DELTA TopGun

(12) Binární halda

Luboš Zápotočný, Tomáš Faltejsek, Michal Havelka

2022

Obsah

Opakování

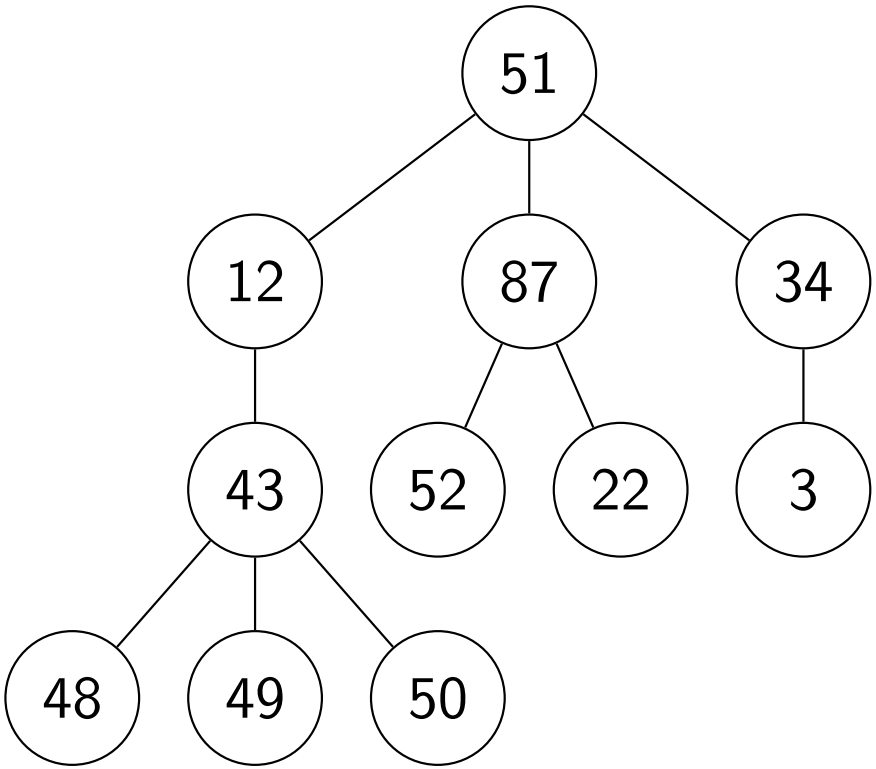
- Strom
- Zakořeněný strom
- Binární strom

Binární minimová halda

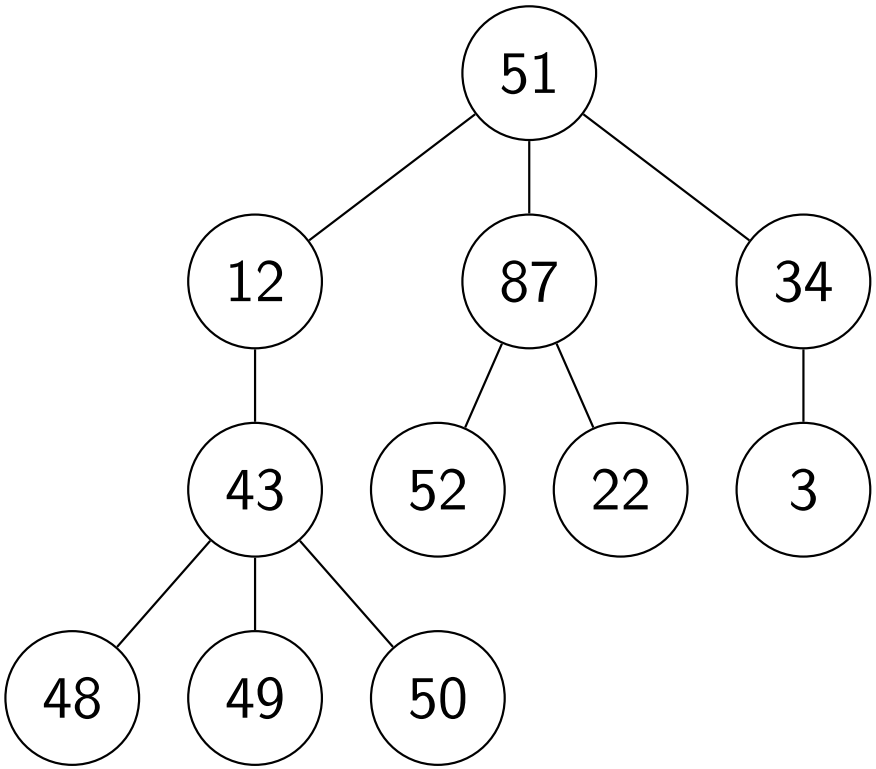
- Motivace
- Tvar haldy
- Haldové uspořádání
- Implementace pomocí pole
- Probublání nahoru
- Probublání dolů
- Vložení prvku
- Nalezení minimálního prvku
- Odstranění minimálního prvku
- Sestavení haldy

Heapsort

Strom

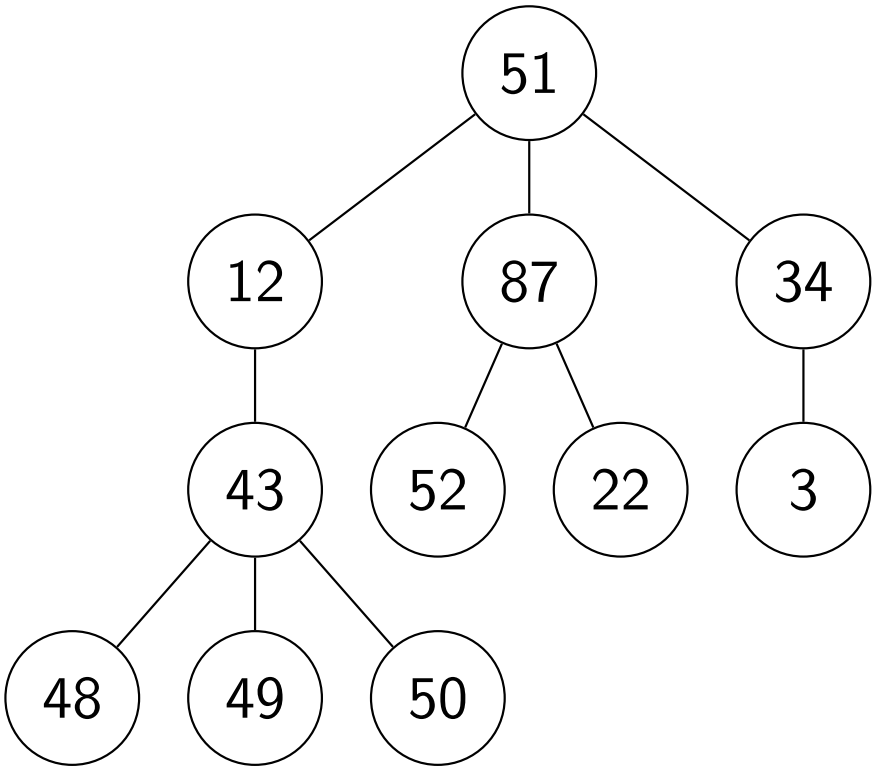


Strom



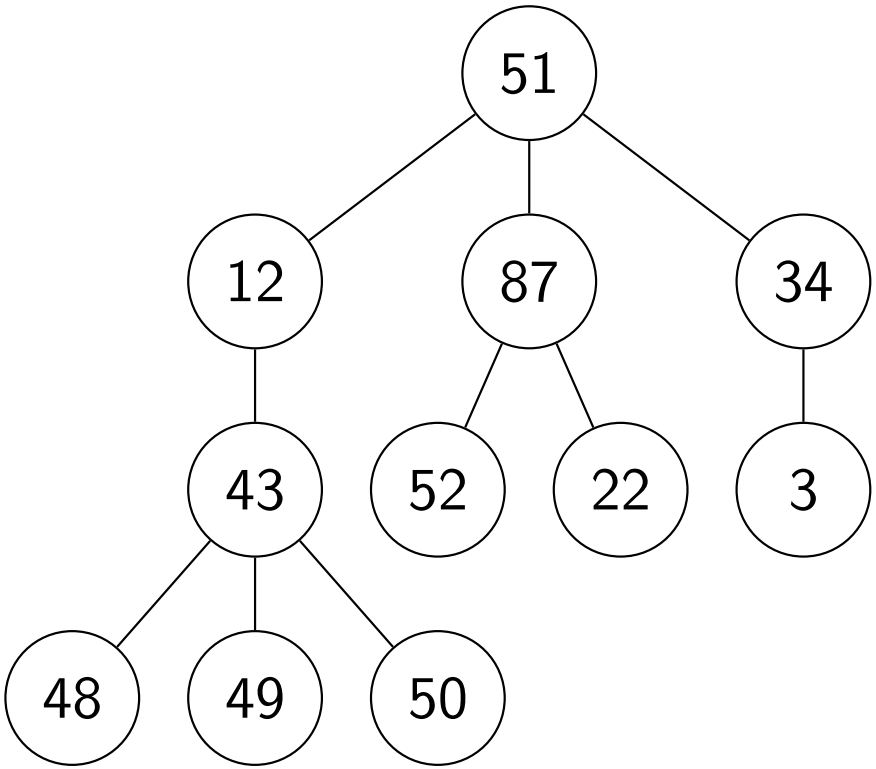
- $G = (V, E)$ je strom

Strom



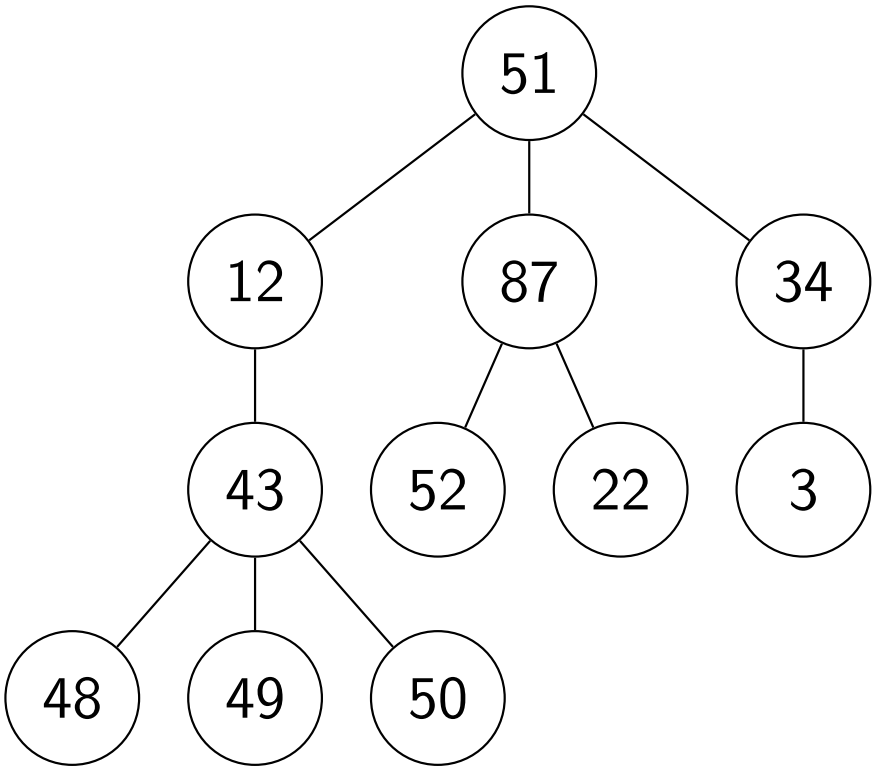
- $G = (V, E)$ je strom
- Mezi každou dvojicí vrcholů u, v existuje pouze jedna u-v cesta

Strom



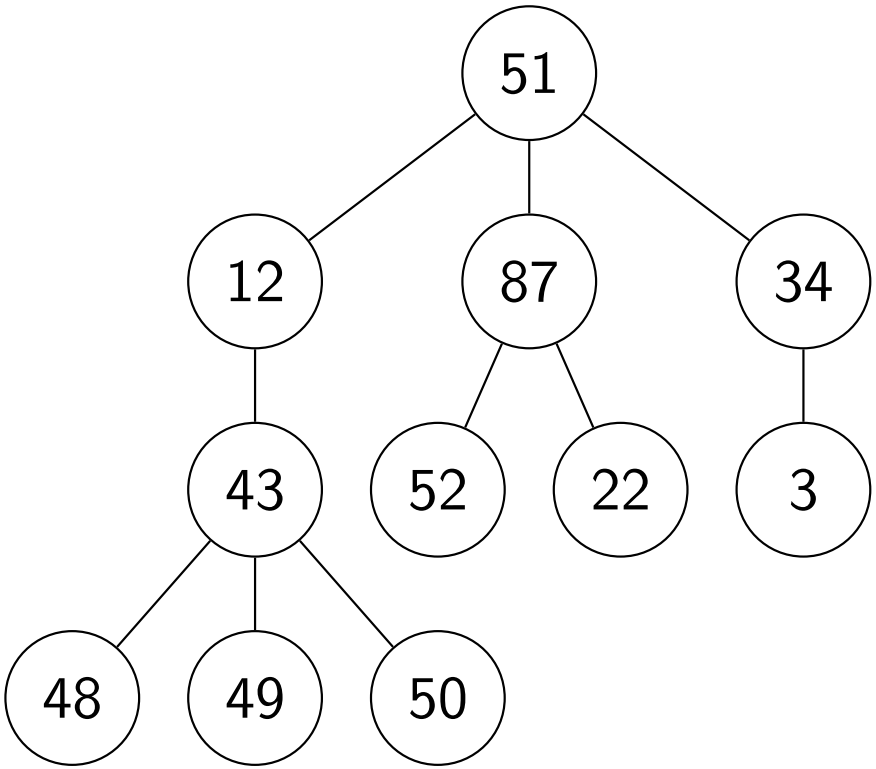
- $G = (V, E)$ je strom
- Mezi každou dvojicí vrcholů u, v existuje pouze jedna u-v cesta
- G je souvislý a odebráním jakékoli hrany se stane nesouvislým

Strom



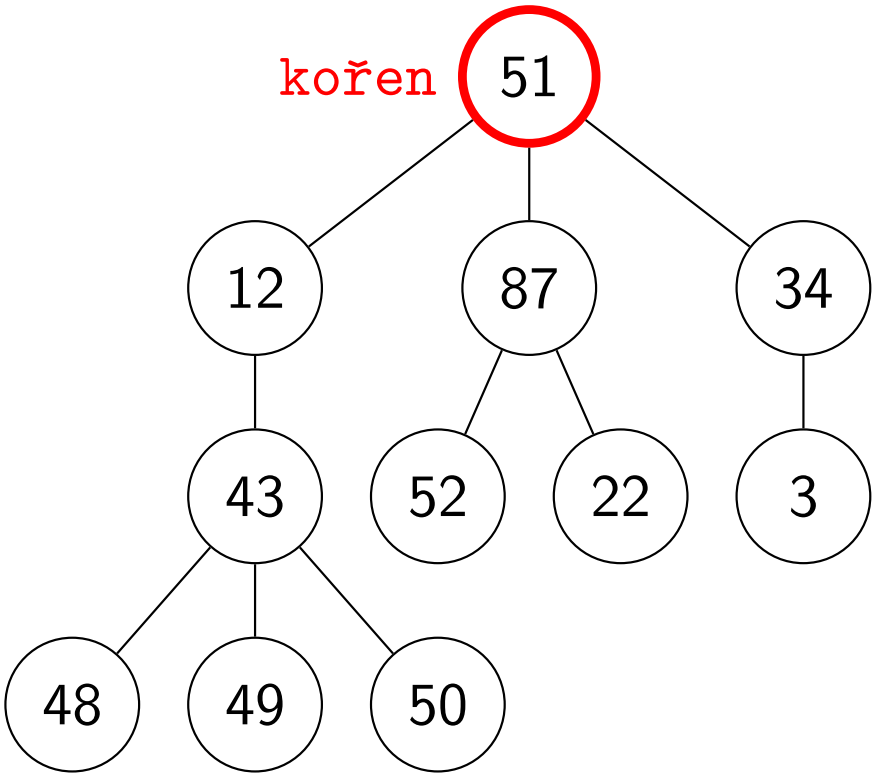
- $G = (V, E)$ je strom
- Mezi každou dvojicí vrcholů u, v existuje pouze jedna u-v cesta
- G je souvislý a odebráním jakékoli hrany se stane nesouvislým
- G neobsahuje kružnici, po přidání libovolné hrany vznikne v G kružnice

Strom

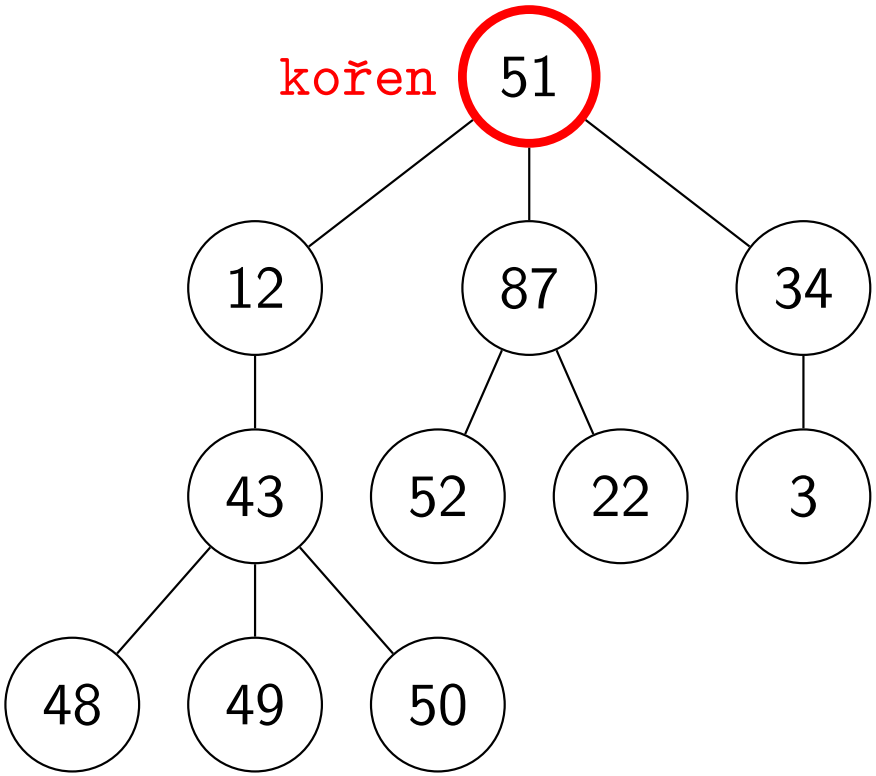


- $G = (V, E)$ je strom
- Mezi každou dvojicí vrcholů u, v existuje pouze jedna u-v cesta
- G je souvislý a odebráním jakékoli hrany se stane nesouvislým
- G neobsahuje kružnici, po přidání libovolné hrany vznikne v G kružnice
- G je souvislý a $|E| = |V| - 1$

Zakořeněný strom

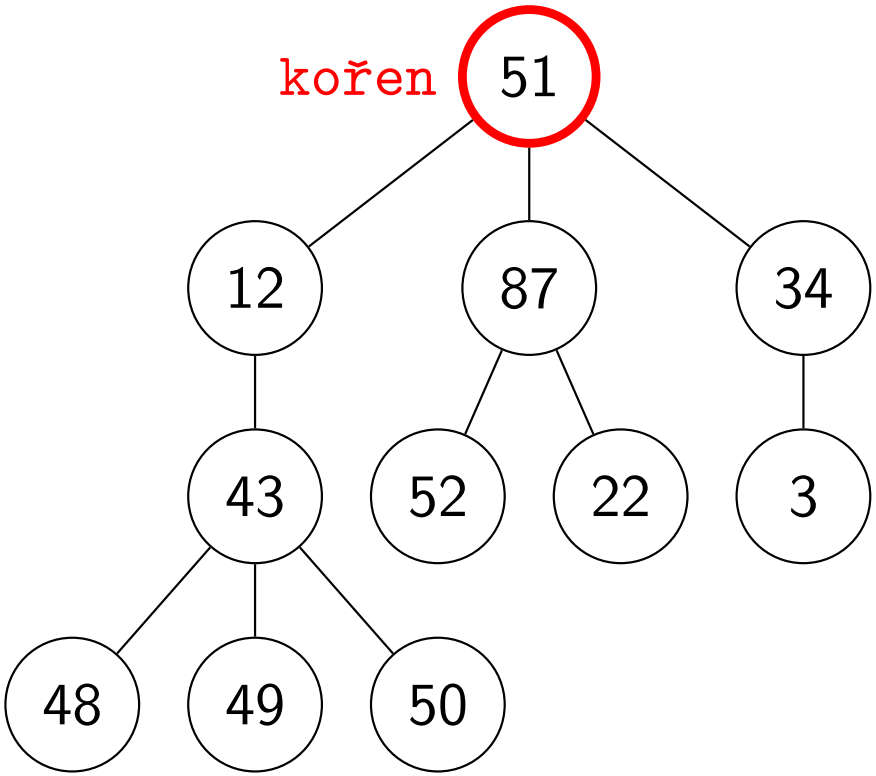


Zakořeněný strom



- Jeden z vrcholů množiny V je označen jako kořen

Zakořeněný strom



- Jeden z vrcholů množiny V je označen jako kořen
- Potomek vrcholu u je každý vrchol v, do kterého vede hrana z u do v

Binární strom

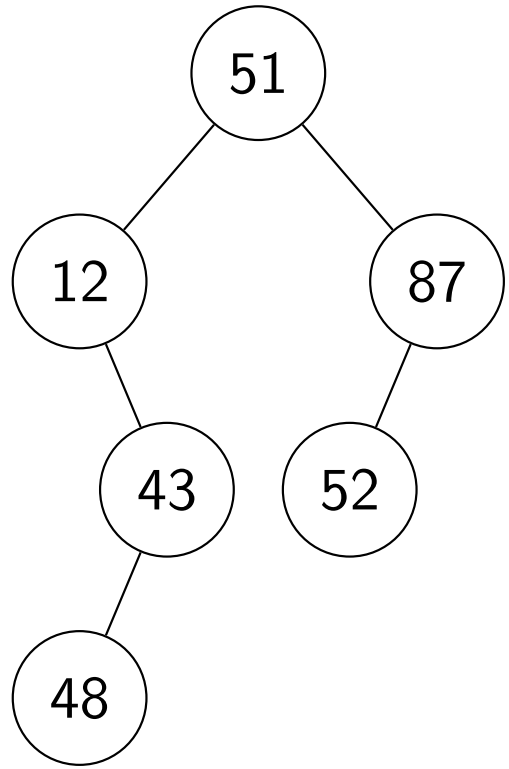
- je zakořeněný

Binární strom

- je zakořeněný
- každý vrchol má maximálně dva syny

Binární strom

- je zakořeněný
- každý vrchol má maximálně dva syny
- rozlišujeme levého a pravého syna



Binární minimová halda - motivace

Slyšeli jste o ní někdy?

Binární minimová halda - motivace

Prioritní fronta

- Hodí se datová struktura fronta nebo zásobník na plánování operačního sálu?

Binární minimová halda - motivace

Prioritní fronta

- Hodí se datová struktura fronta nebo zásobník na plánování operačního sálu?
- Není výhodné odbavit velkou zakázku na e-shopu dříve, než malou?

Binární minimová halda - motivace

Prioritní fronta

- Hodí se datová struktura fronta nebo zásobník na plánování operačního sálu?
- Není výhodné odbavit velkou zakázku na e-shopu dříve, než malou?
- Mezi dlážděnou a bahnitou cestou, jaká je lepší volba?

Binární minimová halda - motivace

Prioritní fronta

- Hodí se datová struktura fronta nebo zásobník na plánování operačního sálu?
- Není výhodné odbavit velkou zakázku na e-shopu dříve, než malou?
- Mezi dlážděnou a bahnitou cestou, jaká je lepší volba?
- Příchod mnoha rozdílně starých lidí k jednomu volnému sedadlu. Kdo má přednost?

Binární minimová halda - motivace

Prioritní fronta

- Hodí se datová struktura fronta nebo zásobník na plánování operačního sálu?
- Není výhodné odbavit velkou zakázku na e-shopu dříve, než malou?
- Mezi dlážděnou a bahnitou cestou, jaká je lepší volba?
- Příchod mnoha rozdílně starých lidí k jednomu volnému sedadlu. Kdo má přednost?
- Má real-time komunikace přednost při routování?

Tvar haldy

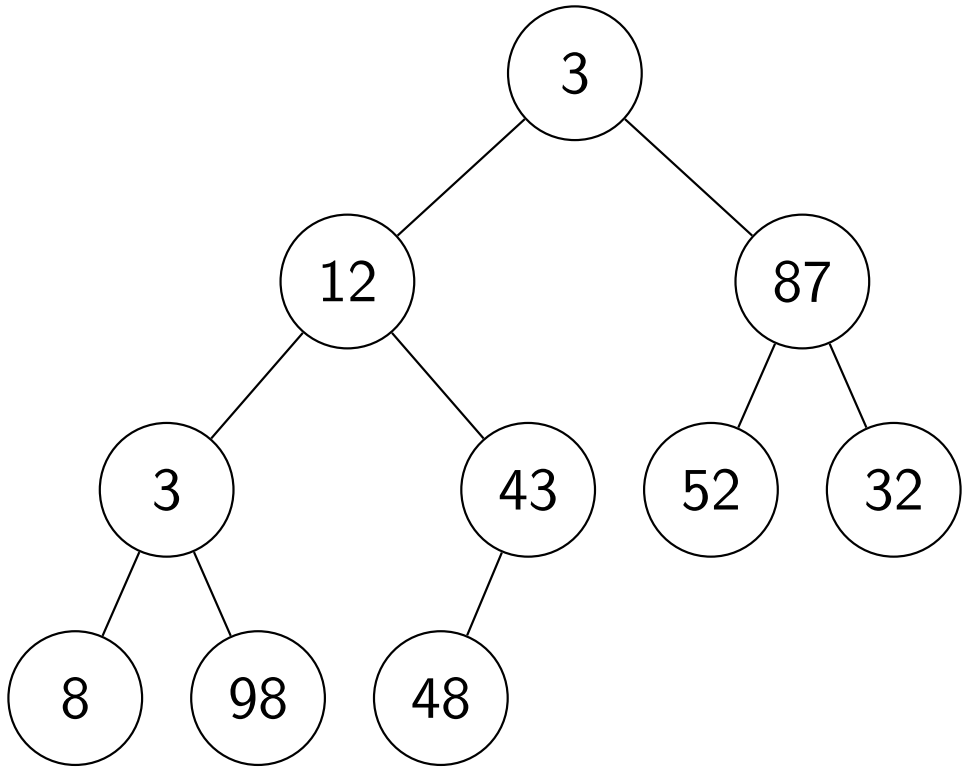
- Strom má všechny hladiny kromě poslední plně obsazené.

Tvar haldy

- Strom má všechny hladiny kromě poslední plně obsazené.
- Poslední hladina je zaplněna od levého okraje směrem k pravému

Tvar haldy

- Strom má všechny hladiny kromě poslední plně obsazené.
- Poslední hladina je zaplněna od levého okraje směrem k pravému



Haldové uspořádání

- Každý vrchol obsahuje hodnotu $key(v)$, kterou lze porovnávat $(\leq, \geq)$

Haldové uspořádání

- Každý vrchol obsahuje hodnotu $key(v)$, kterou lze porovnávat $(\leq, \geq)$
- Strom má všechny hladiny kromě poslední plně obsazené

Haldové uspořádání

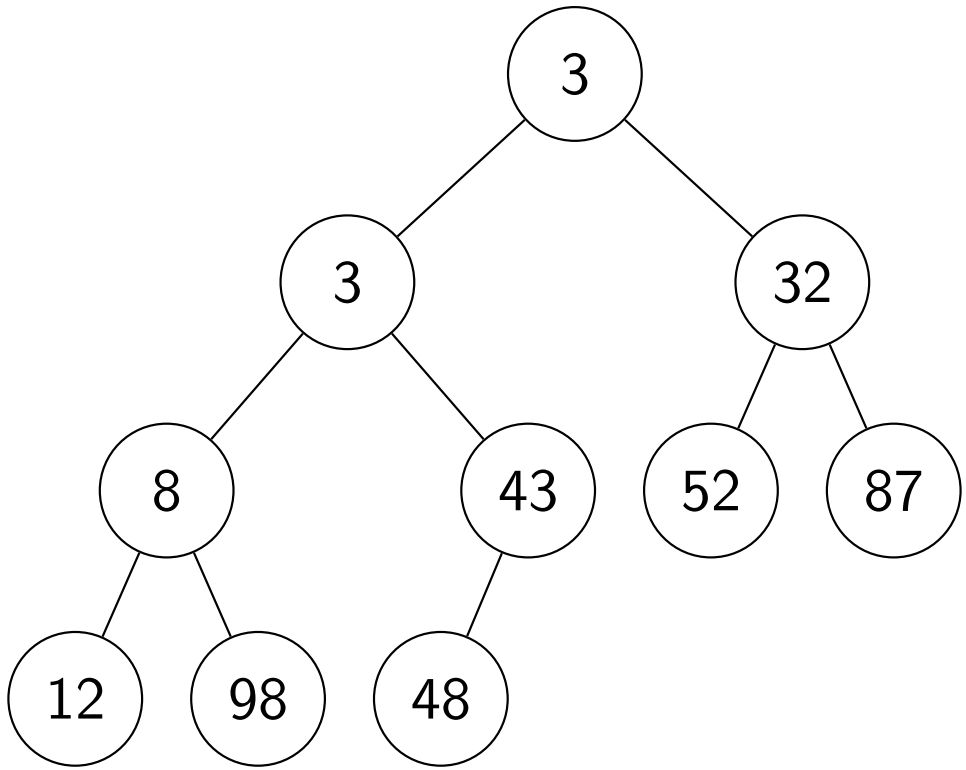
- Každý vrchol obsahuje hodnotu $key(v)$, kterou lze porovnávat $(\leq, \geq)$
- Strom má všechny hladiny kromě poslední plně obsazené
- Je-li v vrchol a s jeho syn, platí $key(v) \leq key(s)$

Haldové uspořádání

- Každý vrchol obsahuje hodnotu $key(v)$, kterou lze porovnávat $(\leq, \geq)$
- Strom má všechny hladiny kromě poslední plně obsazené
- Je-li v vrchol a s jeho syn, platí $key(v) \leq key(s)$
- Pro maximovou haldu: $key(v) \geq key(s)$

Haldové uspořádání

- Každý vrchol obsahuje hodnotu $key(v)$, kterou lze porovnávat $(\leq, \geq)$
- Strom má všechny hladiny kromě poslední plně obsazené
- Je-li v vrchol a s jeho syn, platí $key(v) \leq key(s)$
- Pro maximovou haldu: $key(v) \geq key(s)$



Haldové uspořádání - nejednoznačnost

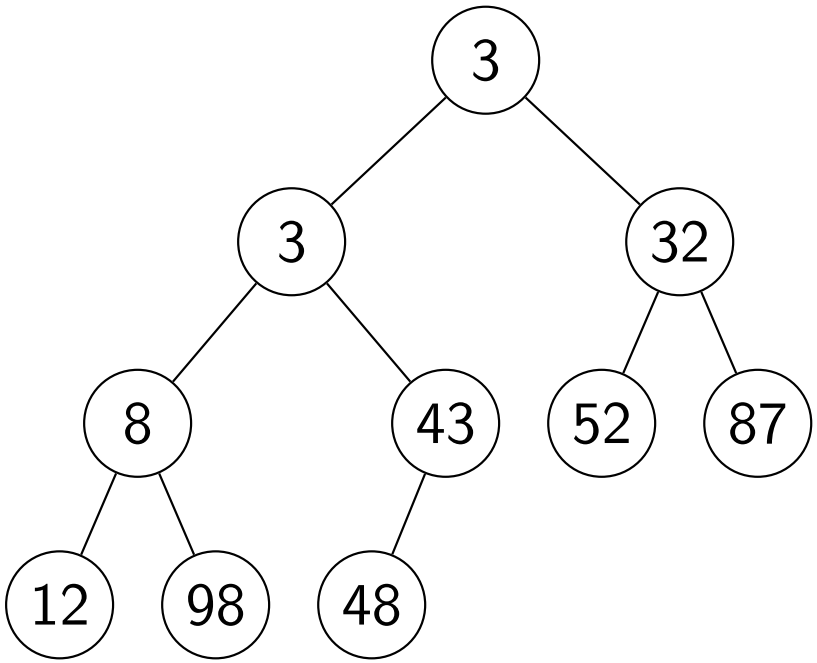
- Množina vrcholů nemá pevně dané haldové uspořádání

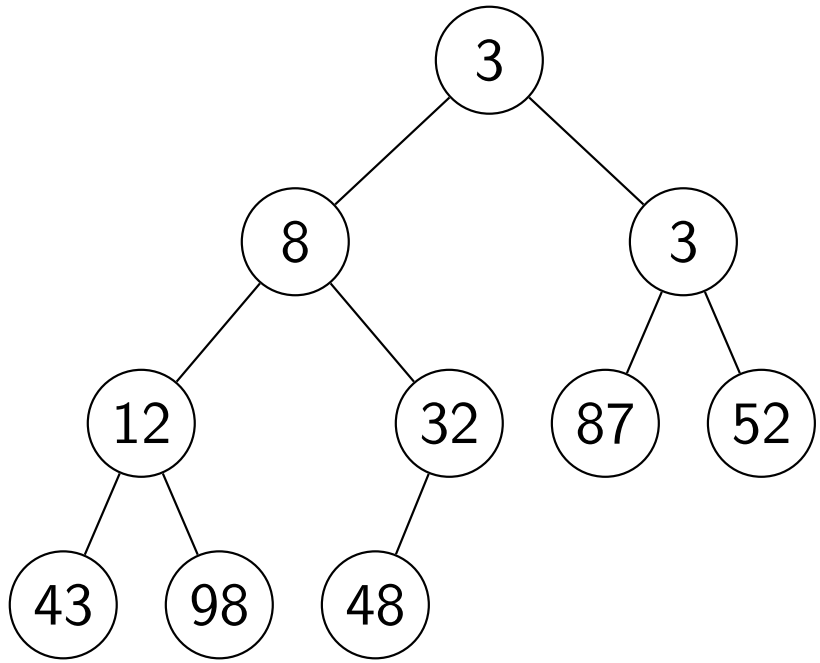
Haldové uspořádání - nejednoznačnost

- Množina vrcholů nemá pevně dané haldové uspořádání
- Také by předchozí příklad mohl vypadat takto:

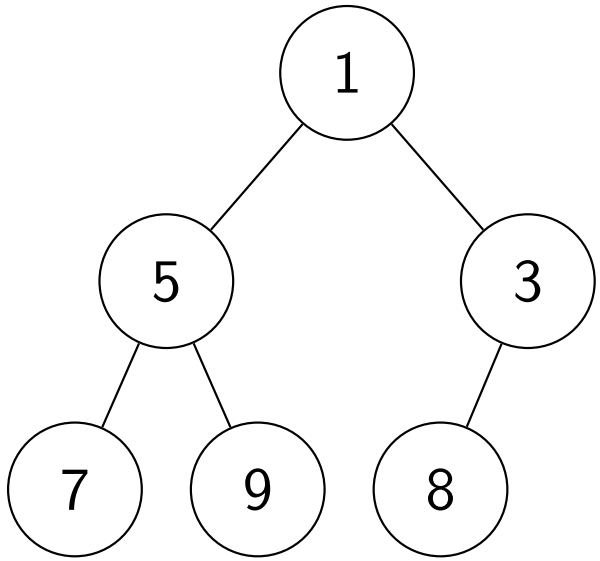
Haldové uspořádání - nejednoznačnost

- Množina vrcholů nemá pevně dané haldové uspořádání
- Také by předchozí příklad mohl vypadat takto:

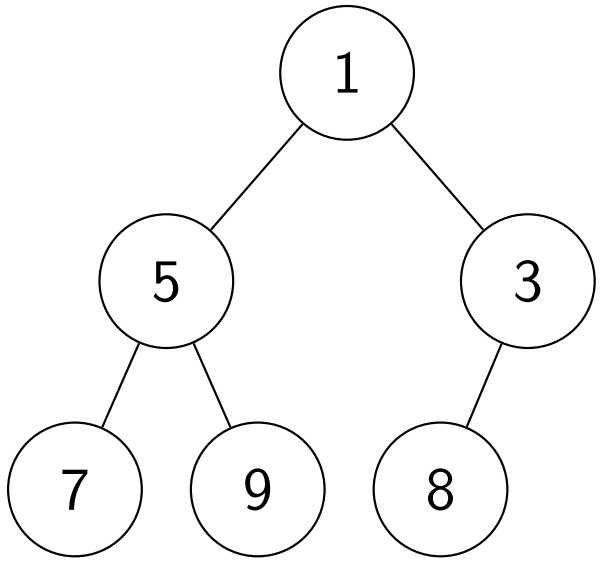




Implementace pomocí pole



Implementace pomocí pole



0	**1**	**2**	**3**	**4**	**5**
1	5	3	7	9	8

Implementace pomocí pole

+ Binární haldu je možné efektivně uložit do pole

Implementace pomocí pole

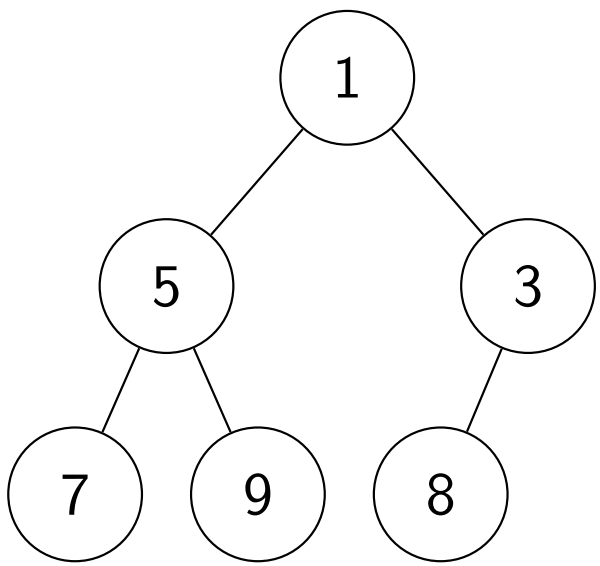
+ Binární haldu je možné efektivně uložit do pole
- Levý syn vrcholu na indexu i se nachází na indexu $2i + 1$

Implementace pomocí pole

+ Binární haldu je možné efektivně uložit do pole
- Levý syn vrcholu na indexu i se nachází na indexu $2i + 1$
- Levý syn vrcholu na indexu i se nachází na indexu $2i + 2$

Implementace pomocí pole

+ Binární haldu je možné efektivně uložit do pole
- Levý syn vrcholu na indexu i se nachází na indexu $2i + 1$
- Levý syn vrcholu na indexu i se nachází na indexu $2i + 2$
- Rodič vrcholu na indexu i se nachází na indexu $\lfloor \frac{i-1}{2} \rfloor$

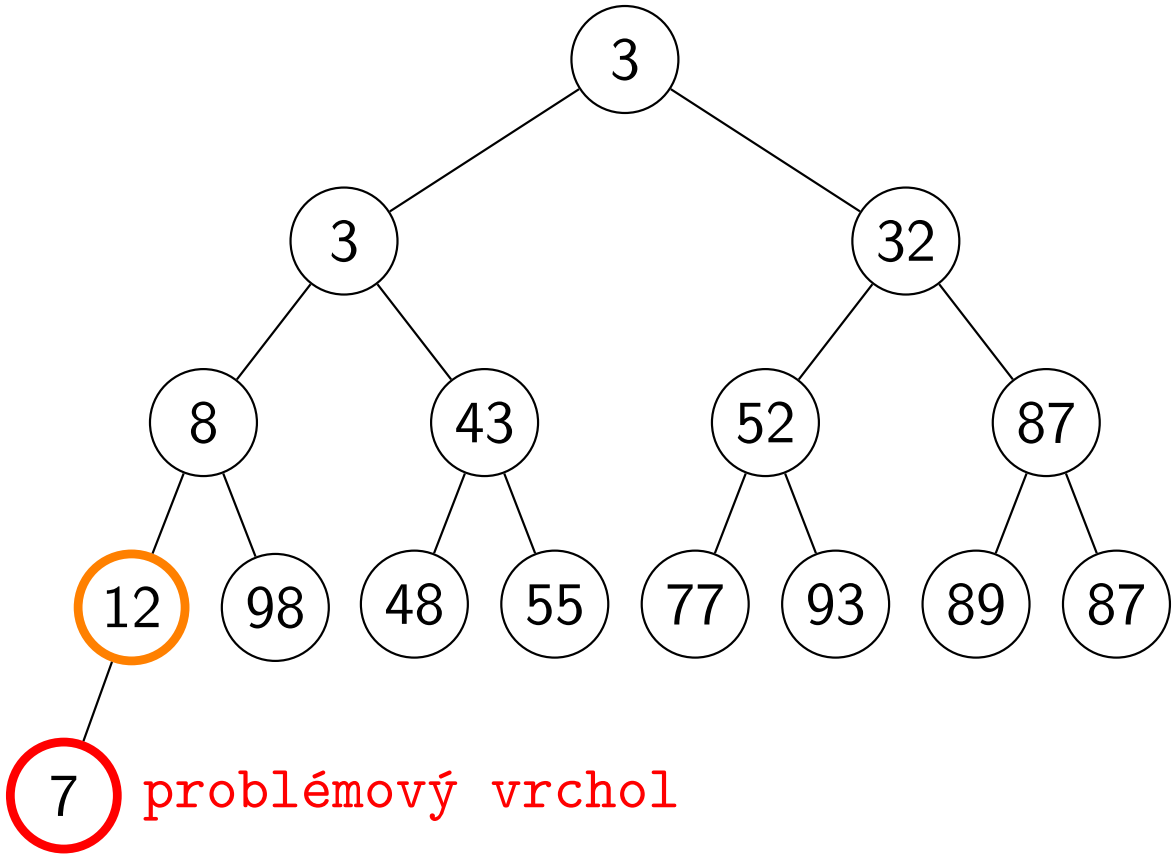


0	**1**	**2**	**3**	**4**	**5**
1	5	3	7	9	8

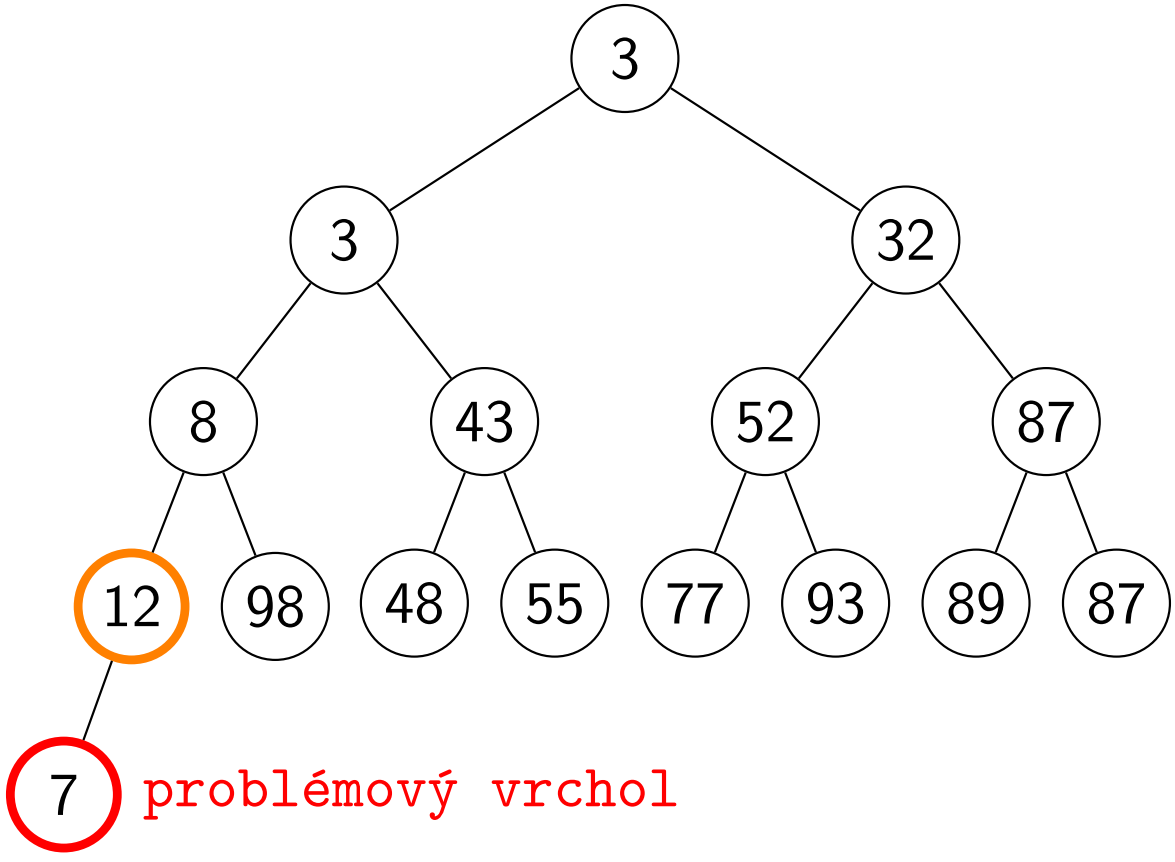
Probublání nahoru

```
void HeapBubbleUp(Heap heap, int index) {
  int parent;

  while (HeapHasParent(heap, index)) {
    parent = HeapParent(index);

    if (HeapElementValue(heap, parent) <= HeapElementValue(heap, index)) {
      break;
    }

    HeapSwap(heap, index, parent);

    index = parent;
  }
}
```

Probublání nahoru - příklad

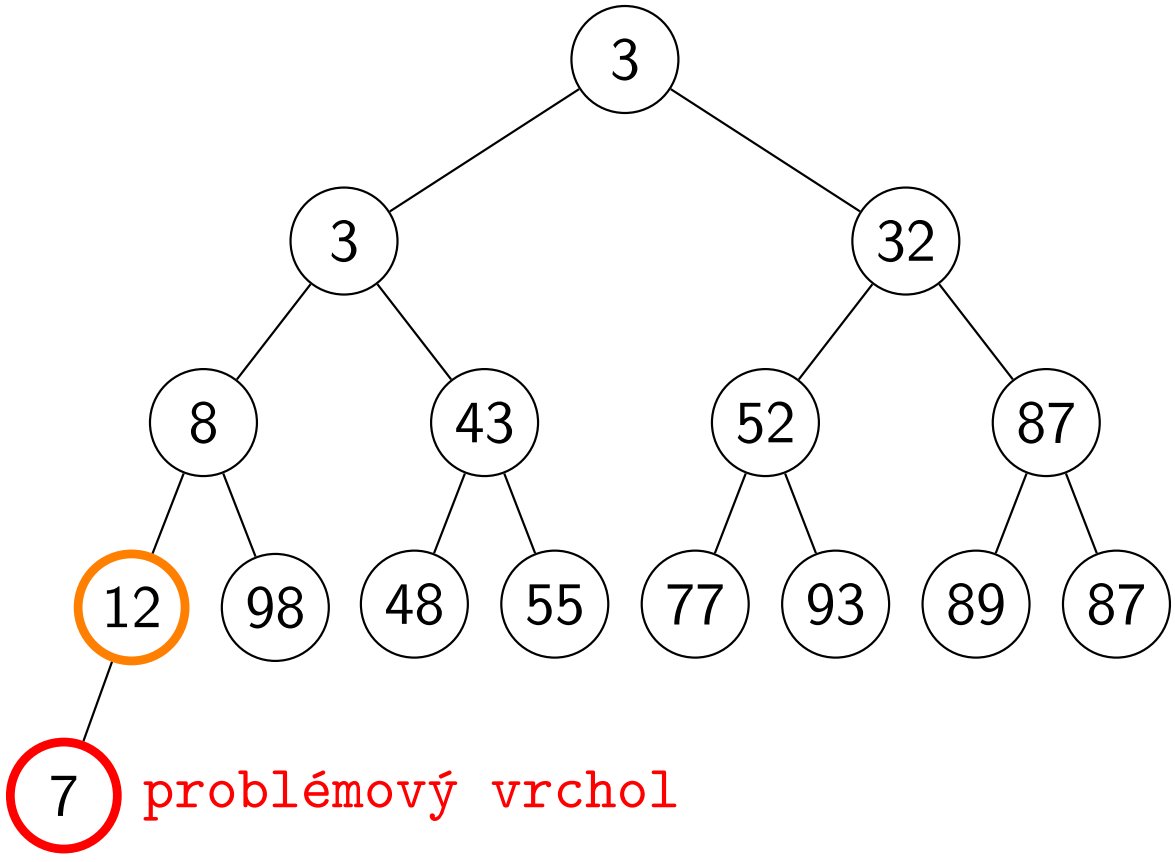


Probublání nahoru - příklad



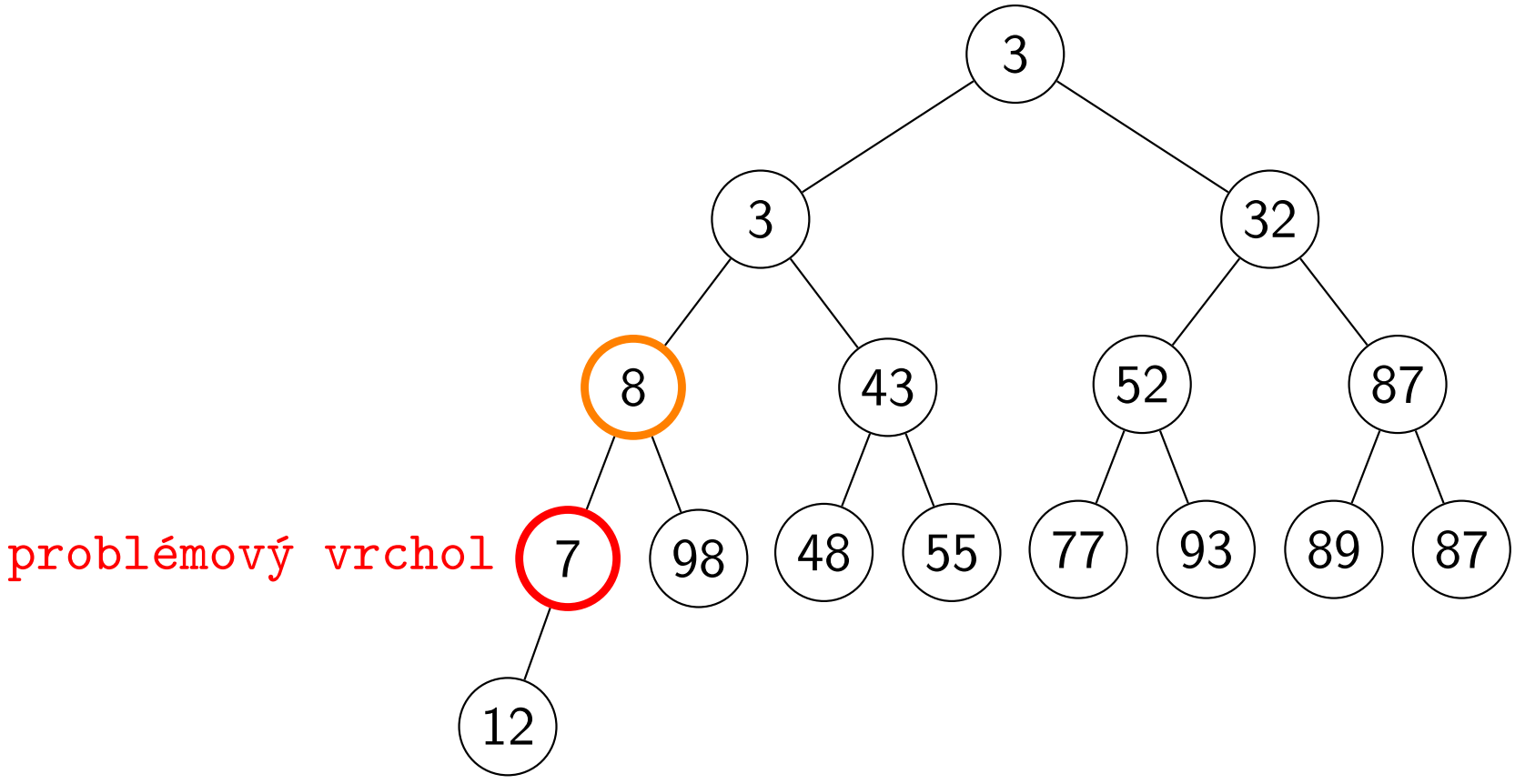
AKCE:

Probublání nahoru - příklad

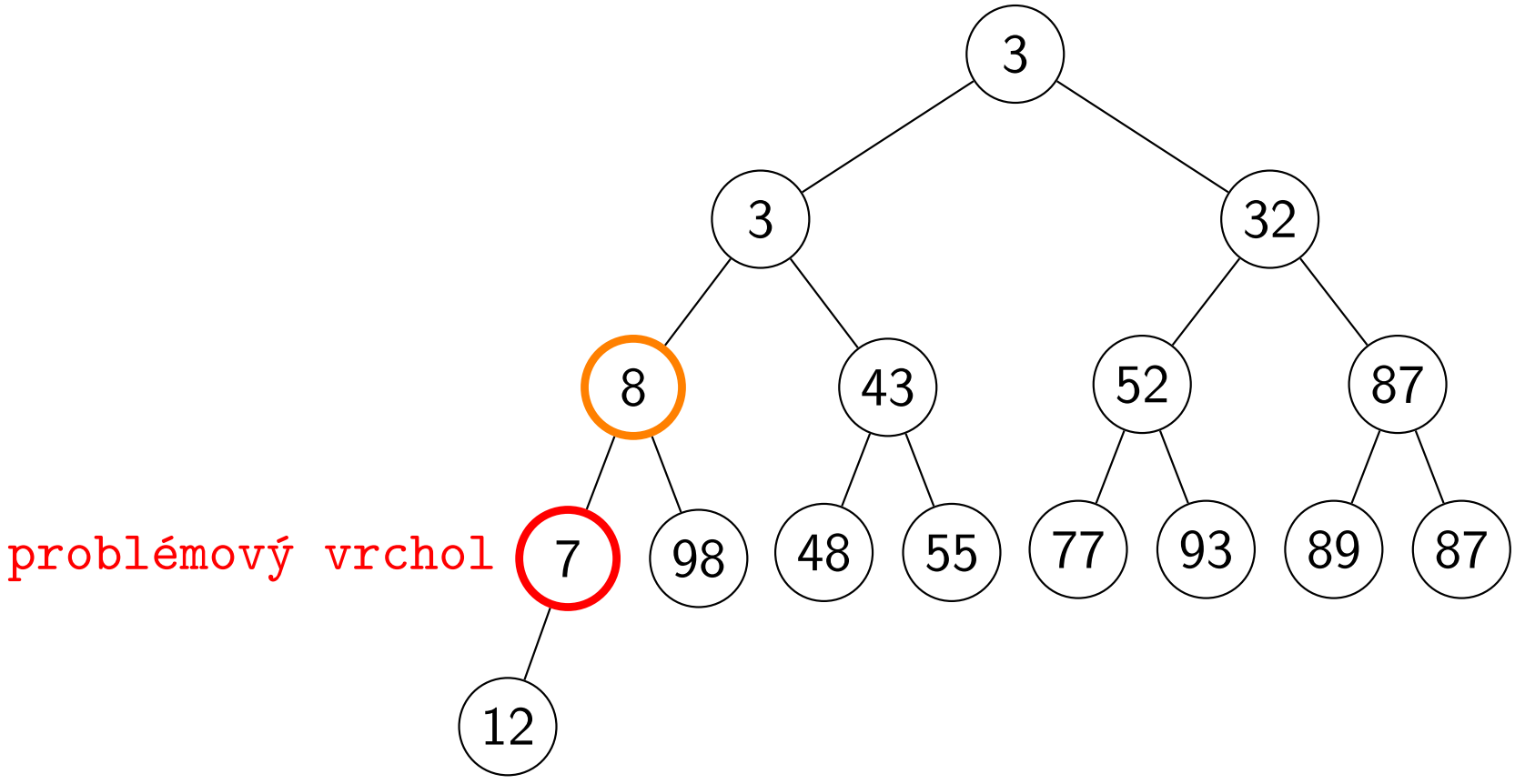


AKCE: probublej 7 nahoru

Probublání nahoru - příklad

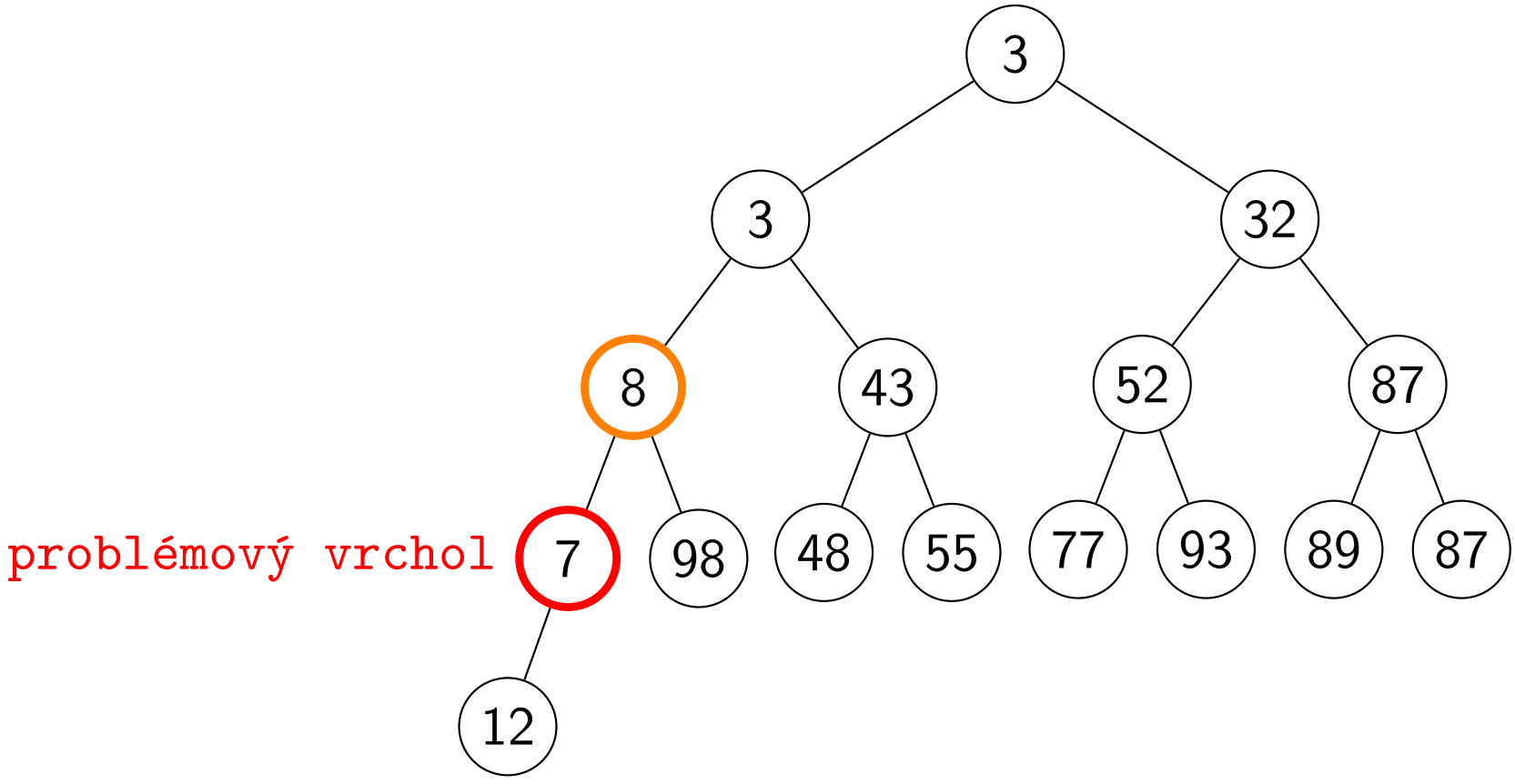


Probublání nahoru - příklad



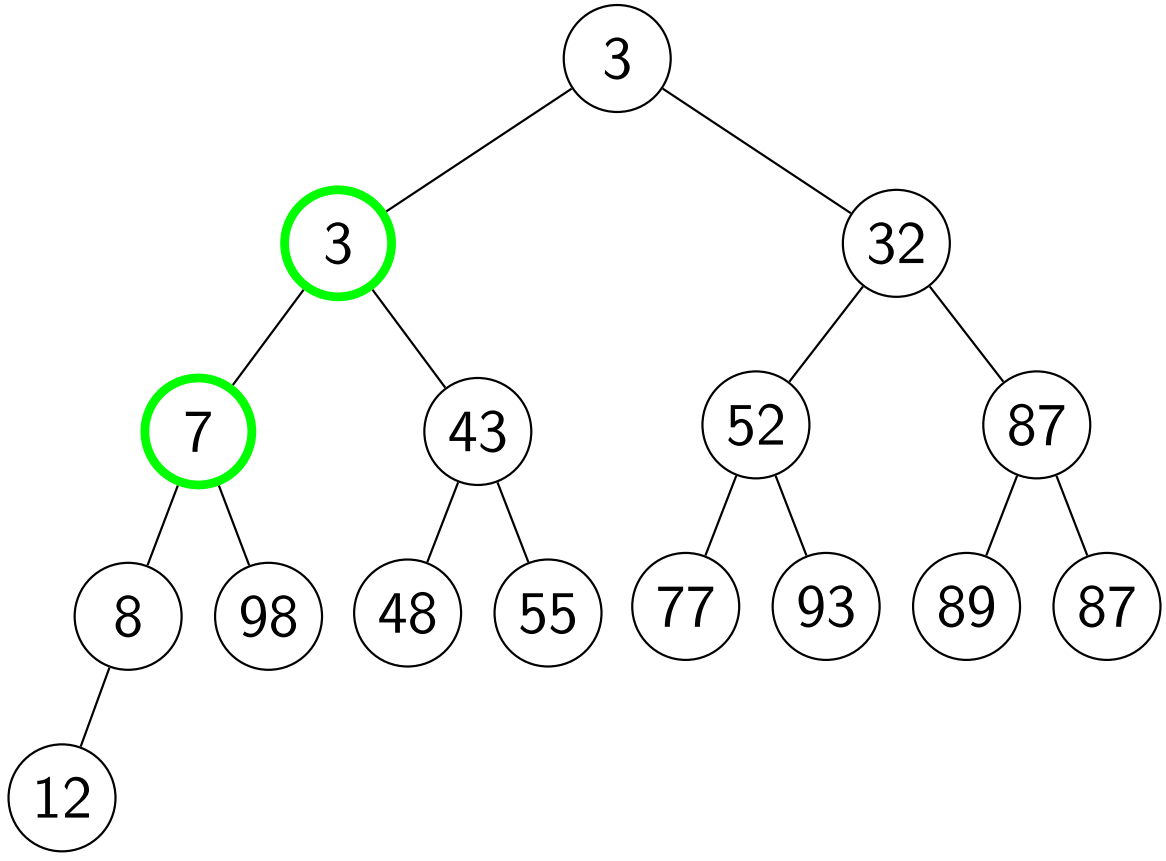
AKCE:

Probublání nahoru - příklad

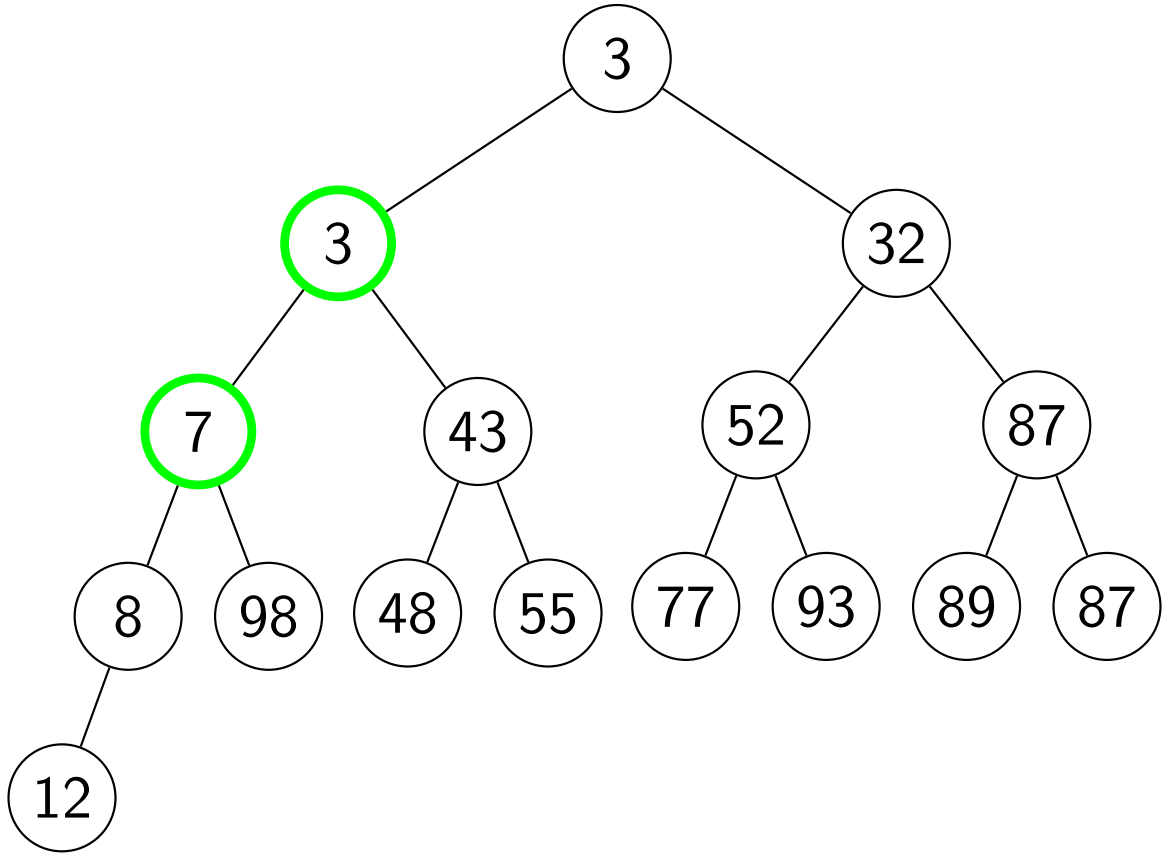


AKCE: probublej 7 nahoru

Probublání nahoru - příklad

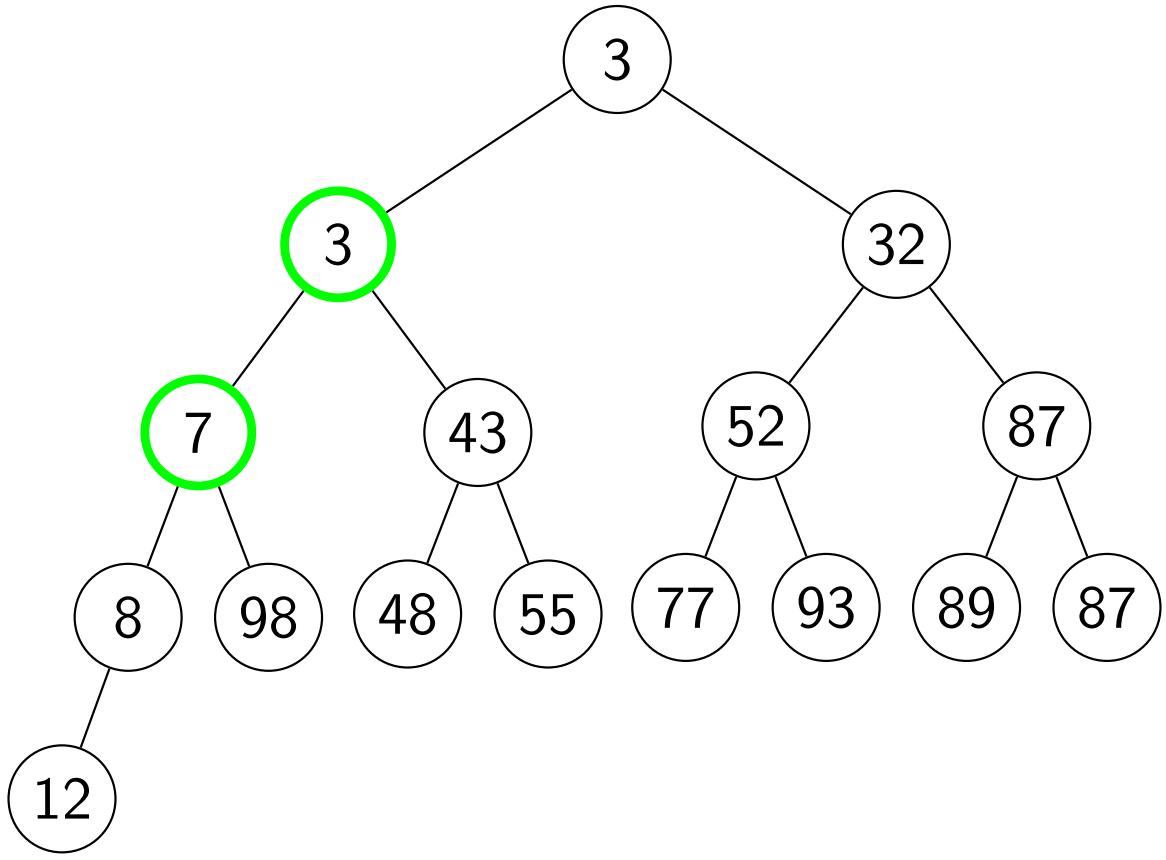


Probublání nahoru - příklad



AKCE:

Probublání nahoru - příklad

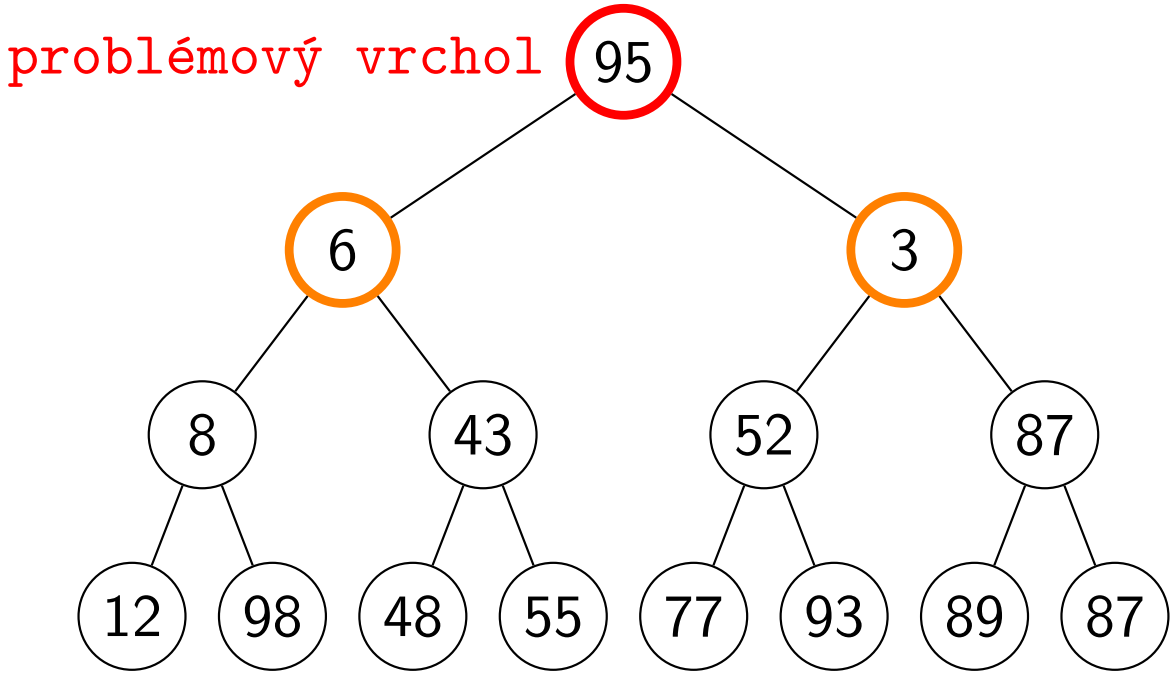


AKCE: stop

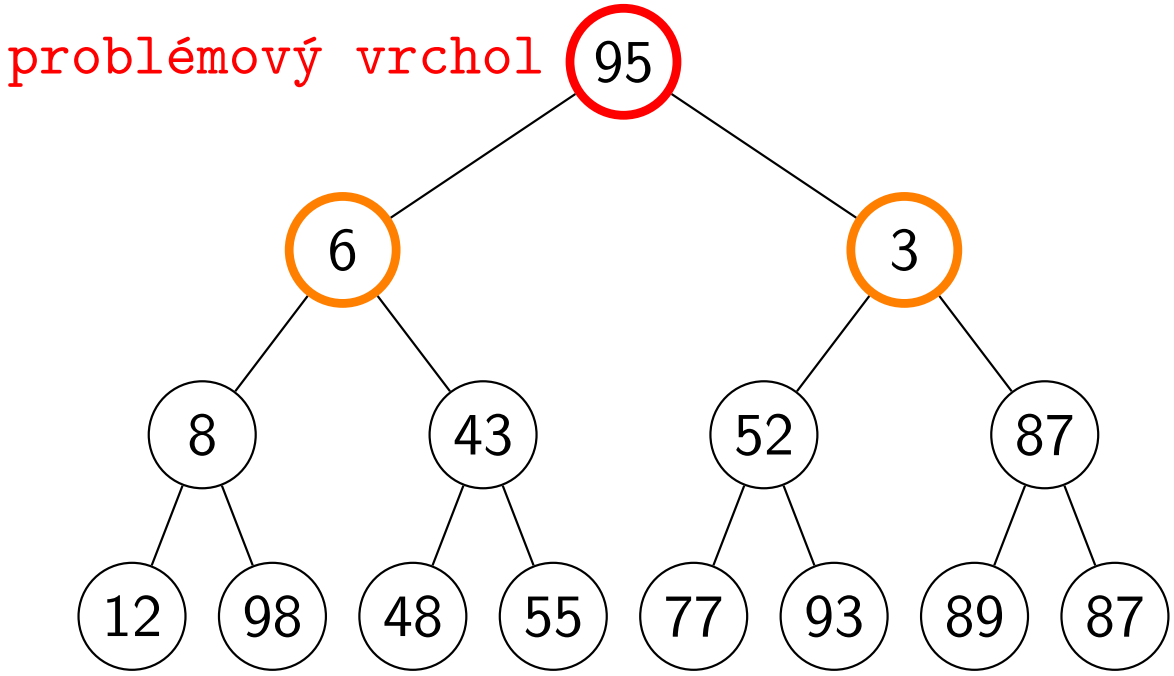
Probublání dolů

```
void HeapBubbleDown(Heap heap, int index) {
    int child;

    while (HeapHasChild(heap, index)) {
        child = HeapLowerChild(heap, index);

        if (HeapElementValue(heap, index) <= HeapElementValue(heap, child)) {
            break;
        }

        HeapSwap(heap, index, child);

        index = child;
    }
}
```

Probublání dolů - příklad

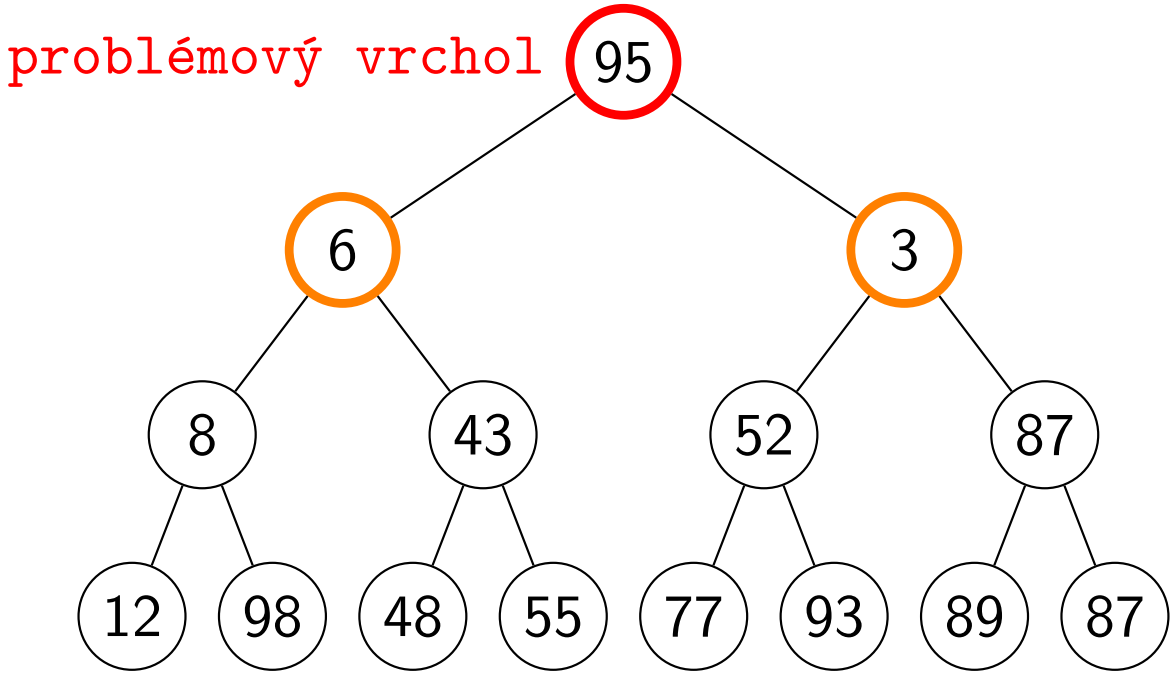


Probublání dolů - příklad



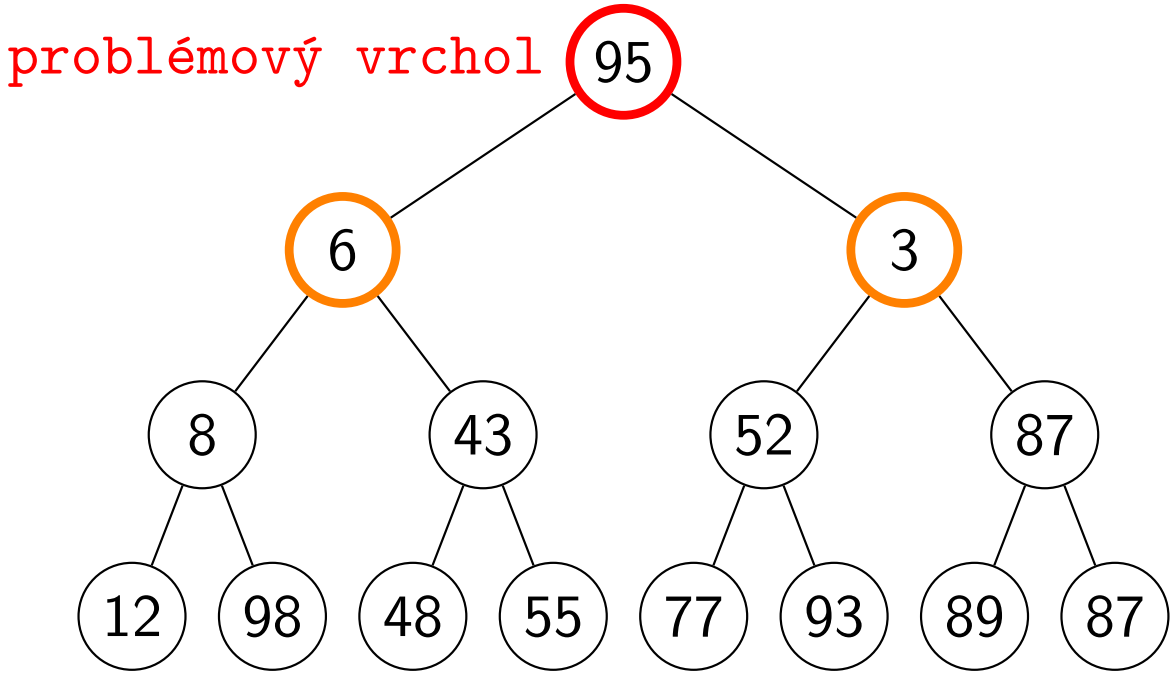
AKCE:

Probublání dolů - příklad



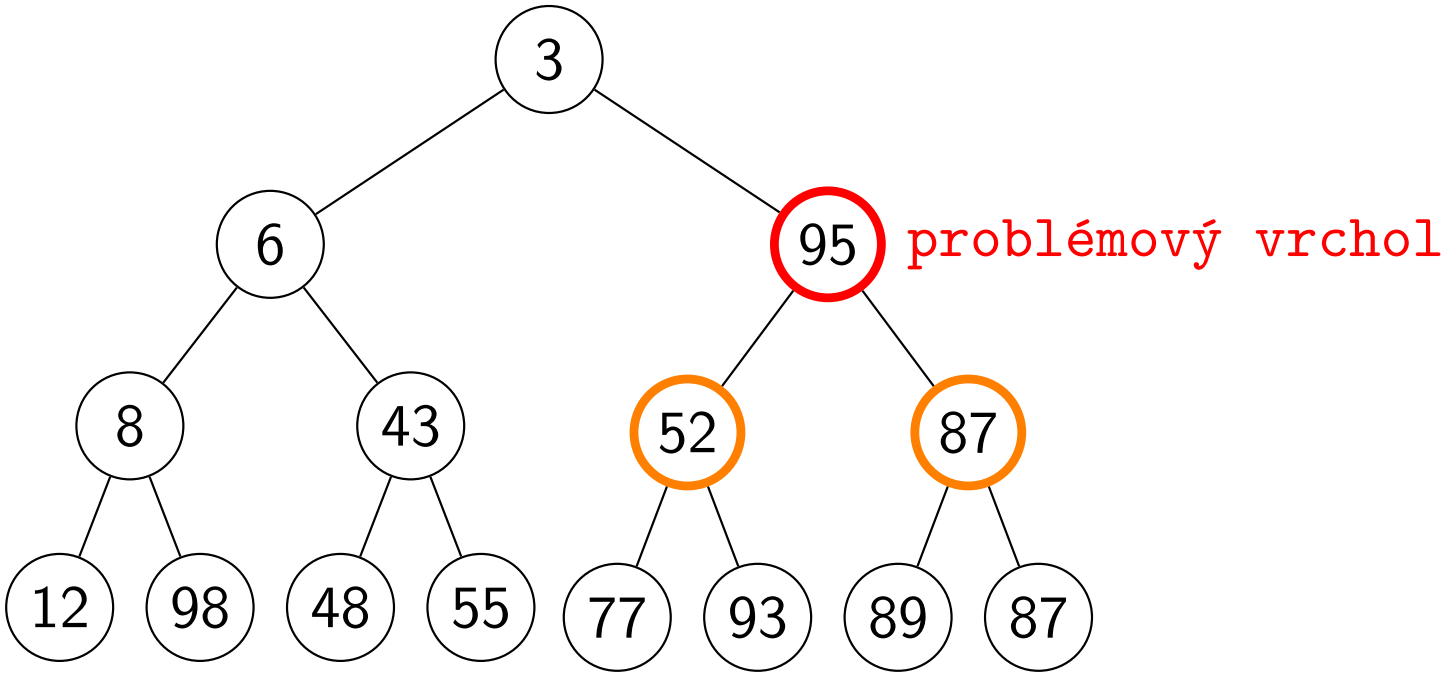
AKCE: probublej 95 dolů

Probublání dolů - příklad

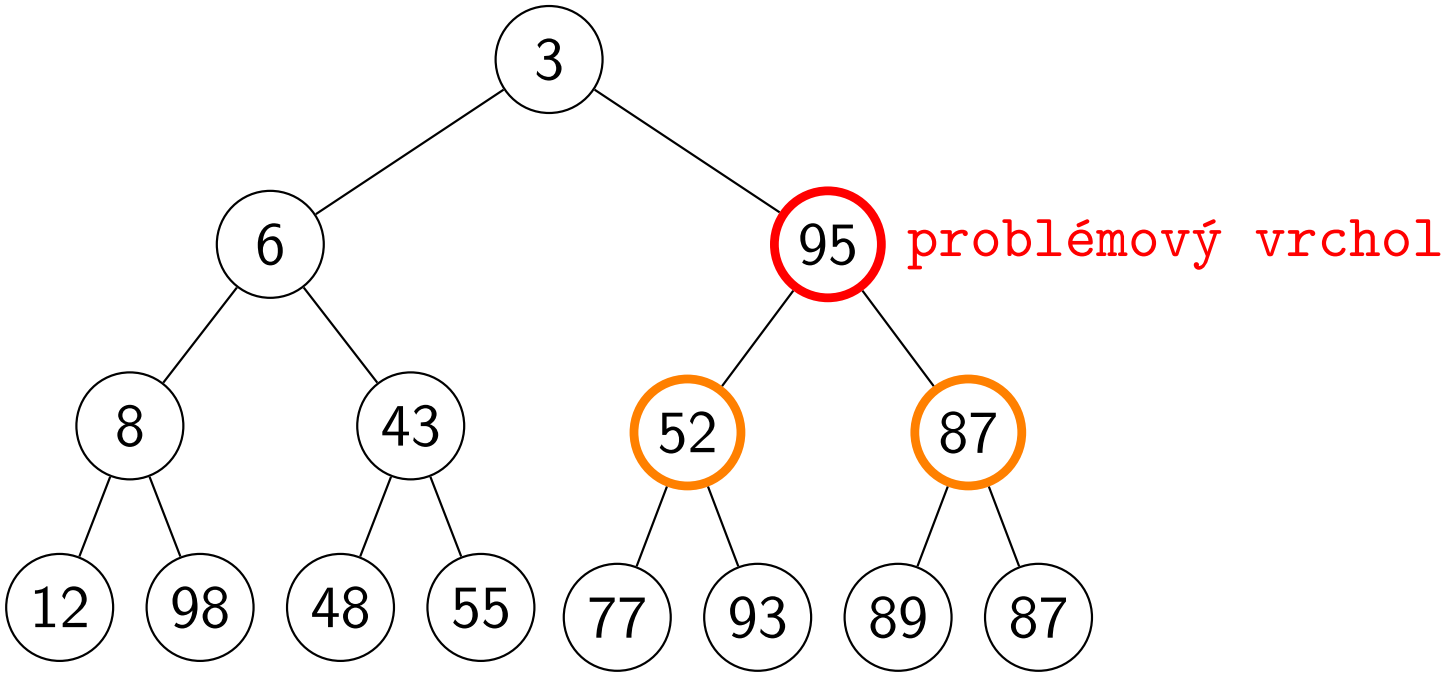


AKCE: probublej 95 dolů doprava

Probublání dolů - příklad

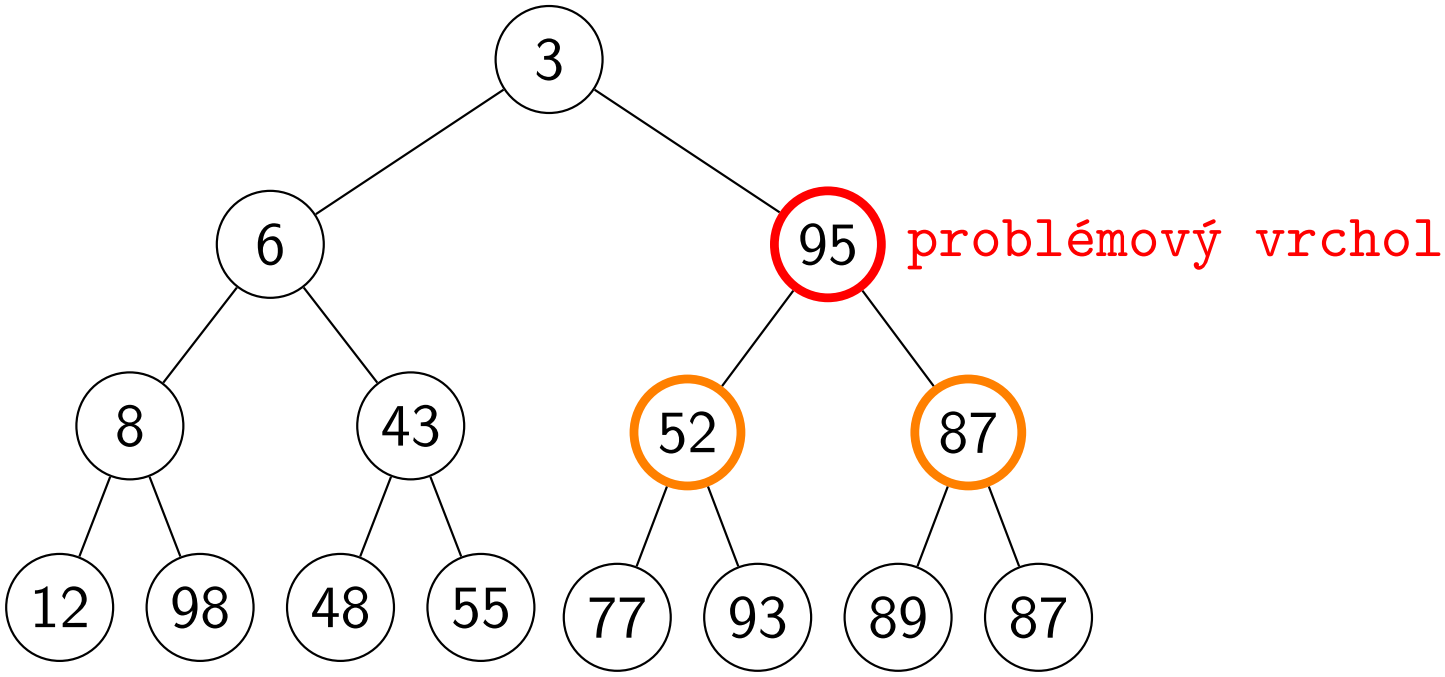


Probublání dolů - příklad



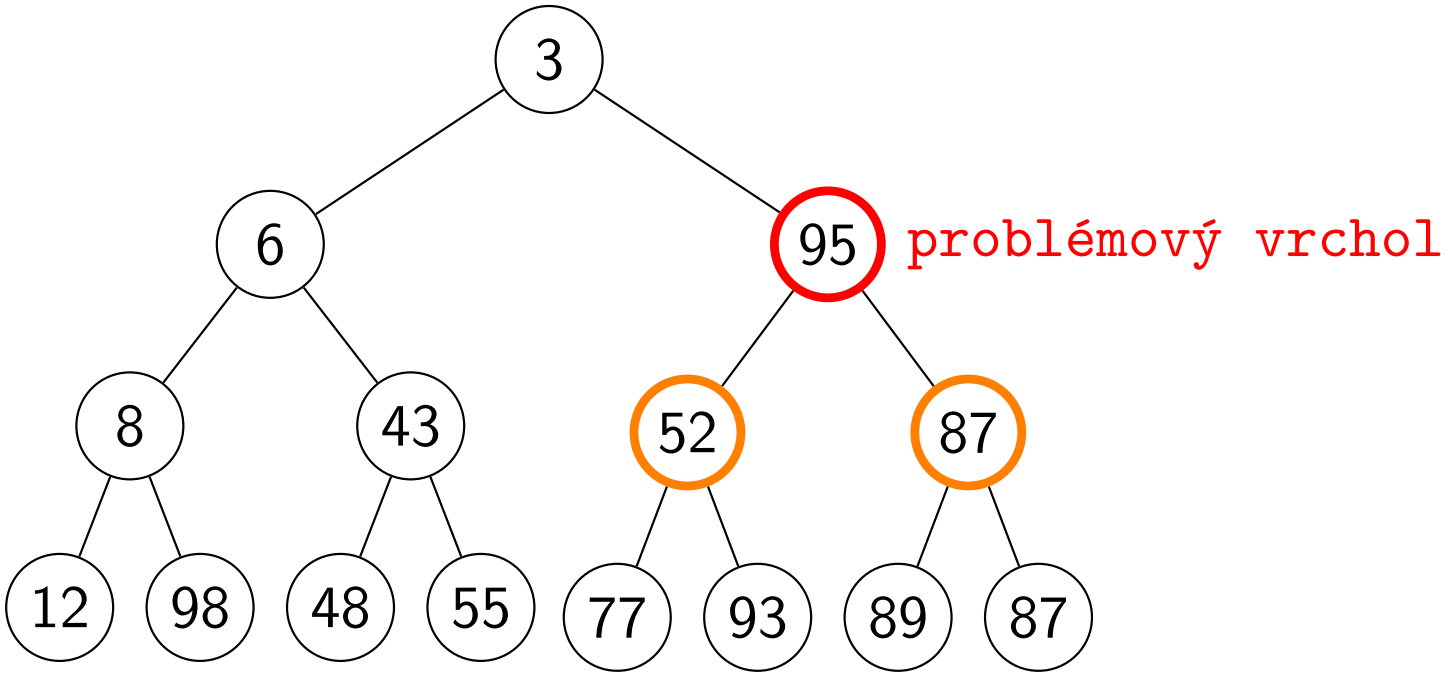
AKCE:

Probublání dolů - příklad



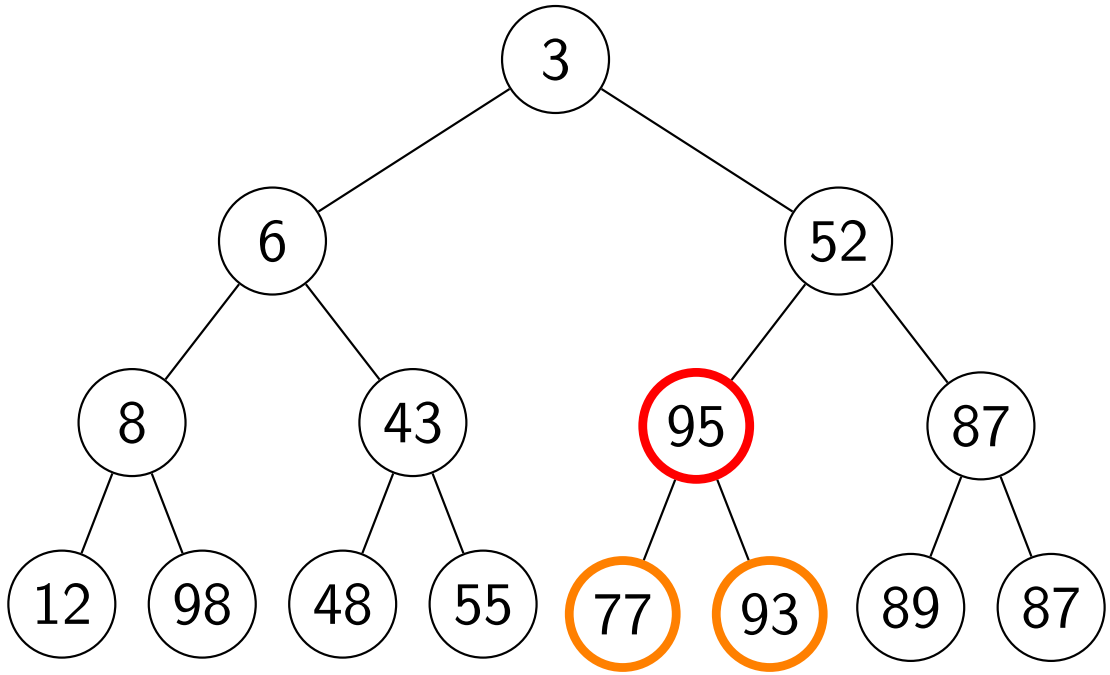
AKCE: probublej 95 dolů

Probublání dolů - příklad

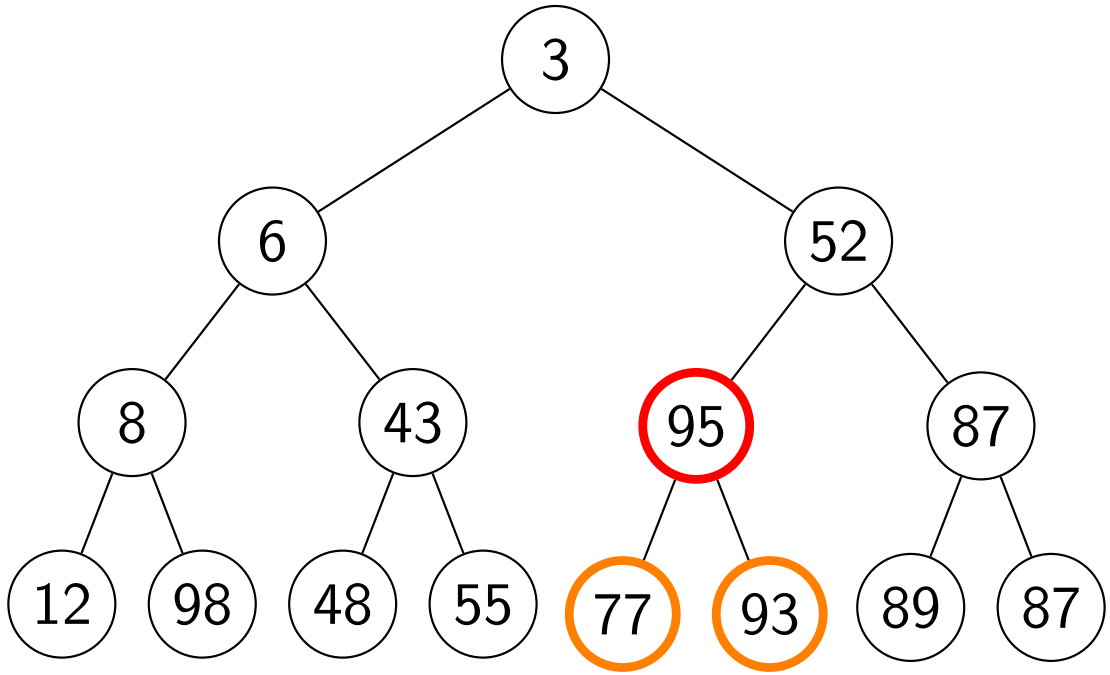


AKCE: probublej 95 dolů doleva

Probublání dolů - příklad

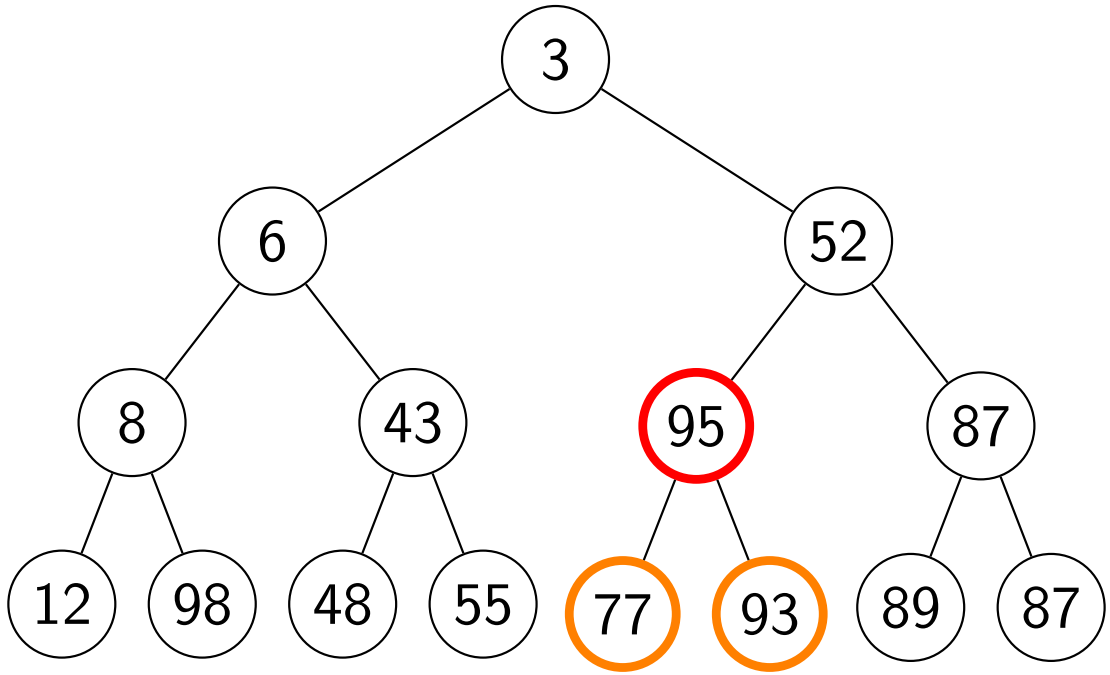


Probublání dolů - příklad



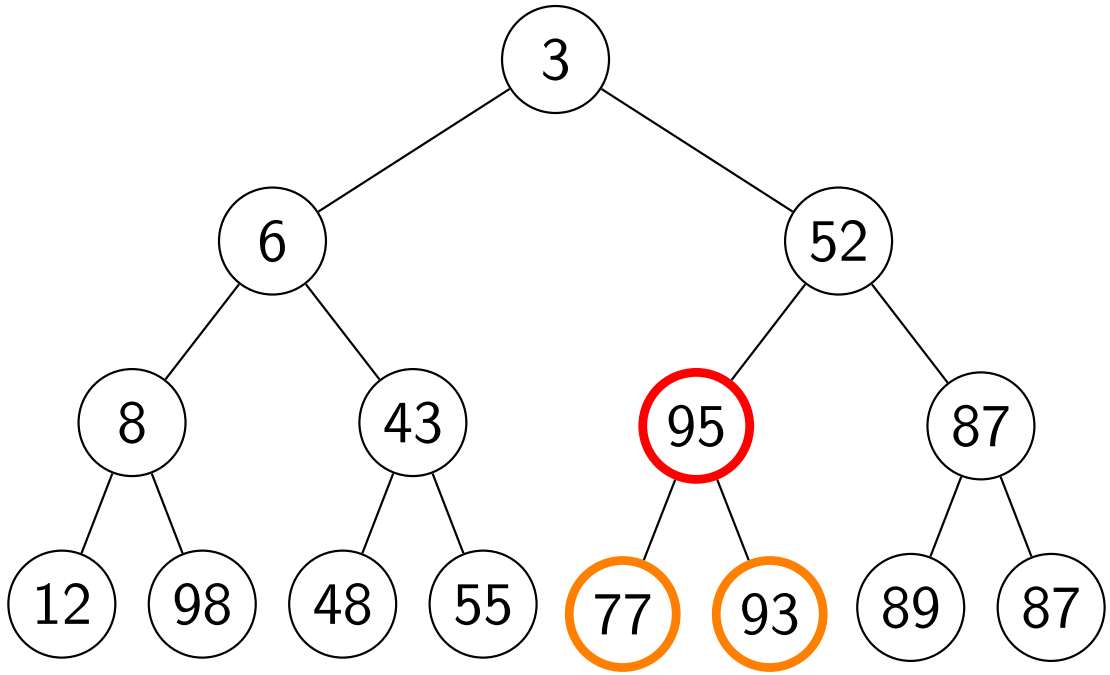
AKCE:

Probublání dolů - příklad



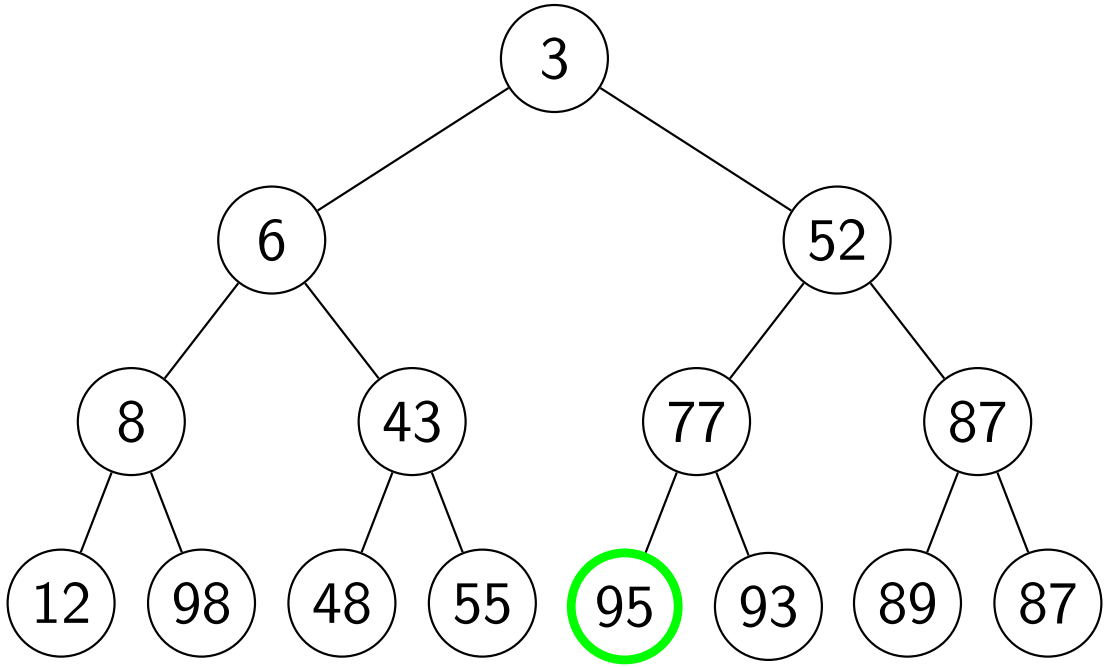
AKCE: probublej 95 dolů

Probublání dolů - příklad

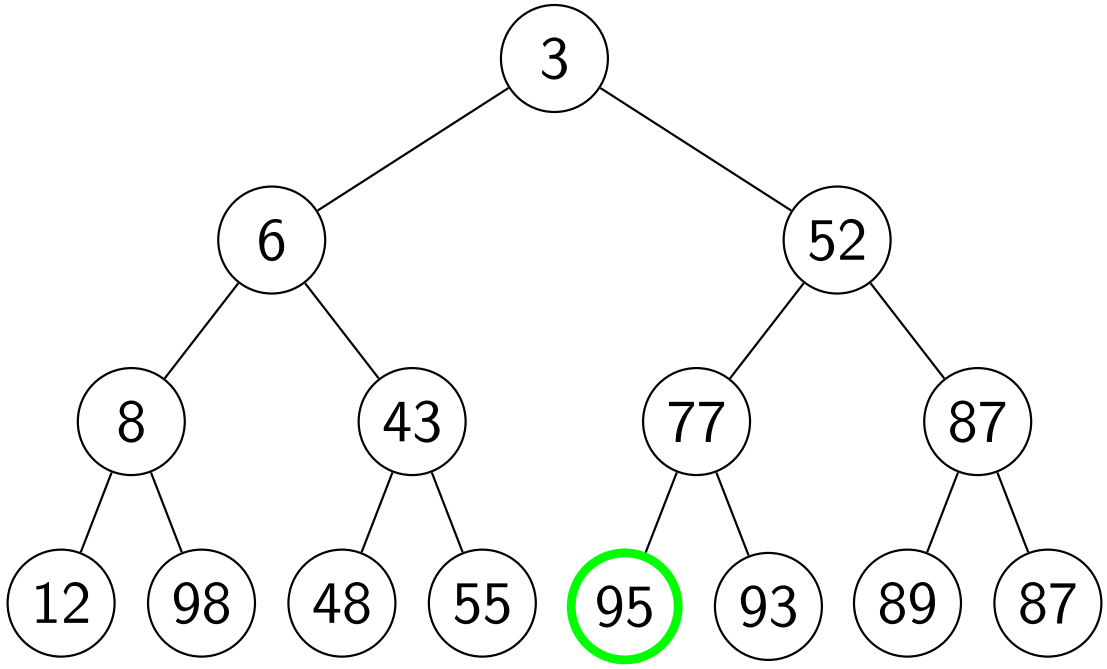


AKCE: probublej 95 dolů doleva

Probublání dolů - příklad

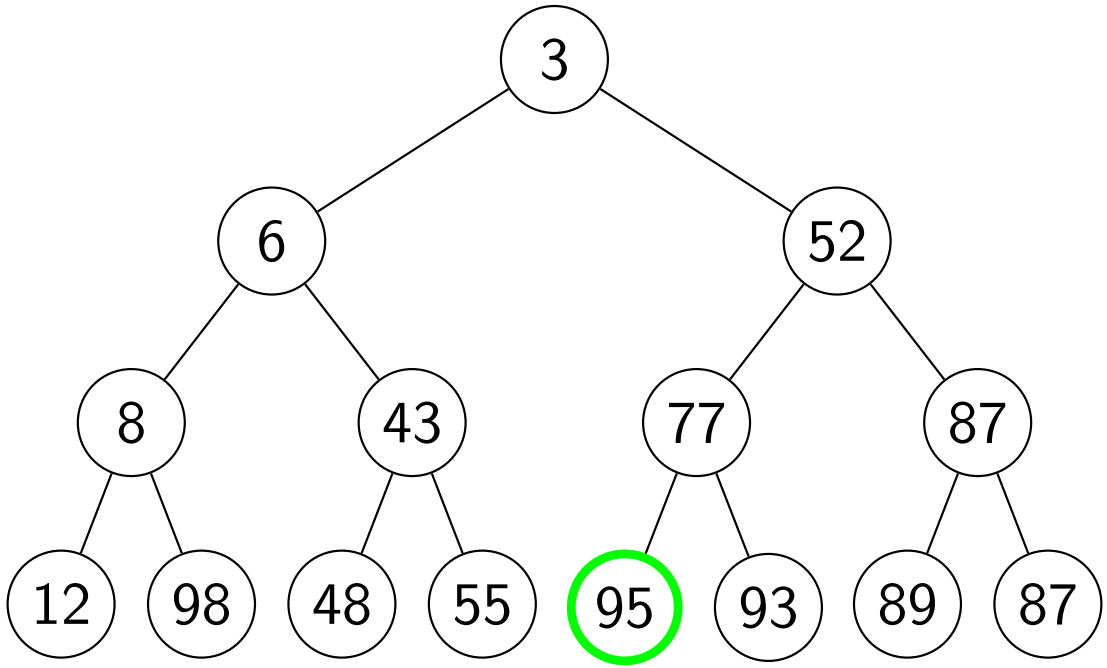


Probublání dolů - příklad



AKCE:

Probublání dolů - příklad

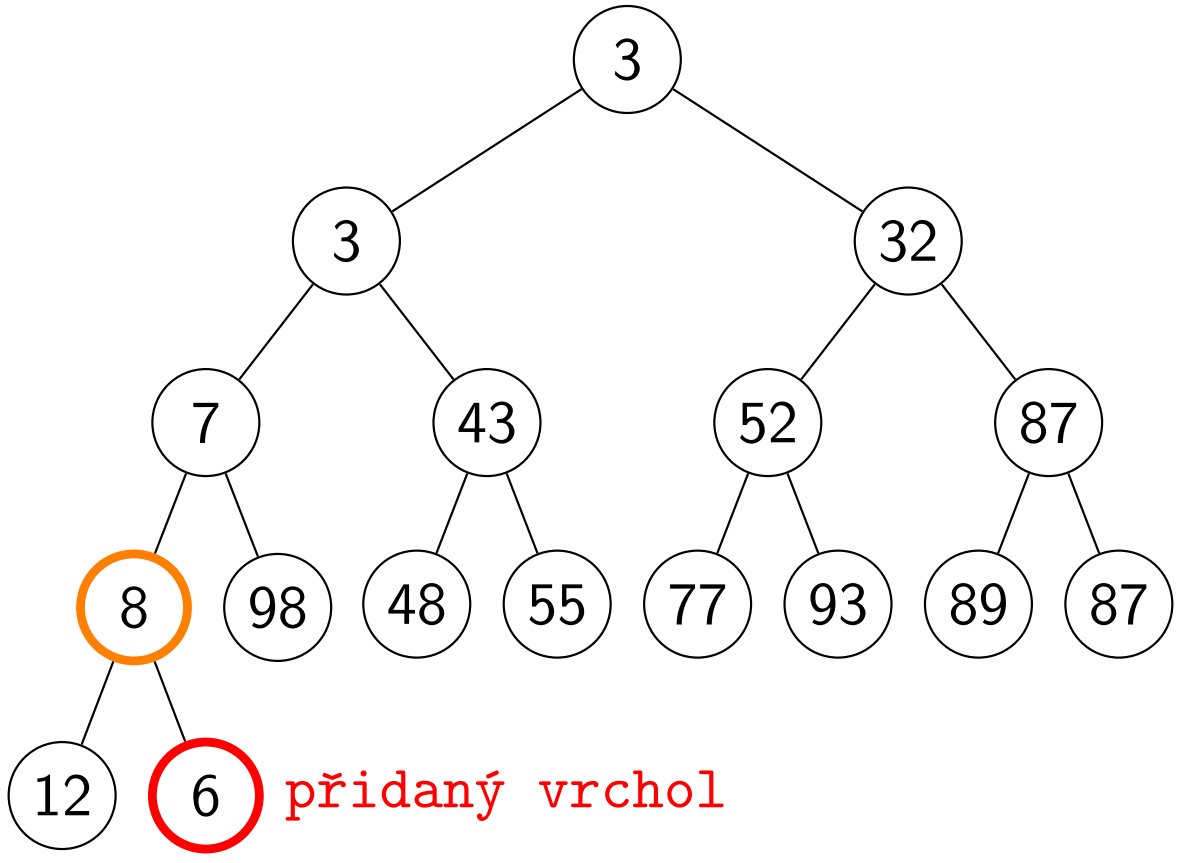


AKCE: stop

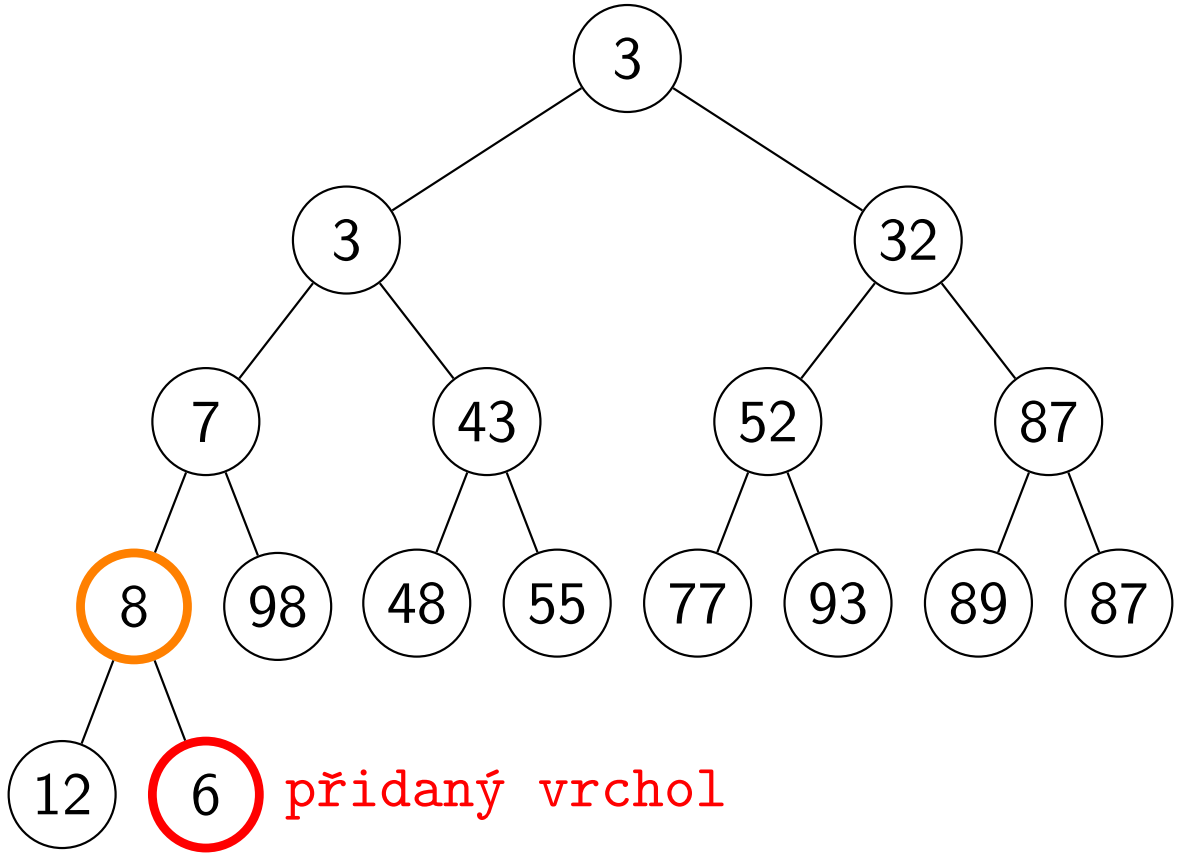
Vložení prvku

```
Heap HeapInsert(Heap heap, HeapElement element) {
  int index = heap.elementsCount;

  heap.elements[index] = element;
  heap.elementsCount++;

  HeapBubbleUp(heap, index);

  return heap;
}
```

Vložení prvku - příklad

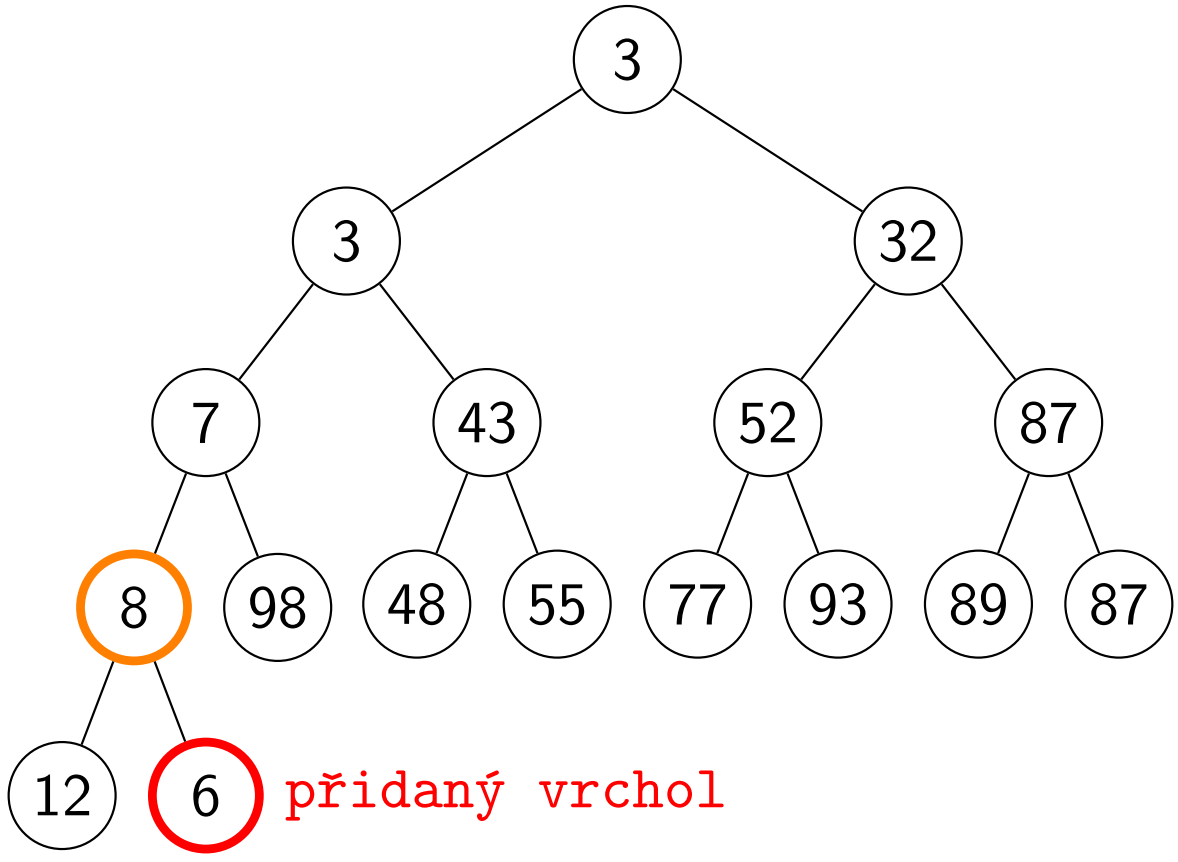


Vložení prvku - příklad



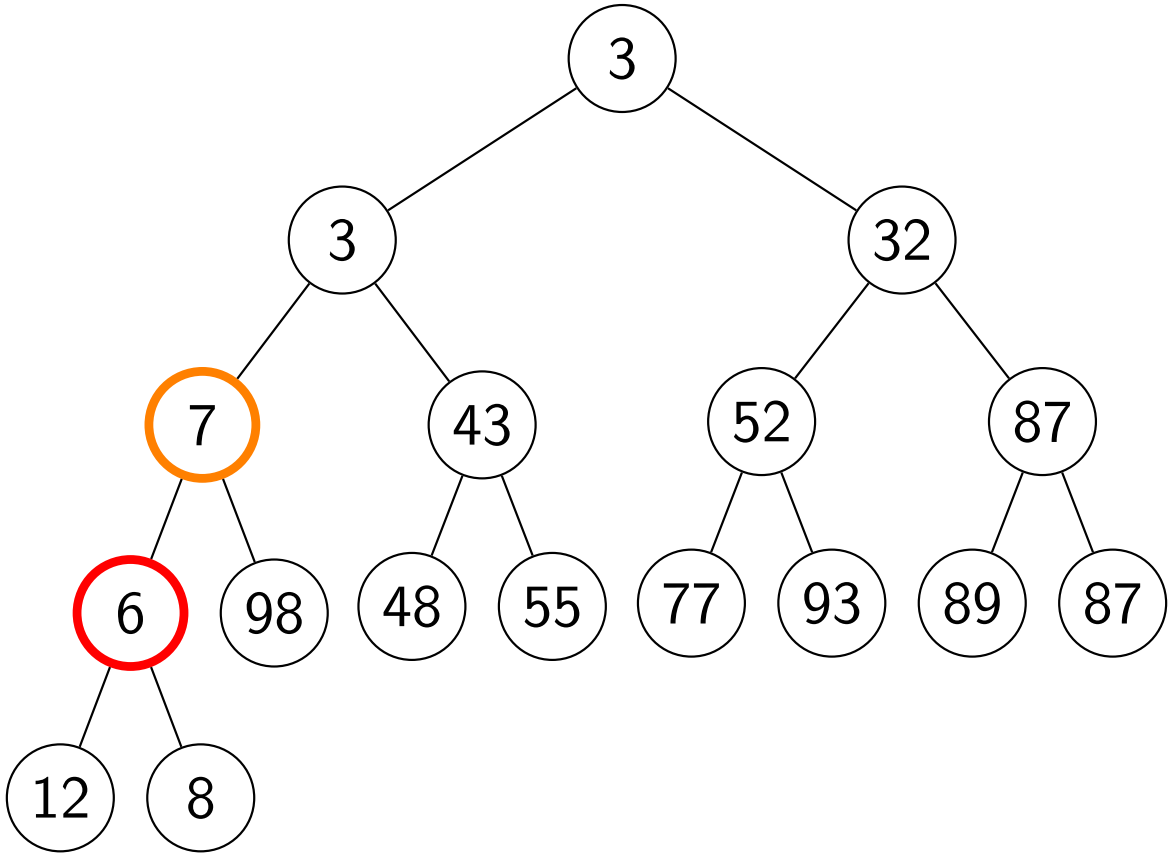
AKCE:

Vložení prvku - příklad

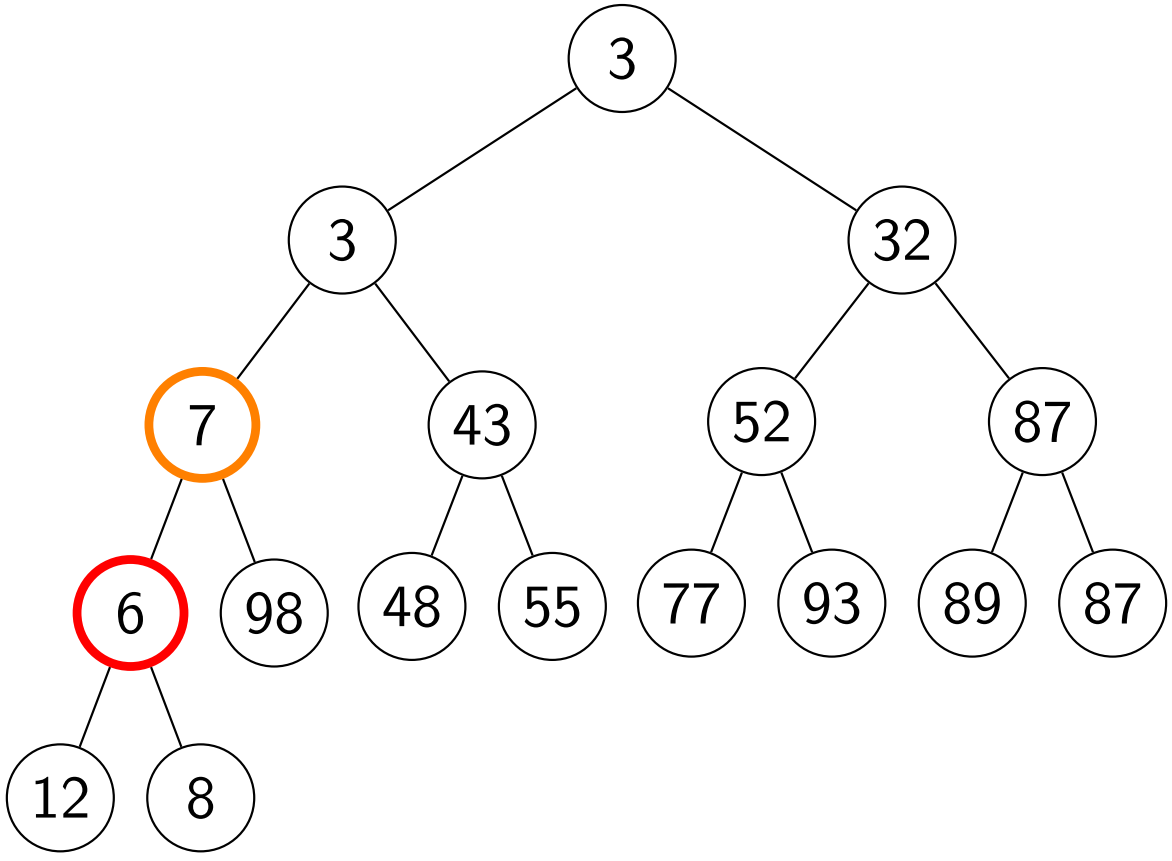


AKCE: probublej 6 nahoru

Vložení prvku - příklad

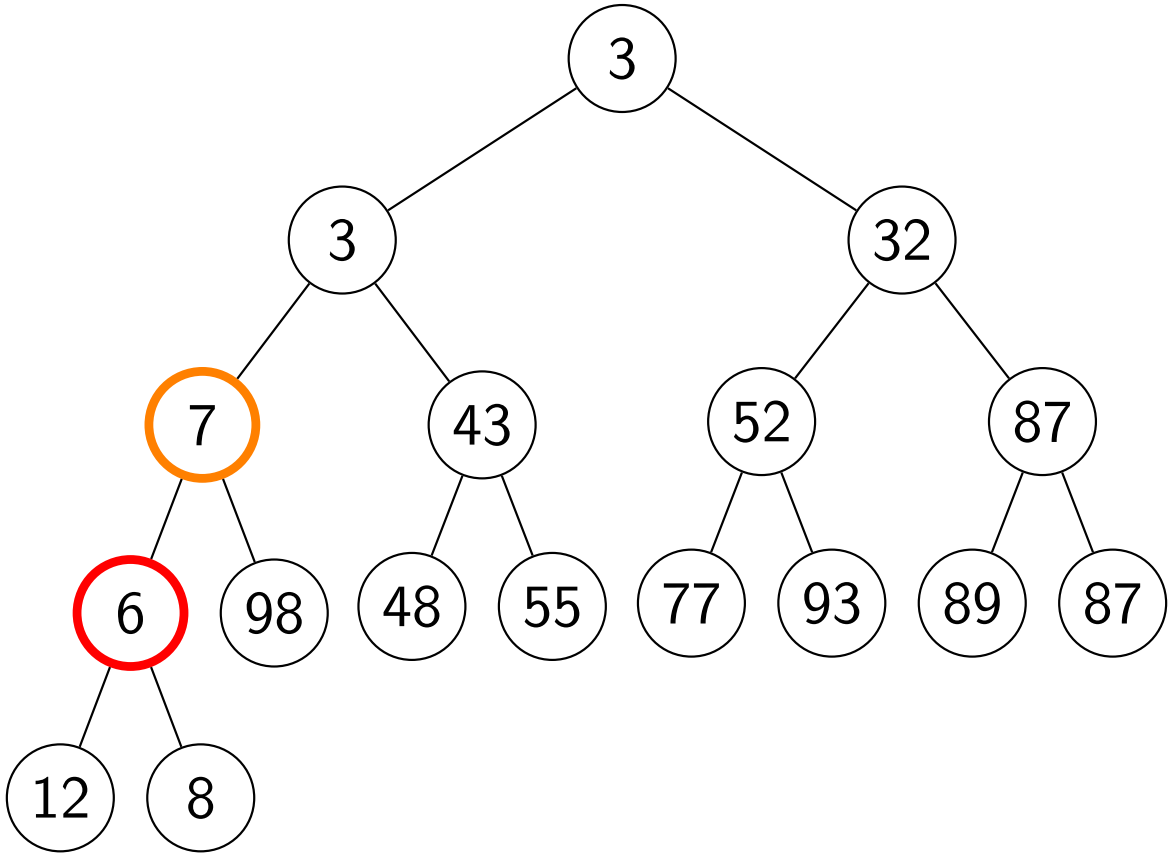


Vložení prvku - příklad



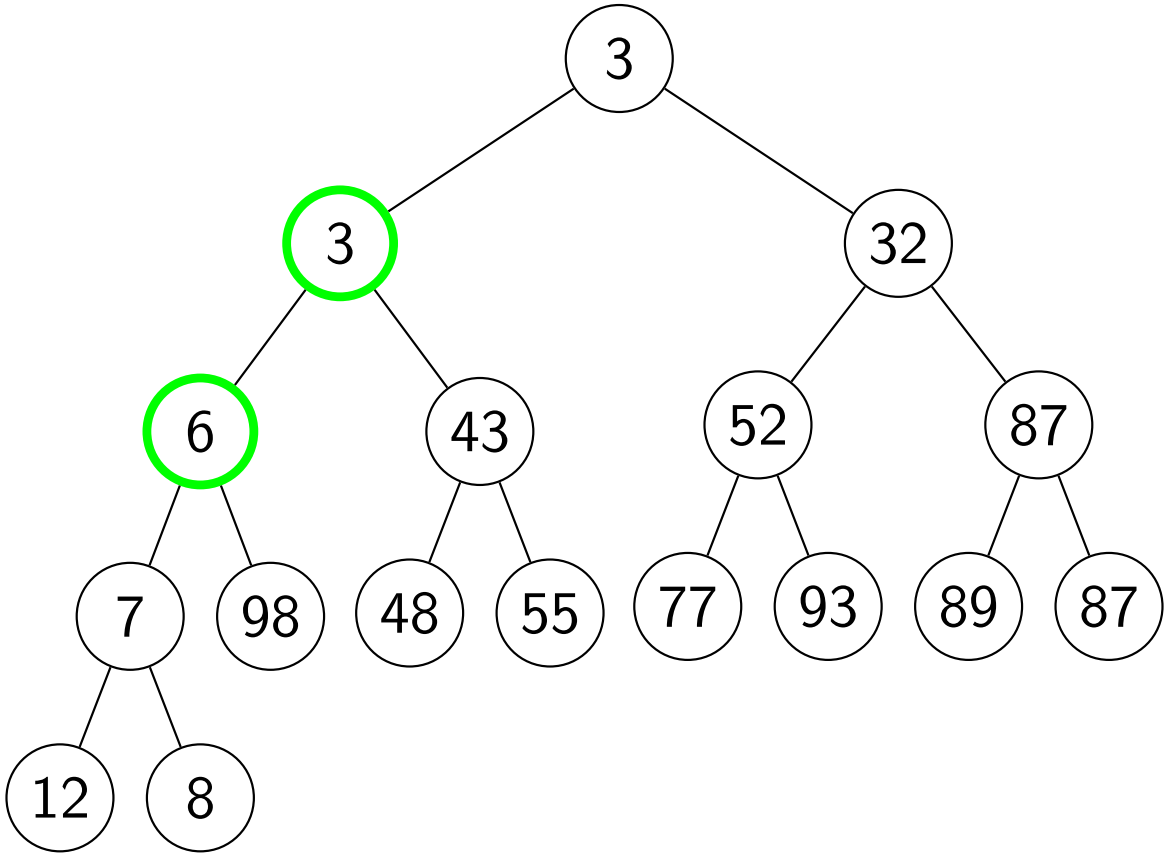
AKCE:

Vložení prvku - příklad

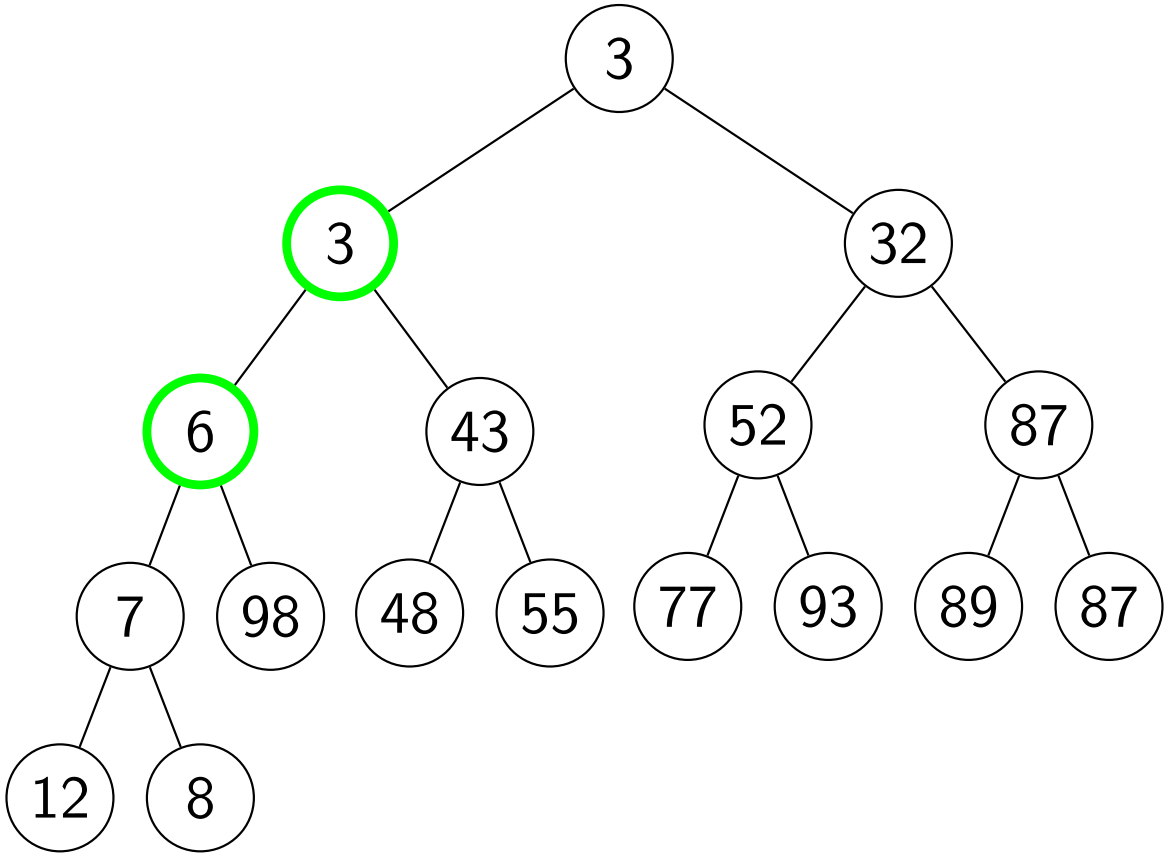


AKCE: probublej 6 nahoru

Vložení prvku - příklad

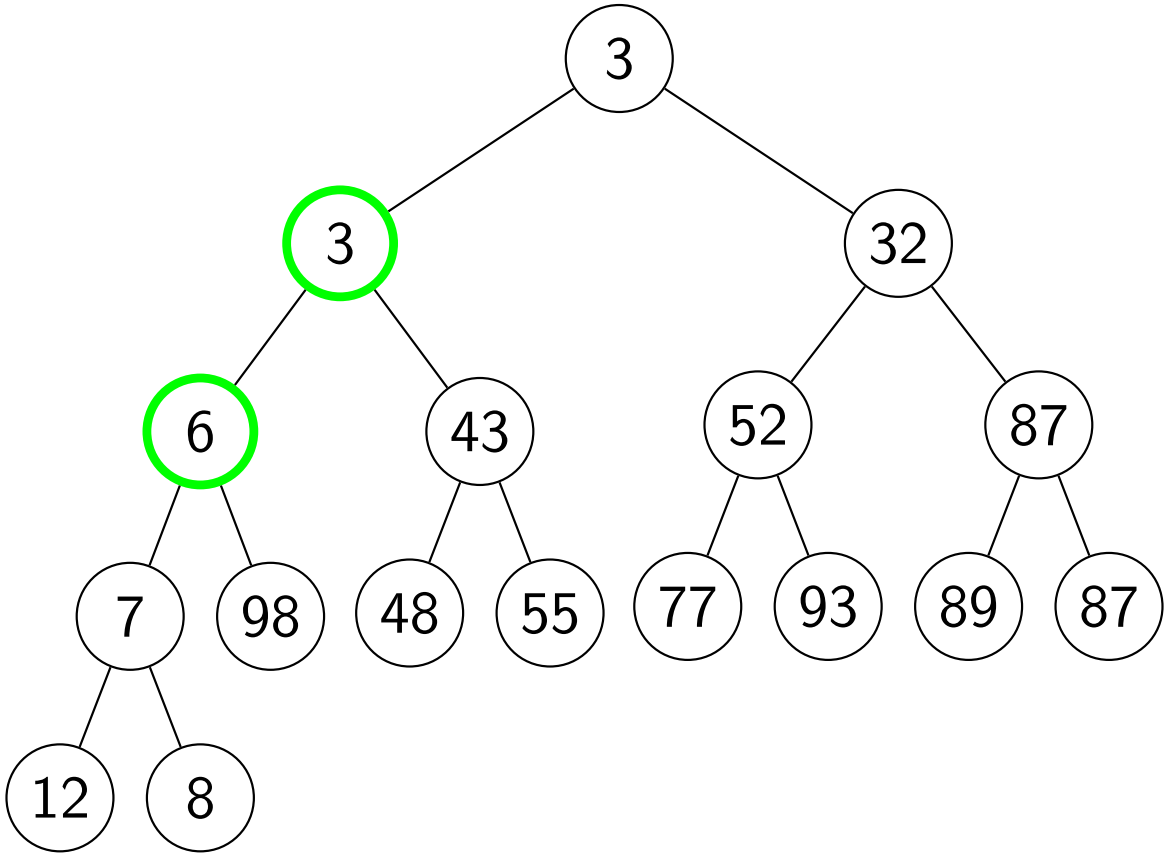


Vložení prvku - příklad



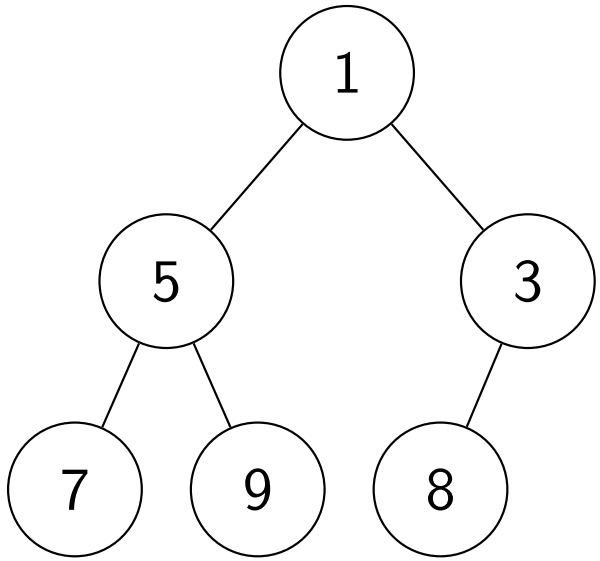
AKCE:

Vložení prvku - příklad

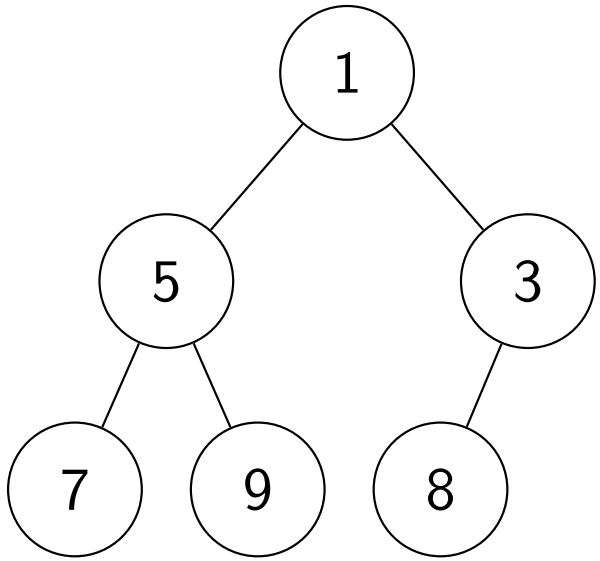


AKCE: stop

Nalezení minimálního prvku

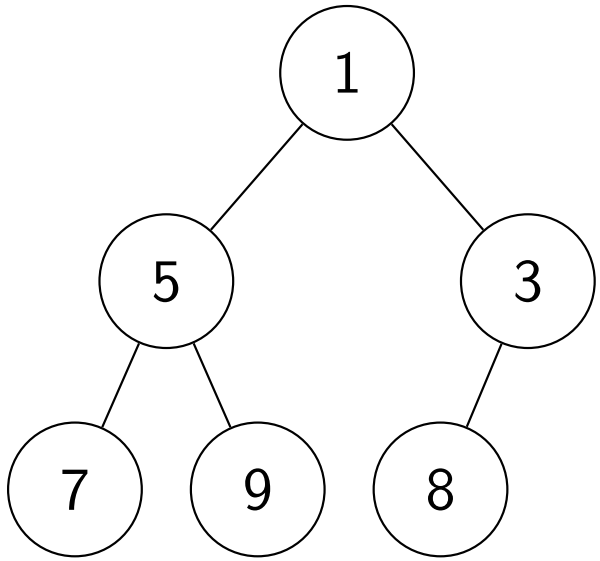


Nalezení minimálního prvku



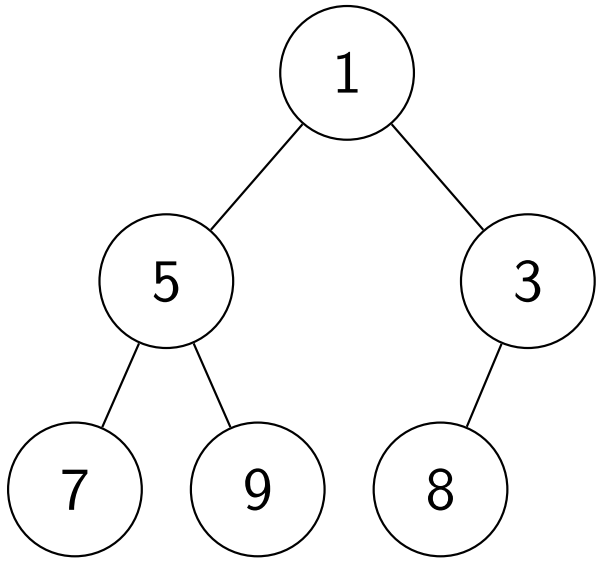
- Kde se nachází minimum?

Nalezení minimálního prvku



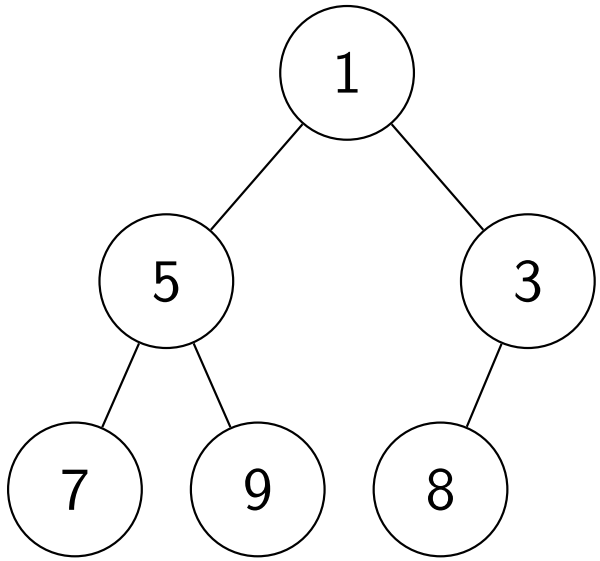
- Kde se nachází minimum? V kořeni

Nalezení minimálního prvku



- Kde se nachází minimum? V kořeni
- Jaká je časová složitost?

Nalezení minimálního prvku

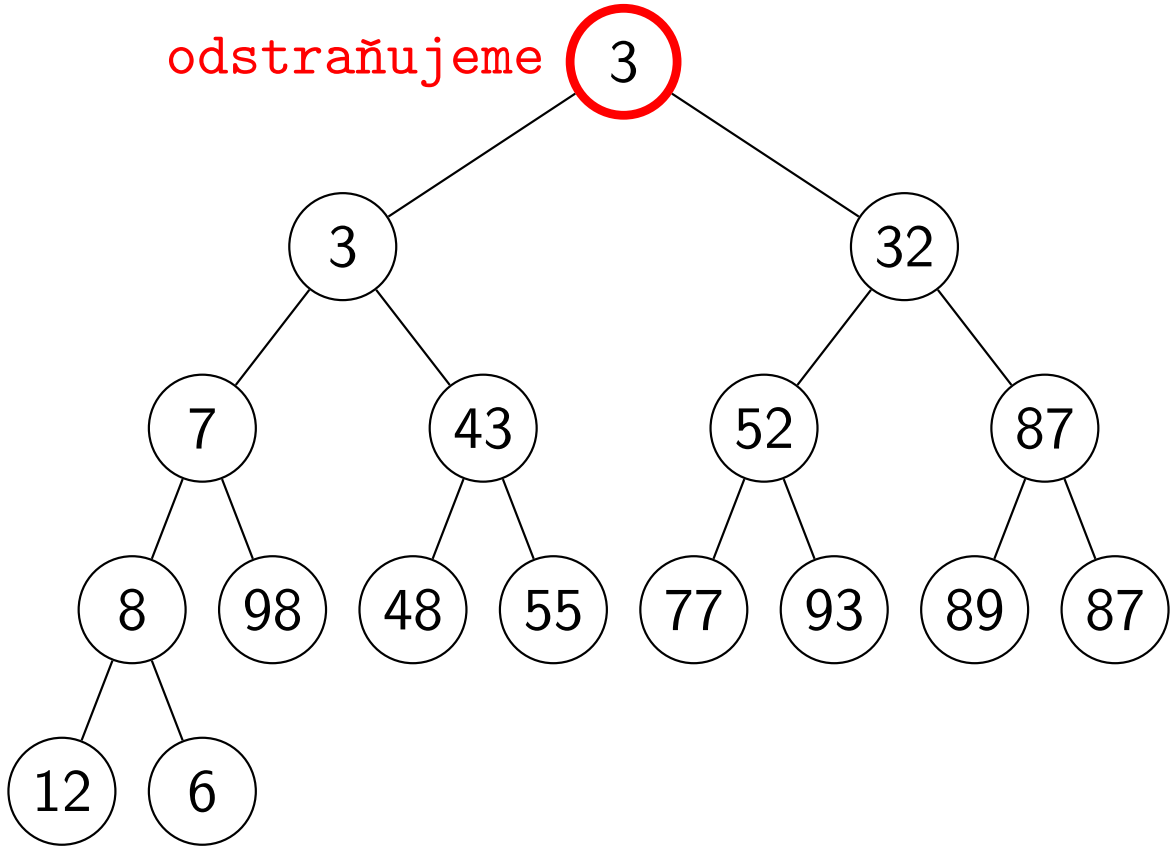


- Kde se nachází minimum? V kořeni
- Jaká je časová složitost? $\mathcal{O}(1)$

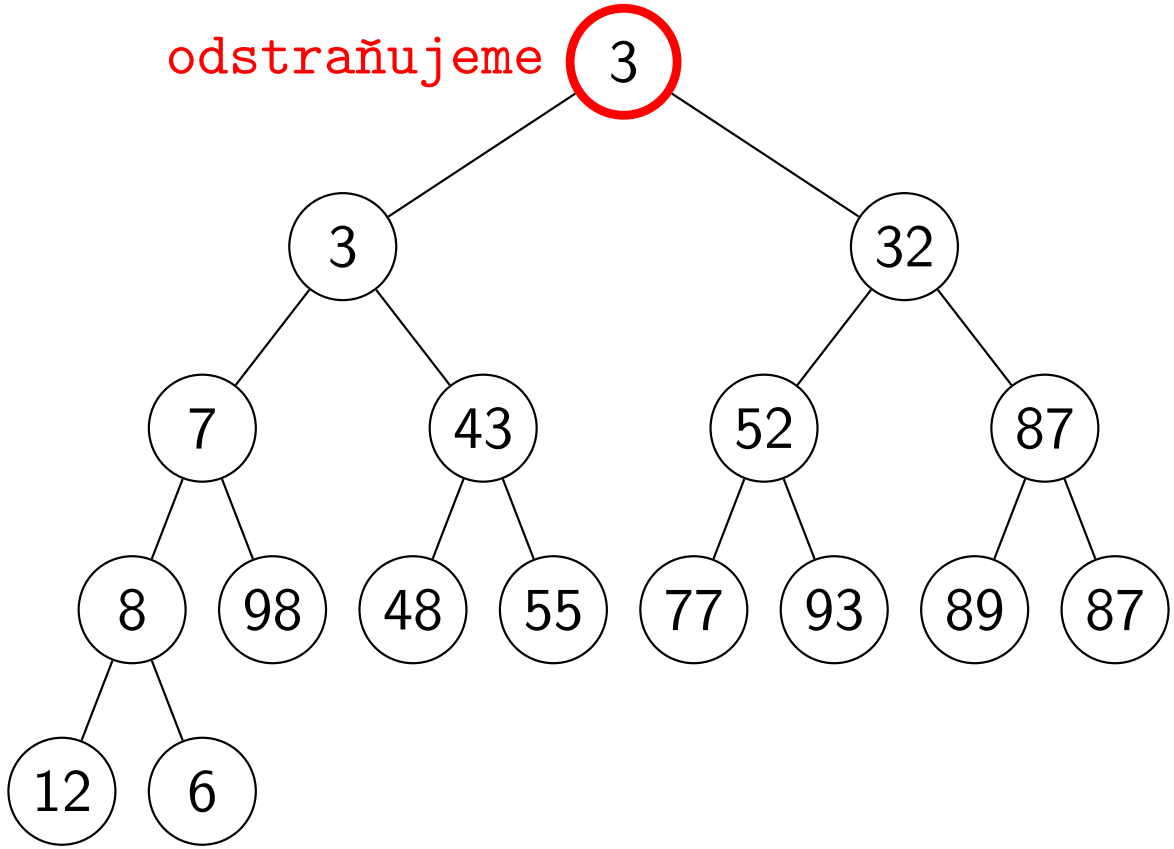
Odstranění minimálního prvku

```
Heap HeapExtractMin(Heap heap) {
  int last = heap.elementsCount - 1;

  HeapSwap(heap, 0, last);

  heap.elementsCount--;

  if (0 < heap.elementsCount) {
    HeapBubbleDown(heap, 0);
  }

  return heap;
}
```

Odstranění minimálního prvku - příklad

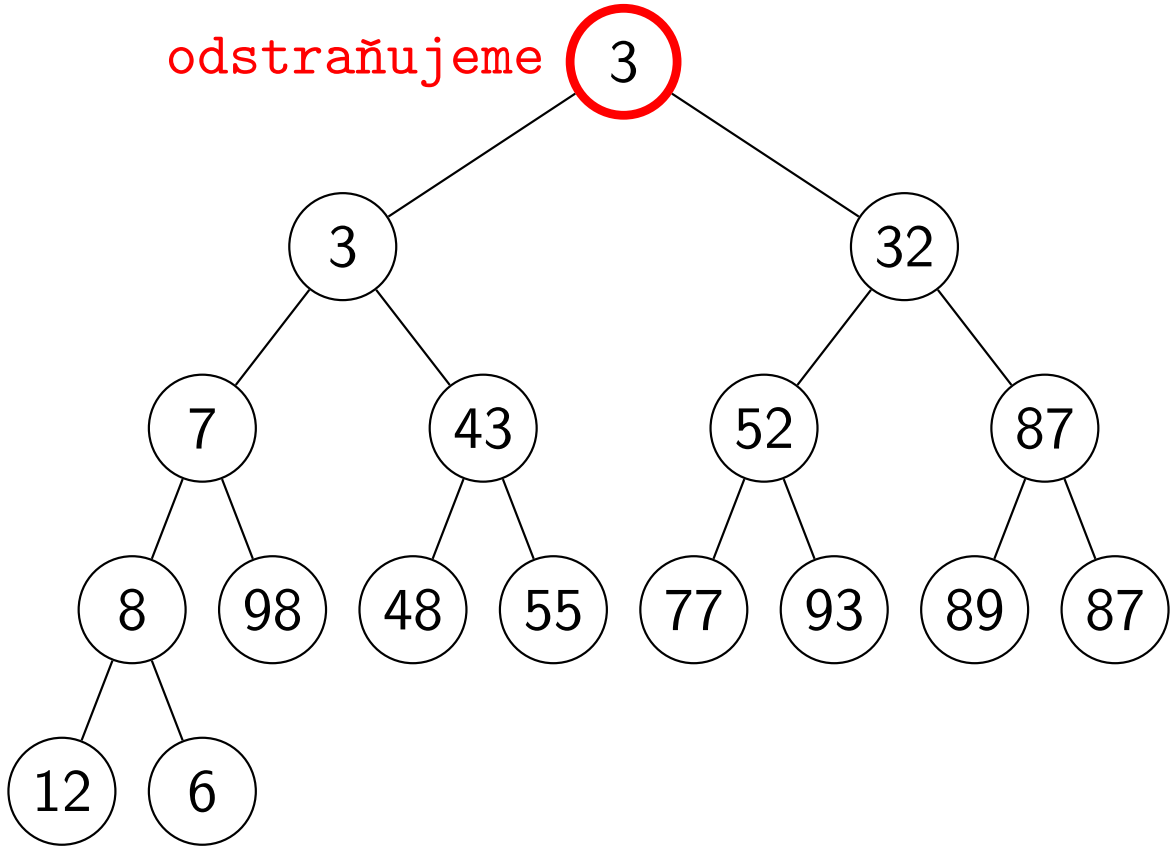


Odstranění minimálního prvku - příklad



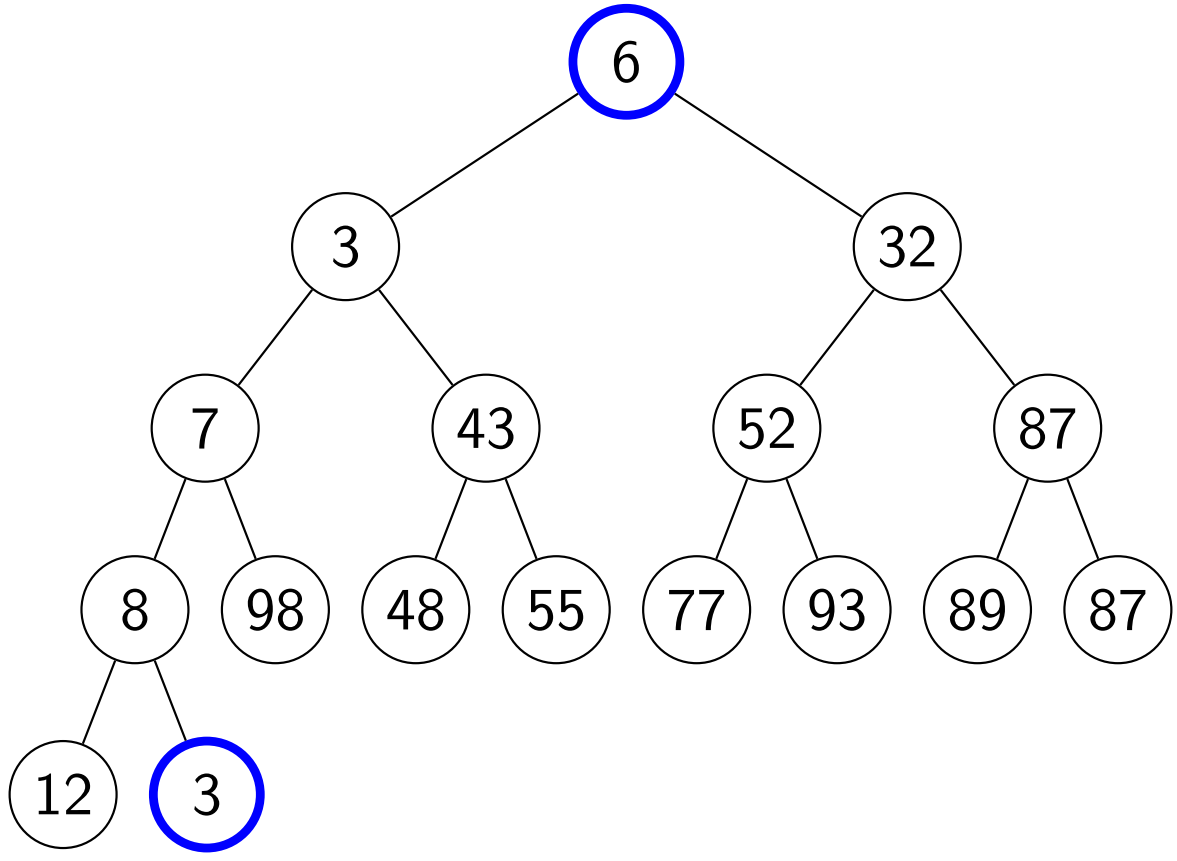
AKCE:

Odstranění minimálního prvku - příklad

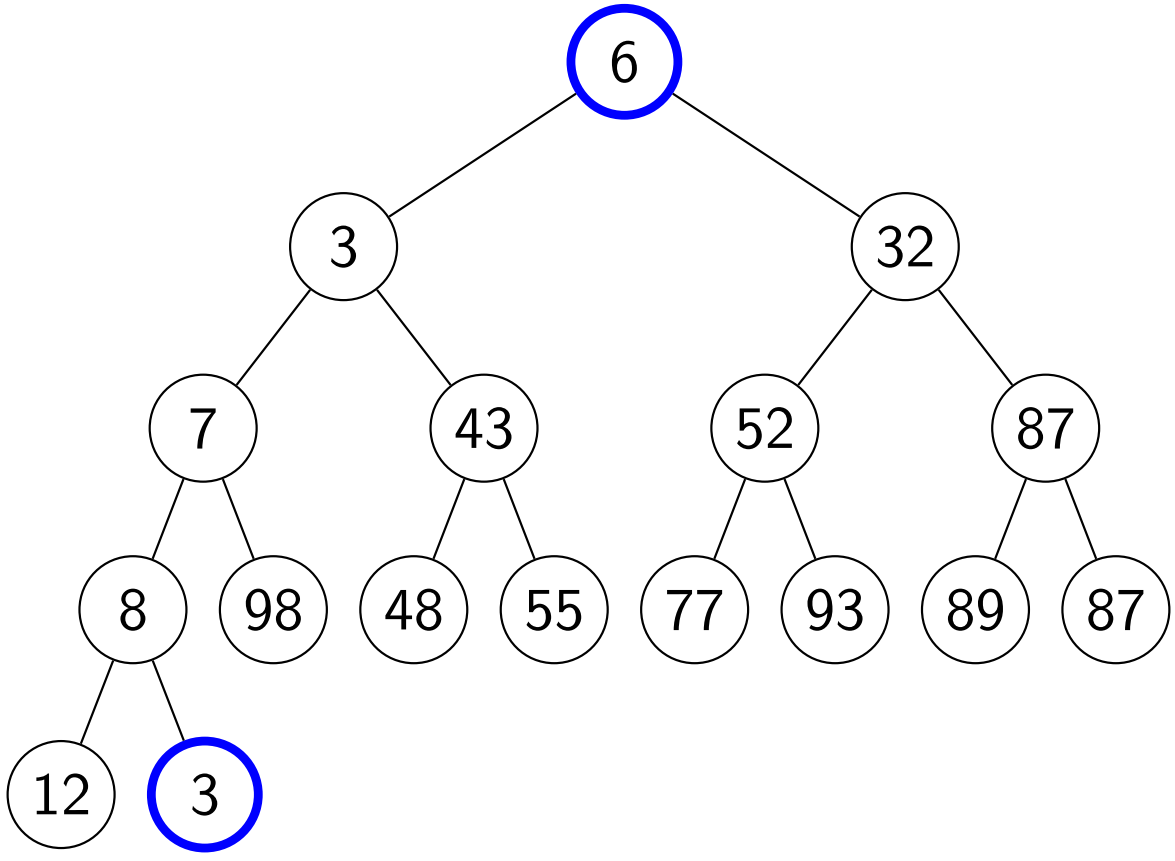


AKCE: prohoď 3 (kořen) a 6 (poslední vrchol)

Odstranění minimálního prvku - příklad

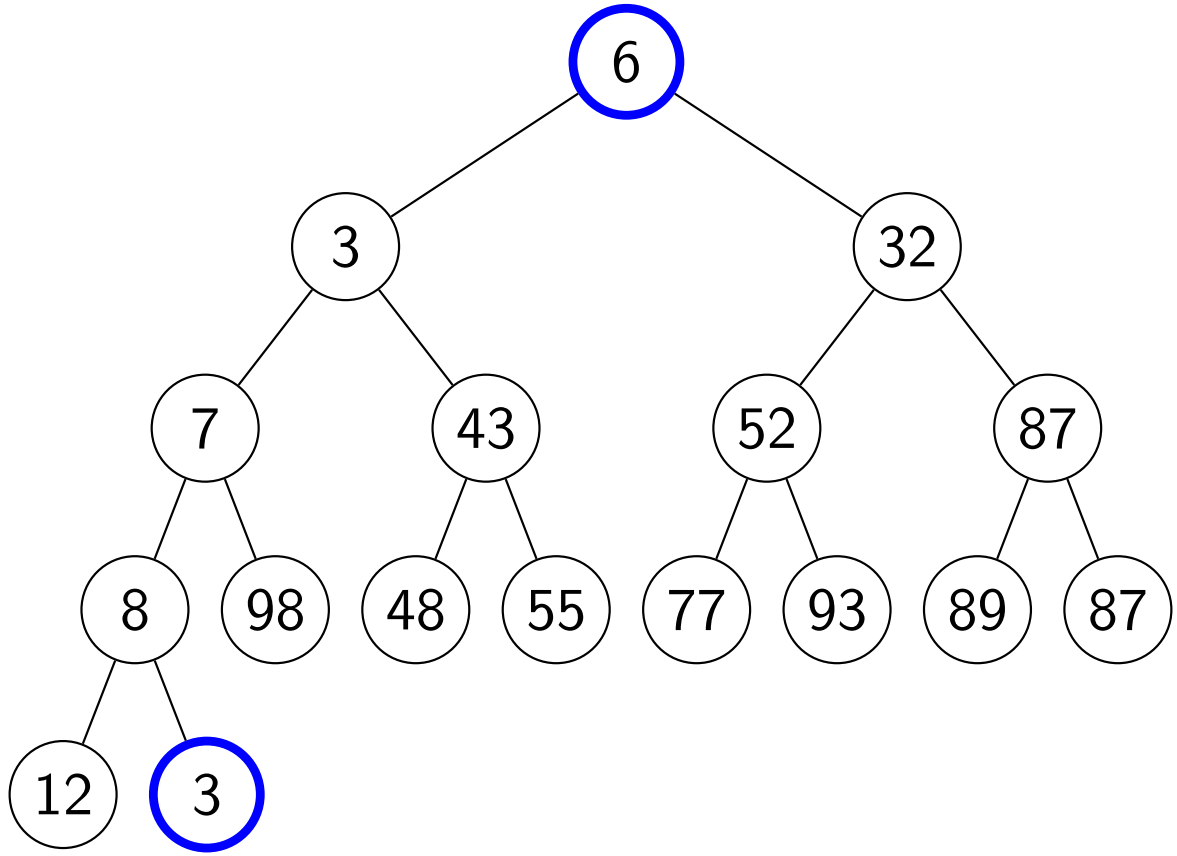


Odstranění minimálního prvku - příklad



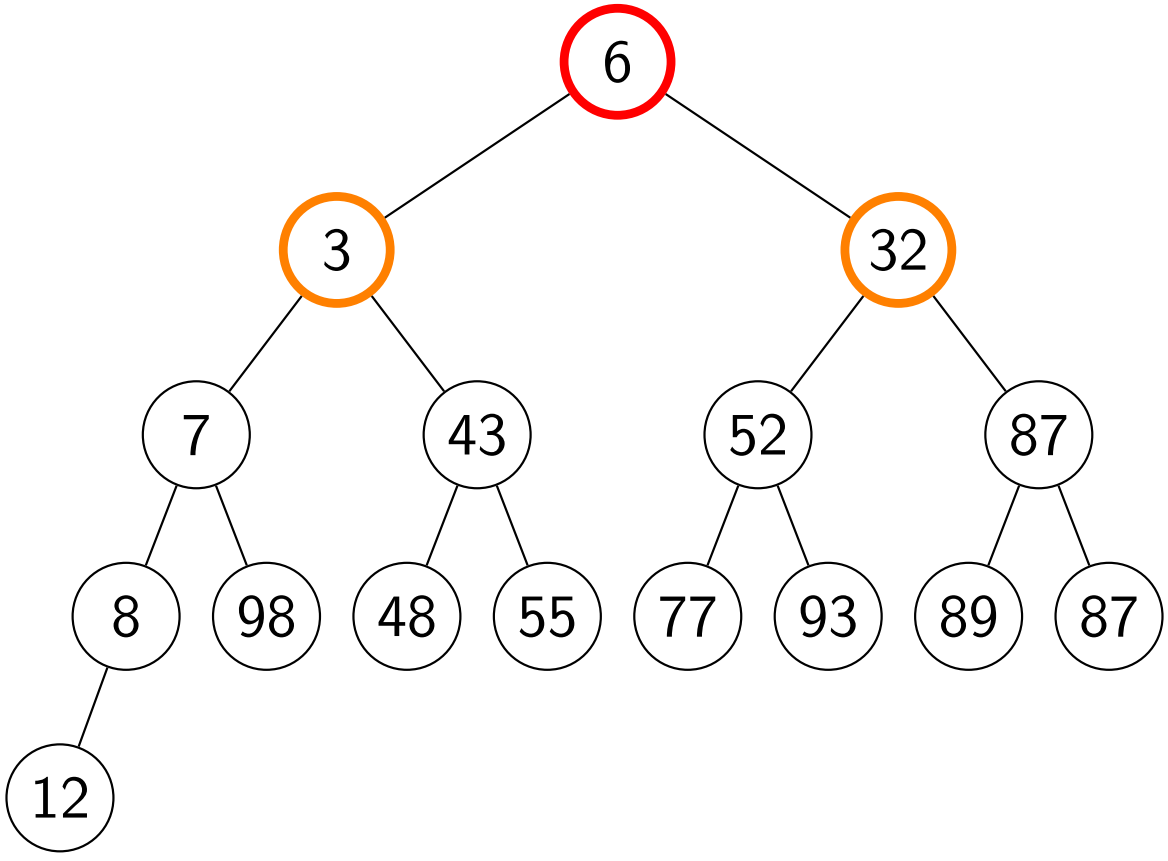
AKCE:

Odstranění minimálního prvku - příklad

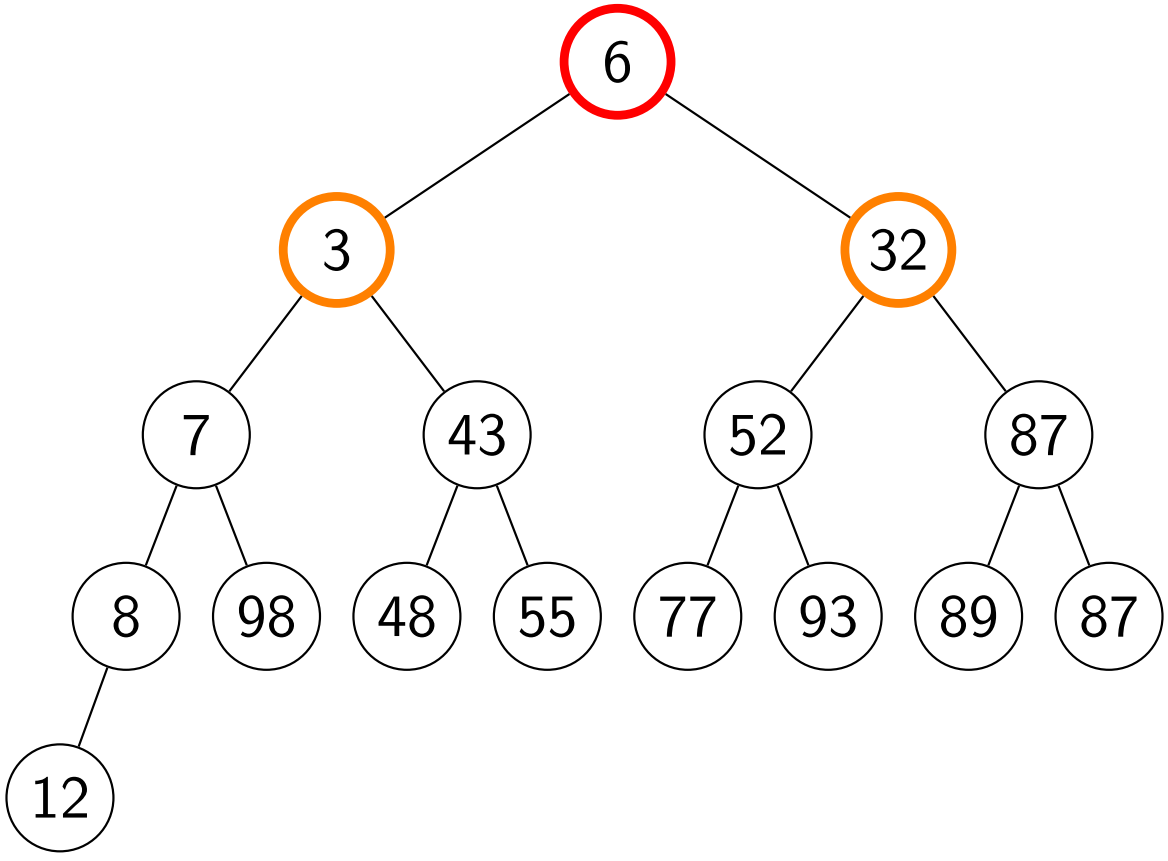


AKCE: odstraň 3 (poslední prvek)

Odstranění minimálního prvku - příklad

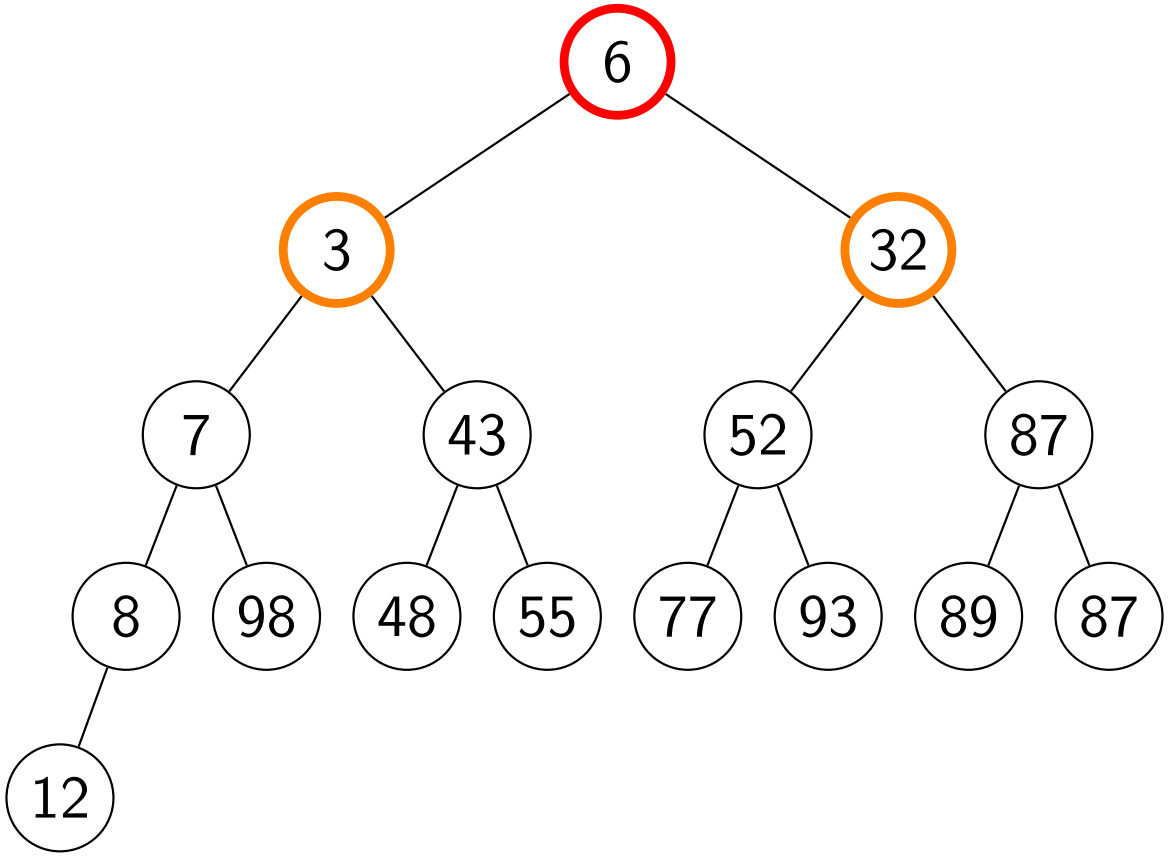


Odstranění minimálního prvku - příklad



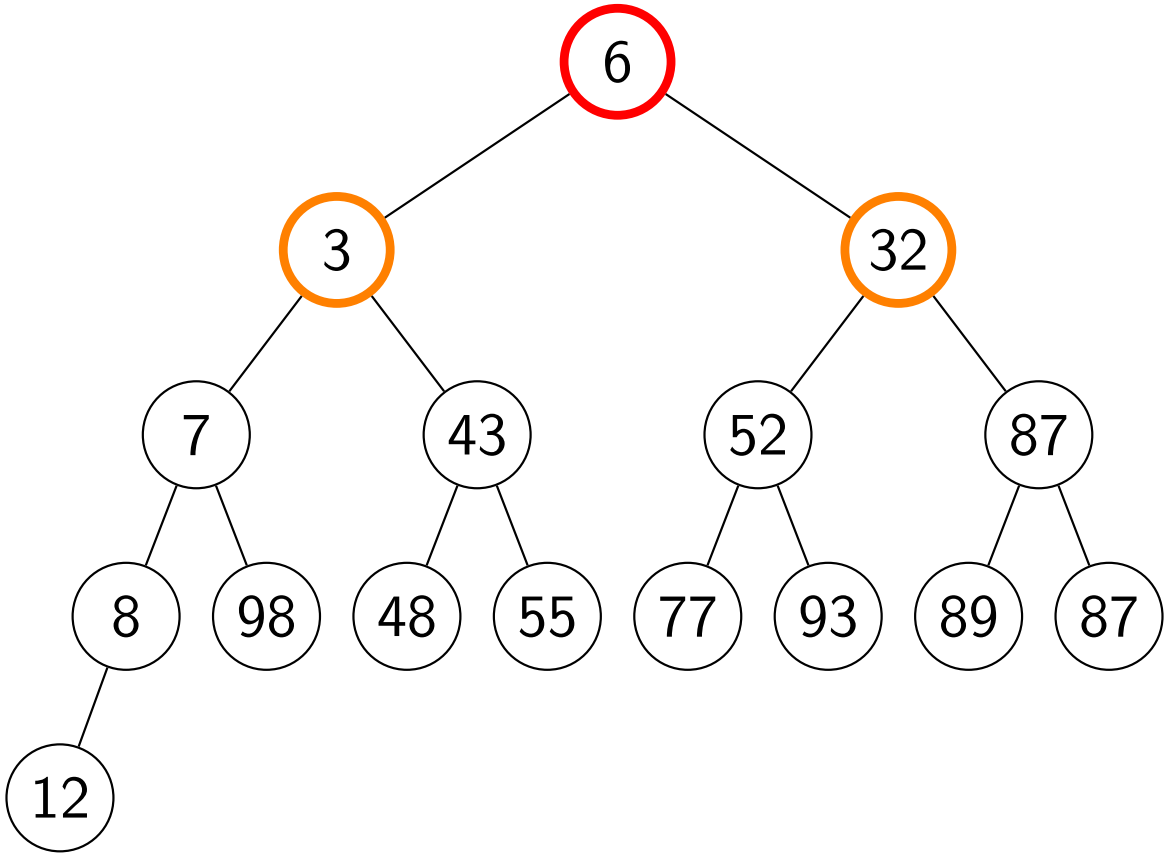
AKCE:

Odstranění minimálního prvku - příklad



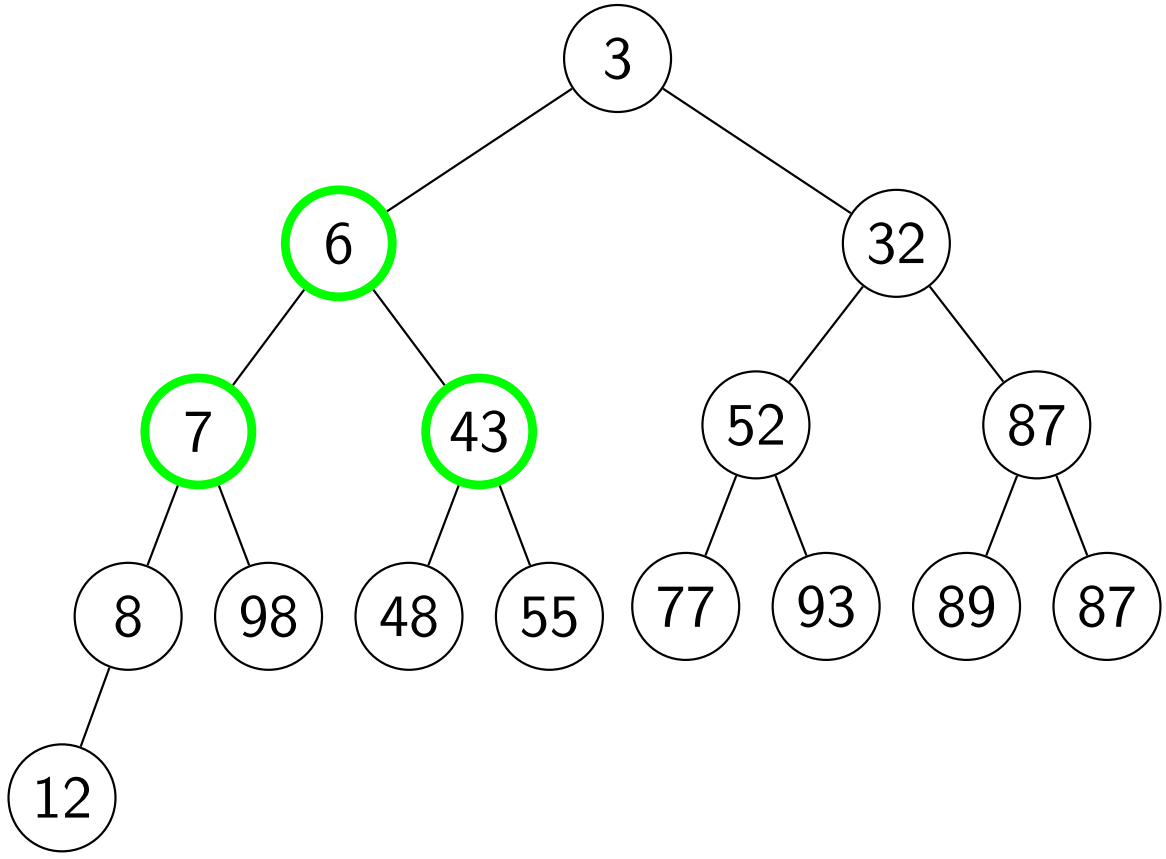
AKCE: probublej 6 dolů

Odstranění minimálního prvku - příklad

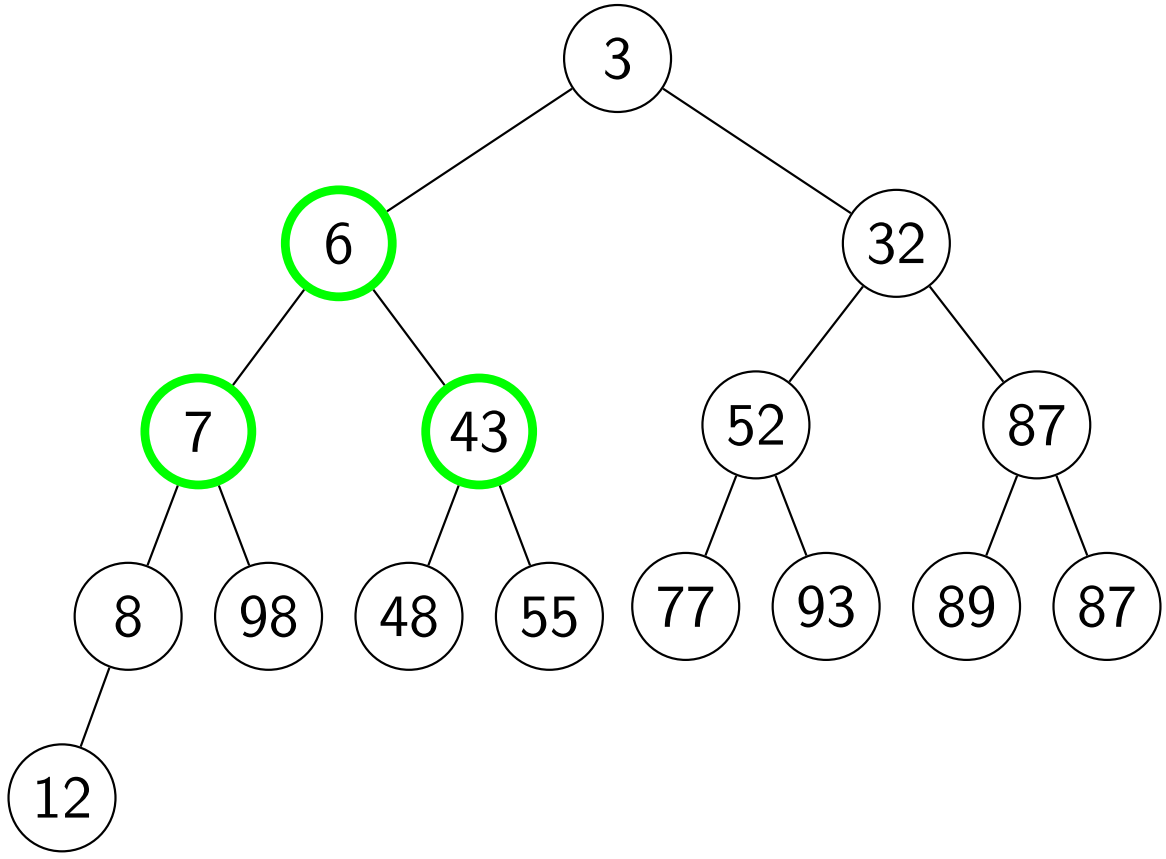


AKCE: probublej 6 dolů doleva

Odstranění minimálního prvku - příklad

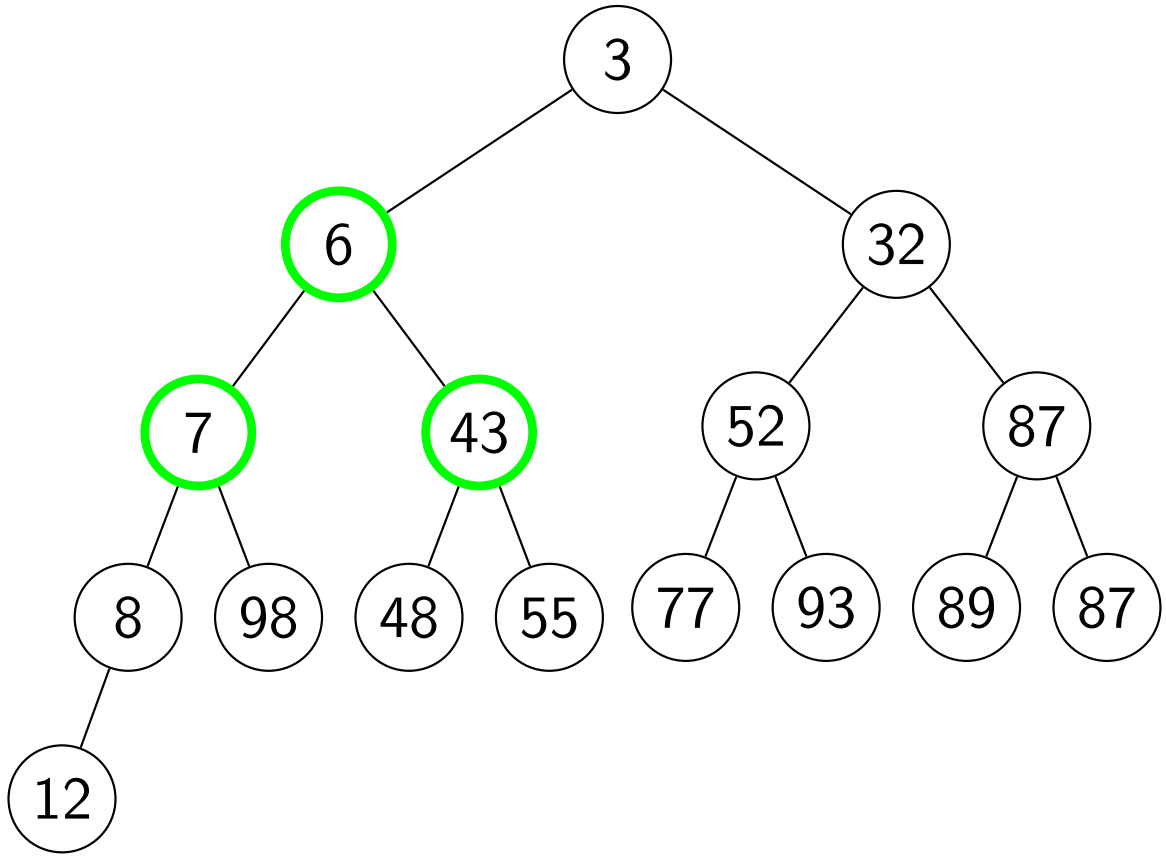


Odstranění minimálního prvku - příklad



AKCE:

Odstranění minimálního prvku - příklad



AKCE: stop

Sestavení haldy

```
Heap HeapBuild(HeapElement *elements, int elementsCount, int elementsCapacity) {
  Heap heap;
  heap.elements = elements;
  heap.elementsCount = elementsCount;
  heap.elementsCapacity = elementsCapacity;
  for (int index = HeapInternalNodeCount(heap) - 1; index >= 0; index--) {
    HeapBubbleDown(heap, index);
  }

  return heap;
}
```

Heapsort

- Používá maximovou haldu

Heapsort

- Používá maximovou haldu
- Třídí pole „od konce“

Algoritmus HeapSort (třídění haldou)
Vstup: Pole $x_1, \ldots, x_n$

1. Pro $i = \lfloor n/2 \rfloor, \ldots, 1$:
2. HsBubbleDown(n, i)
3. Pro $i = n, \ldots, 2$:
4. Prohodíme x_1 s x_i.
5. HsBubbleDown$(i - 1, 1)$

Výstup: Setříděné pole $x_1, \ldots, x_n$