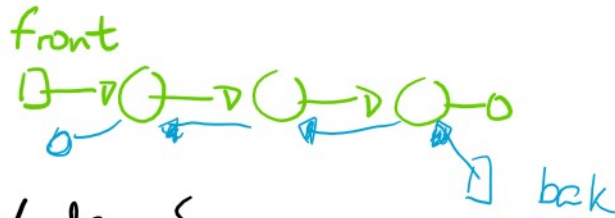


Lineart:



```
class CharNode {
```

```
CharNode * m_neste; // rekursiv struktur
CharNode * m_forrige;
```

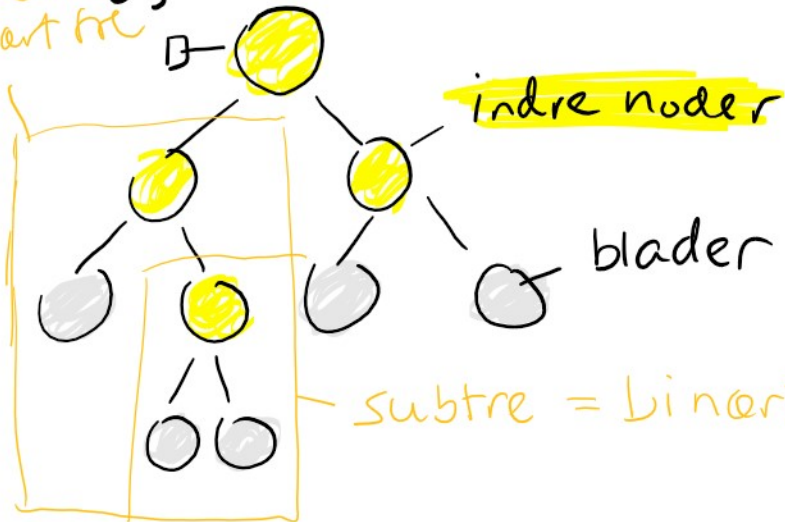
3;

```
class BinaryTree {
```

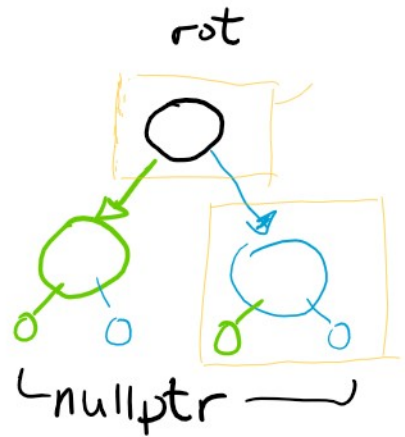
```
BinaryTree* m-left;
```

BinaryTree \* m\_right;

35



subtree = linært tre



nivå:  $0 \quad 2^0 = 1$

$$1 \quad 2' = 2$$
$$2 \quad 2^2 = 4$$
$$3 \quad 2^3 = 8$$


Sub linear  $O(\sqrt{n})$

Bindert sok: kortere vei til blad

- hvis vi kan klare oss med en sammenheng

- hvis vi kan klare oss med en sammenheng  
på hvert nivå, blir det raskere ~~se~~.
- Hvor mye raskere?

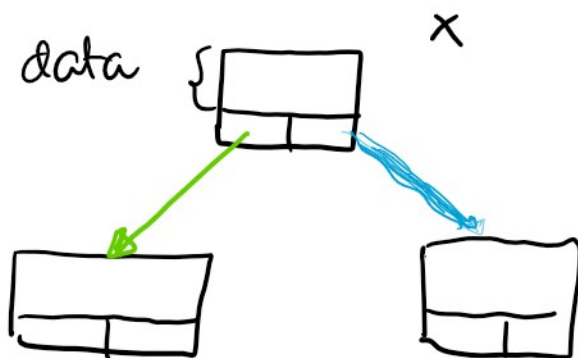
$$\log_2 8 = 3 \quad (2^3 = 8)$$

$$\log_2 4 = 2$$

Binært tre kan altså brukes til søking.

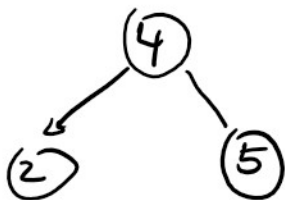
Binert soketre

- krever: litt jobb når vi setter inn noder
- definisjon: For enhver node i treet gjelder:


$$x \rightarrow \text{left} \rightarrow \text{data} < x \rightarrow \text{data}$$

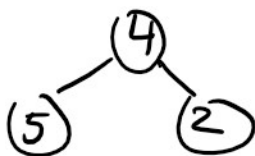
$x \rightarrow \text{right} \rightarrow \text{data} \geq x - \text{data}$

Kun  $\rightarrow$  hvis vi ikke tillater duplikater



V

binært søke tre



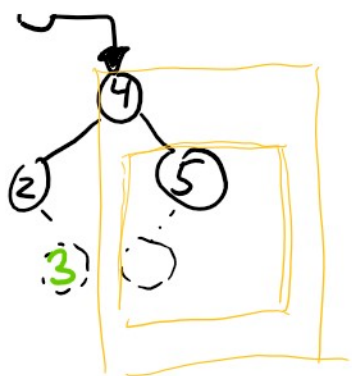
X

ikke binert skoletre



✓

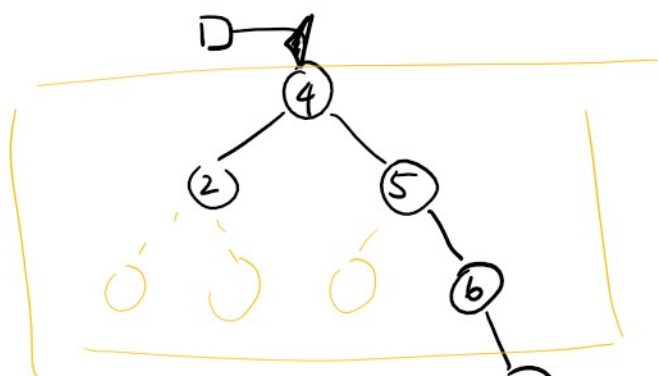
Hvordan plassere en ny node  
? 1.1. 1.2. 1.3. 1.4. 1.5. 1.6. 1.7. 1.8. 1.9. 1.10. 1.11. 1.12. 1.13. 1.14. 1.15. 1.16. 1.17. 1.18. 1.19. 1.20. 1.21. 1.22. 1.23. 1.24. 1.25. 1.26. 1.27. 1.28. 1.29. 1.30. 1.31. 1.32. 1.33. 1.34. 1.35. 1.36. 1.37. 1.38. 1.39. 1.40. 1.41. 1.42. 1.43. 1.44. 1.45. 1.46. 1.47. 1.48. 1.49. 1.50. 1.51. 1.52. 1.53. 1.54. 1.55. 1.56. 1.57. 1.58. 1.59. 1.60. 1.61. 1.62. 1.63. 1.64. 1.65. 1.66. 1.67. 1.68. 1.69. 1.70. 1.71. 1.72. 1.73. 1.74. 1.75. 1.76. 1.77. 1.78. 1.79. 1.80. 1.81. 1.82. 1.83. 1.84. 1.85. 1.86. 1.87. 1.88. 1.89. 1.90. 1.91. 1.92. 1.93. 1.94. 1.95. 1.96. 1.97. 1.98. 1.99. 2.00. 2.01. 2.02. 2.03. 2.04. 2.05. 2.06. 2.07. 2.08. 2.09. 2.10. 2.11. 2.12. 2.13. 2.14. 2.15. 2.16. 2.17. 2.18. 2.19. 2.20. 2.21. 2.22. 2.23. 2.24. 2.25. 2.26. 2.27. 2.28. 2.29. 2.30. 2.31. 2.32. 2.33. 2.34. 2.35. 2.36. 2.37. 2.38. 2.39. 2.40. 2.41. 2.42. 2.43. 2.44. 2.45. 2.46. 2.47. 2.48. 2.49. 2.50. 2.51. 2.52. 2.53. 2.54. 2.55. 2.56. 2.57. 2.58. 2.59. 2.60. 2.61. 2.62. 2.63. 2.64. 2.65. 2.66. 2.67. 2.68. 2.69. 2.70. 2.71. 2.72. 2.73. 2.74. 2.75. 2.76. 2.77. 2.78. 2.79. 2.80. 2.81. 2.82. 2.83. 2.84. 2.85. 2.86. 2.87. 2.88. 2.89. 2.90. 2.91. 2.92. 2.93. 2.94. 2.95. 2.96. 2.97. 2.98. 2.99. 3.00. 3.01. 3.02. 3.03. 3.04. 3.05. 3.06. 3.07. 3.08. 3.09. 3.10. 3.11. 3.12. 3.13. 3.14. 3.15. 3.16. 3.17. 3.18. 3.19. 3.20. 3.21. 3.22. 3.23. 3.24. 3.25. 3.26. 3.27. 3.28. 3.29. 3.30. 3.31. 3.32. 3.33. 3.34. 3.35. 3.36. 3.37. 3.38. 3.39. 3.40. 3.41. 3.42. 3.43. 3.44. 3.45. 3.46. 3.47. 3.48. 3.49. 3.50. 3.51. 3.52. 3.53. 3.54. 3.55. 3.56. 3.57. 3.58. 3.59. 3.60. 3.61. 3.62. 3.63. 3.64. 3.65. 3.66. 3.67. 3.68. 3.69. 3.70. 3.71. 3.72. 3.73. 3.74. 3.75. 3.76. 3.77. 3.78. 3.79. 3.80. 3.81. 3.82. 3.83. 3.84. 3.85. 3.86. 3.87. 3.88. 3.89. 3.90. 3.91. 3.92. 3.93. 3.94. 3.95. 3.96. 3.97. 3.98. 3.99. 4.00. 4.01. 4.02. 4.03. 4.04. 4.05. 4.06. 4.07. 4.08. 4.09. 4.10. 4.11. 4.12. 4.13. 4.14. 4.15. 4.16. 4.17. 4.18. 4.19. 4.20. 4.21. 4.22. 4.23. 4.24. 4.25. 4.26. 4.27. 4.28. 4.29. 4.30. 4.31. 4.32. 4.33. 4.34. 4.35. 4.36. 4.37. 4.38. 4.39. 4.40. 4.41. 4.42. 4.43. 4.44. 4.45. 4.46. 4.47. 4.48. 4.49. 4.50. 4.51. 4.52. 4.53. 4.54. 4.55. 4.56. 4.57. 4.58. 4.59. 4.60. 4.61. 4.62. 4.63. 4.64. 4.65. 4.66. 4.67. 4.68. 4.69. 4.70. 4.71. 4.72. 4.73. 4.74. 4.75. 4.76. 4.77. 4.78. 4.79. 4.80. 4.81. 4.82. 4.83. 4.84. 4.85. 4.86. 4.87. 4.88. 4.89. 4.90. 4.91. 4.92. 4.93. 4.94. 4.95. 4.96. 4.97. 4.98. 4.99. 5.00. 5.01. 5.02. 5.03. 5.04. 5.05. 5.06. 5.07. 5.08. 5.09. 5.10. 5.11. 5.12. 5.13. 5.14. 5.15. 5.16. 5.17. 5.18. 5.19. 5.20. 5.21. 5.22. 5.23. 5.24. 5.25. 5.26. 5.27. 5.28. 5.29. 5.30. 5.31. 5.32. 5.33. 5.34. 5.35. 5.36. 5.37. 5.38. 5.39. 5.40. 5.41. 5.42. 5.43. 5.44. 5.45. 5.46. 5.47. 5.48. 5.49. 5.50. 5.51. 5.52. 5.53. 5.54. 5.55. 5.56. 5.57. 5.58. 5.59. 5.60. 5.61. 5.62. 5.63. 5.64. 5.65. 5.66. 5.67. 5.68. 5.69. 5.70. 5.71. 5.72. 5.73. 5.74. 5.75. 5.76. 5.77. 5.78. 5.79. 5.80. 5.81. 5.82. 5.83. 5.84. 5.85. 5.86. 5.87. 5.88. 5.89. 5.90. 5.91. 5.92. 5.93. 5.94. 5.95. 5.96. 5.97. 5.98. 5.99. 6.00. 6.01. 6.02. 6.03. 6.04. 6.05. 6.06. 6.07. 6.08. 6.09. 6.10. 6.11. 6.12. 6.13. 6.14. 6.15. 6.16. 6.17. 6.18. 6.19. 6.20. 6.21. 6.22. 6.23. 6.24. 6.25. 6.26. 6.27. 6.28. 6.29. 6.30. 6.31. 6.32. 6.33. 6.34. 6.35. 6.36. 6.37. 6.38. 6.39. 6.40. 6.41. 6.42. 6.43. 6.44. 6.45. 6.46. 6.47. 6.48. 6.49. 6.50. 6.51. 6.52. 6.53. 6.54. 6.55. 6.56. 6.57. 6.58. 6.59. 6.60. 6.61. 6.62. 6.63. 6.64. 6.65. 6.66. 6.67. 6.68. 6.69. 6.70. 6.71. 6.72. 6.73. 6.74. 6.75. 6.76. 6.77. 6.78. 6.79. 6.80. 6.81. 6.82. 6.83. 6.84. 6.85. 6.86. 6.87. 6.88. 6.89. 6.90. 6.91. 6.92. 6.93. 6.94. 6.95. 6.96. 6.97. 6.98. 6.99. 7.00. 7.01. 7.02. 7.03. 7.04. 7.05. 7.06. 7.07. 7.08. 7.09. 7.10. 7.11. 7.12. 7.13. 7.14. 7.15. 7.16. 7.17. 7.18. 7.19. 7.20. 7.21. 7.22. 7.23. 7.24. 7.25. 7.26. 7.27. 7.28. 7.29. 7.30. 7.31. 7.32. 7.33. 7.34. 7.35. 7.36. 7.37. 7.38. 7.39. 7.40. 7.41. 7.42. 7.43. 7.44. 7.45. 7.46. 7.47. 7.48. 7.49. 7.50. 7.51. 7.52. 7.53. 7.54. 7.55. 7.56. 7.57. 7.58. 7.59. 7.60. 7.61. 7.62. 7.63. 7.64. 7.65. 7.66. 7.67. 7.68. 7.69. 7.70. 7.71. 7.72. 7.73. 7.74. 7.75. 7.76. 7.77. 7.78. 7.79. 7.80. 7.81. 7.82.



✓

Hvordan plassere en ny node  
3 slik at vi fortsatt har  
et ✓ binært søketre

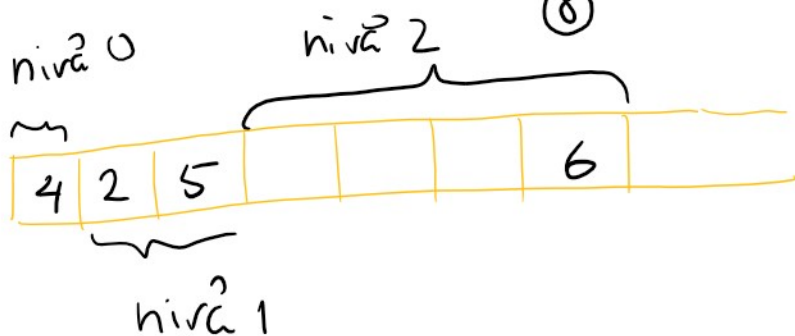
Sette inn 6, 7, 9, 8 i tur og orden. Tegn!



ganske sorterte innsett



ganske lineært



etc.

jfr. figur 4.1.

Søking, innsetting, traversering.

Gitt søkedata (heltall, char < >)

overlastes

Eksempel: Søke etter 8 på fig. 4.2

Eksempel: Søke etter 8 på fig. 4.2

5 ml 8 med rot (5)

```
BinaryNode* node = rot;  
data node->m_data  
if (data < node->data)  
{  
    node = node->m_left;
```

// søk på nytt

}

rekursjon ↙  
m\_left->find() else // data >= node->data  
{

m\_right->find() ↗  
 node = node->m\_right;  
 // søk på nytt  
}

```
BinaryNode* BinaryNode::find(int data)  
{  
    // se s. 30  
}
```

Binært søk er  $O(\log n)$

Traversering

- in order
- preorder
- postorder

```
void BinaryTree::intra()
{
    mleft -> intra();
    // besøk
    mright -> intra();
}
```

inorder traversering  $\Rightarrow$   
Stigende sortert rekkefølge  
innsetting: Les/studer