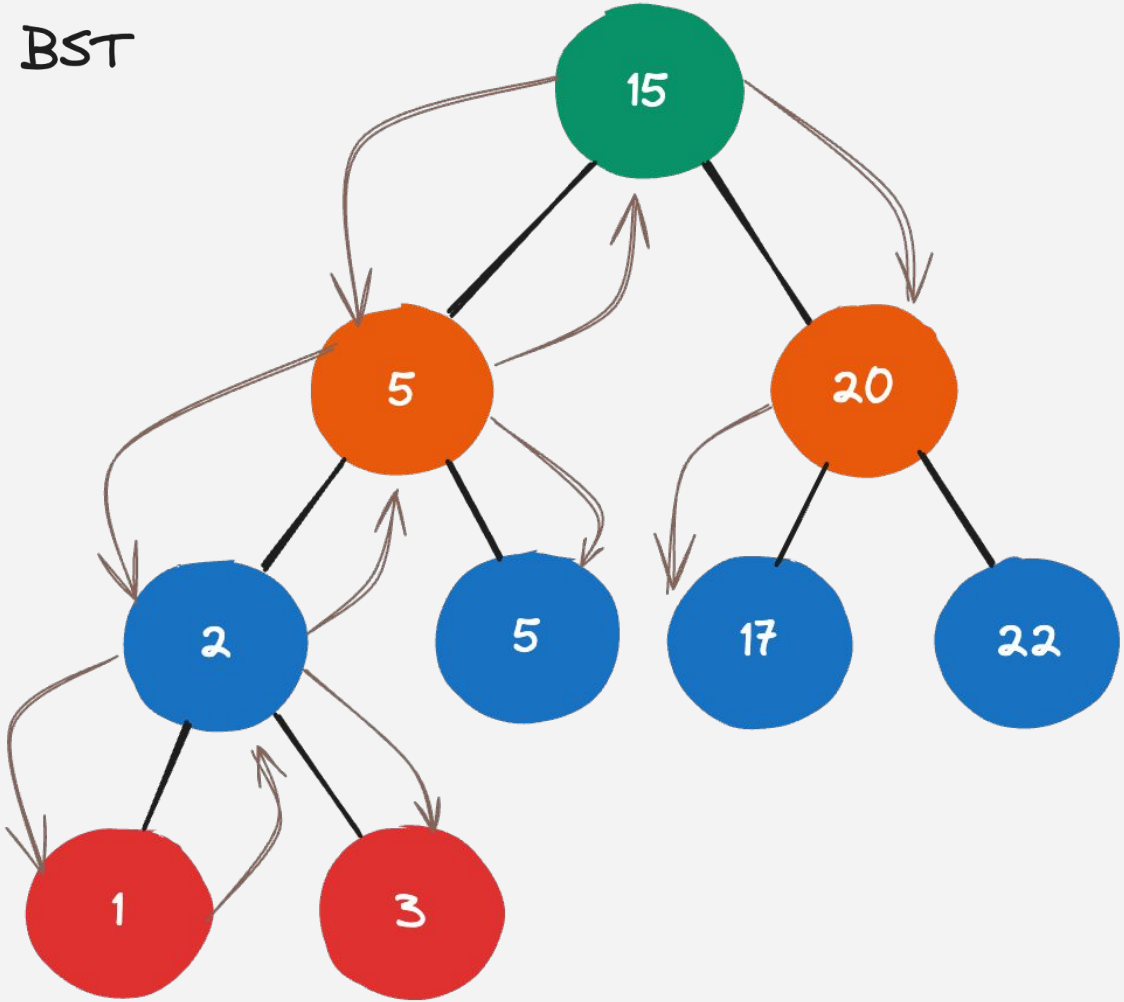
Idea 01: in-order traversal

Left
Visit
Right

Find Kth Largest Value in BST

k = 3

output = 17



Idea 01: in-order traversal

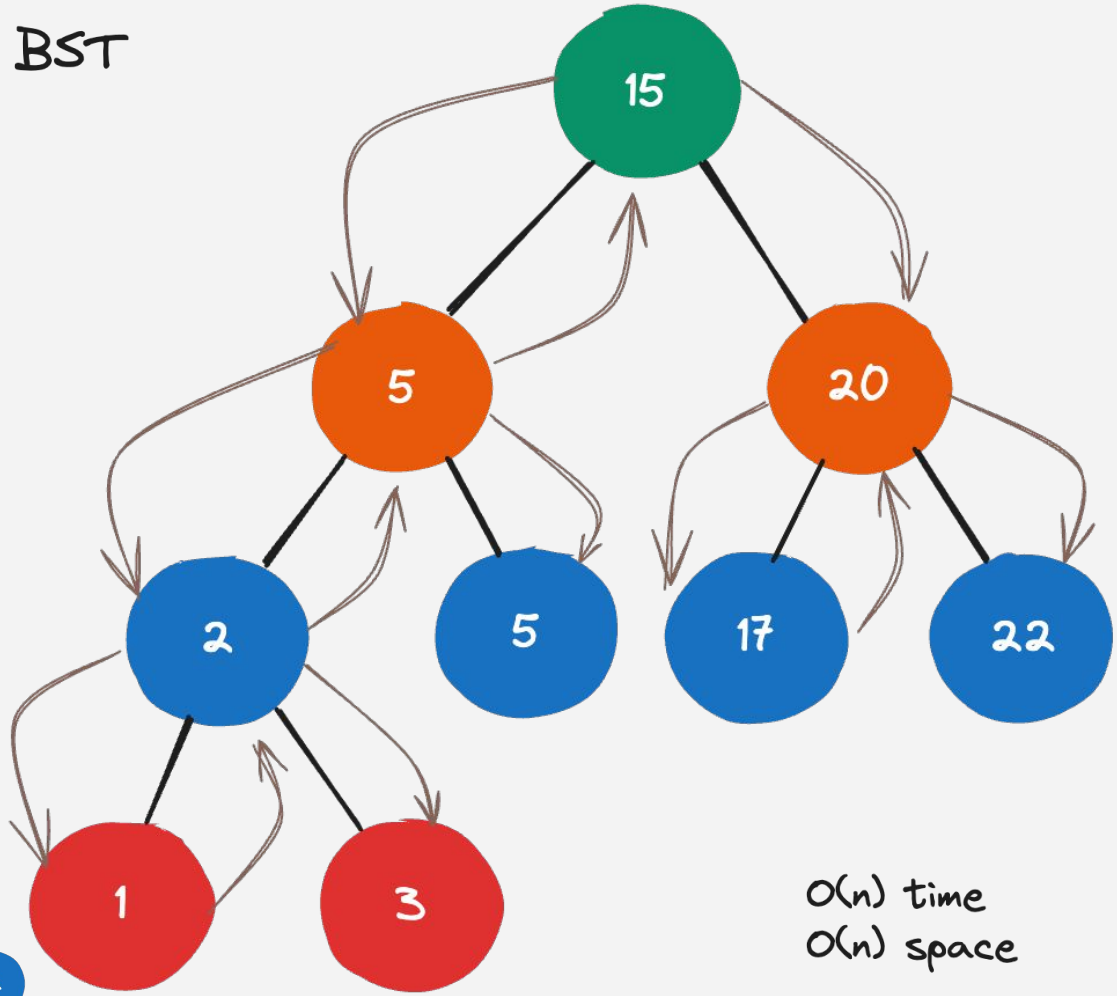
Left
Visit
Right

1 2 3 5 5 15 17

Find Kth Largest Value in BST

k = 3

output = 17



Idea 01: in-order traversal

Left
Visit
Right

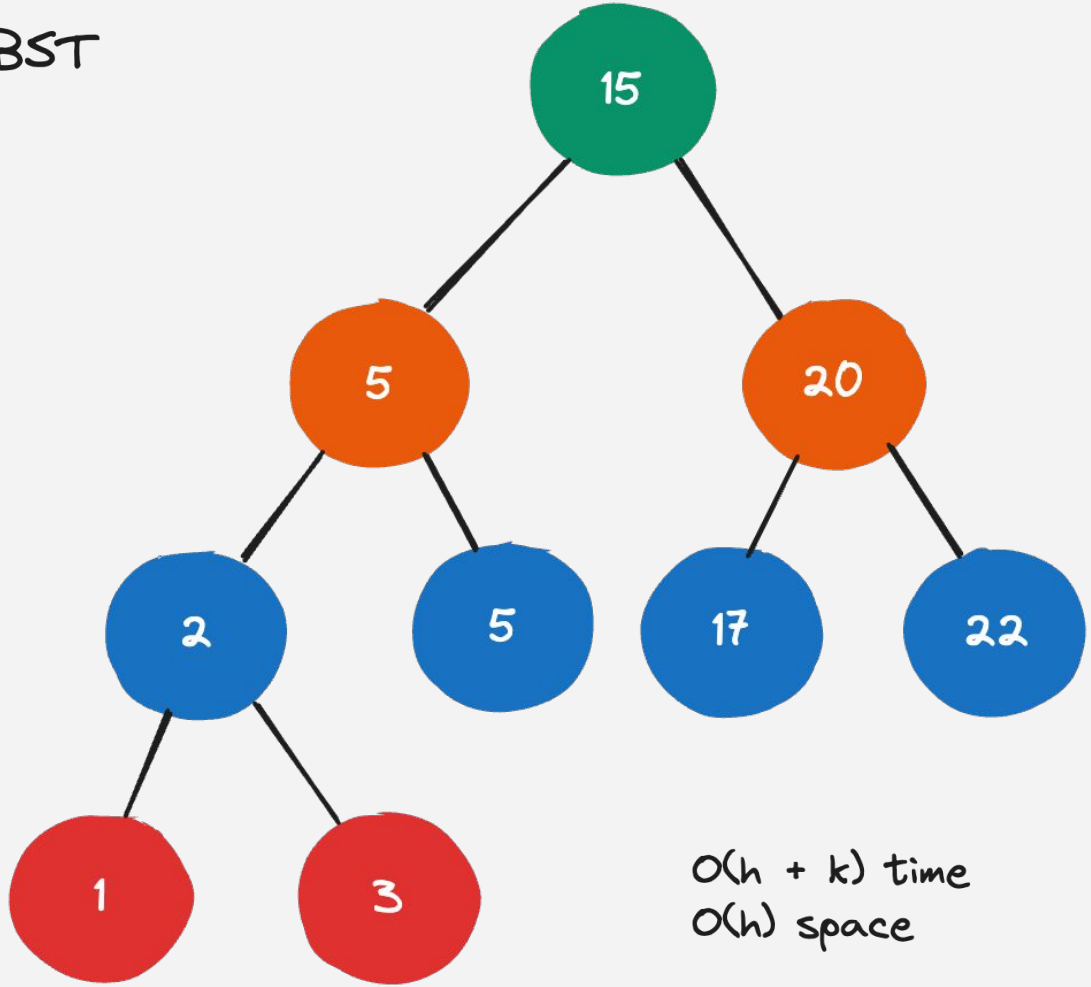
k = 3

O(n) time
O(n) space

Find Kth Largest Value in BST

k = 3

output = 17



Idea 02: reverse in-order traversal

Right
Visit
Left

O(h + k) time
O(h) space

Como avaliar e comparar algoritmos?
Avaliação manual usando diferentes cenários? Como garantir o mesmo conjunto de entrada?
Quantos experimentos são necessários para investigar e obter um intervalo de confiança desejado?

Objetivo

Avaliar o desempenho de dois algoritmos fornecidos (*solver_closest.ipynb* e *solver_kth_largest.ipynb*) considerando diversas entradas aleatórias e reproduzíveis, variando o tamanho do vetor de entrada até um valor N grande.

Requisitos

1. Instrumentar os códigos fornecidos com o módulo ***time*** para medir o tempo de execução.
2. Realizar testes com vetores de tamanhos variados.
3. Para cada tamanho de vetor, realizar múltiplas execuções e calcular o tempo médio e o intervalo de confiança.
4. Gerar gráficos que relacionam:
    a. Tamanho do vetor (eixo x).
    b. Tempo médio de execução (eixo y).
    c. Intervalos de confiança como barras de erro no gráfico.
5. Compartilhar o trabalho em um repositório no GitHub, incluindo:
    a. Código-fonte (com instruções de execução e explicação de tudo o que foi feito, incluindo resultados, no README).
    b. Arquivos gerados (gráficos e logs, se aplicável).
    c. Um vídeo de até 5 minutos explicando a análise realizada (condição necessária/majoritária para avaliação)
    d. Certifique-se de que os dados gerados sejam aleatórios, mas reproduzíveis (exemplo: configure um seed com numpy.random.seed).
    e. Tamanho N: Sugestão de *pelo menos* 10^6, se viável para execução.

Avaliação

No repositório da disciplina (semana 9) são dados:

1. Os notebooks com as soluções dos dois desafios apresentados, nomeadamente: *solver_closest.ipynb* e *solver_kth_largest.ipynb*
2. Um notebook auxiliar para a geração dos gráficos e reprodução dos resultados: main.ipynb
3. Trabalho individual
4. 2 pontos na unidade 2.
5. Prazo de entrega: 11/12 as 23h59
6. Submissão: link do repositório no Github correspondente a tarefa