

## Binære søgetræer med ekstra information i knuderne

## Tilføj ekstra information i knuderne

Konkret eksempel på ekstra information i knuder:

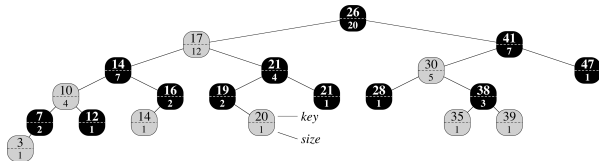
Alle knuder gemmer størrelsen af deres undertræ (dvs. antallet af knuder i deres undertræ).

# Tilføj ekstra information i knuderne

Konkret eksempel på ekstra information i knuder:

Alle knuder gemmer størrelsen af deres undertræ (dvs. antallet af knuder i deres undertræ).

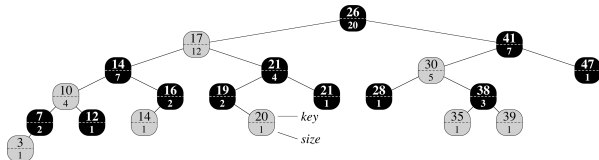
Et binært søgetræ (rød-sort) med denne information tilføjet:



## Ny funktionalitet

Målet med ekstra information i knuderne er at tilføje ekstra funktionalitet. F.eks. kan man i eksemplet ovenfor (med størrelse af undertræer gemt i knuder) udføre flg. operationer i  $O(\log n)$  tid:

- Find rang af en given nøgle.
- Find nøgle som har en given rang.

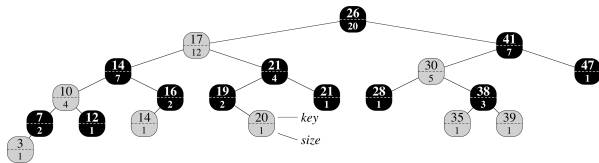


Her er **rang** nøglens nummer i sorteret orden (blandt de gemte).

|        |   |   |    |    |    |    |    |    |    |    |    |     |
|--------|---|---|----|----|----|----|----|----|----|----|----|-----|
| Nøgle: | 3 | 7 | 10 | 12 | 14 | 14 | 16 | 17 | 19 | 20 | 21 | ... |
| Rang:  | 1 | 2 | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | ... |

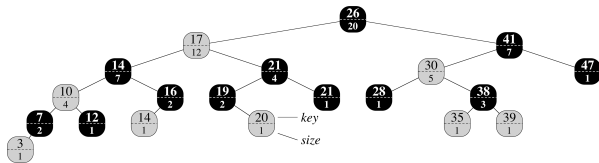
# Ny funktionalitet, implementation

- Find rang af en given nøgle.
- Find nøgle som har en given rang.



# Ny funktionalitet, implementation

- ▶ Find rang af en given nøgle.
- ▶ Find nøgle som har en given rang.



OS-RANK( $T, x$ )

$r = x.\text{left.size} + 1$

$y = x$

**while**  $y \neq T.\text{root}$

**if**  $y == y.p.\text{right}$

$r = r + y.p.\text{left.size} + 1$

$y = y.p$

**return**  $r$

OS-SELECT( $x, i$ )

$r = x.\text{left.size} + 1$

**if**  $i == r$

**return**  $x$

**elseif**  $i < r$

**return** OS-SELECT( $x.\text{left}, i$ )

**else return** OS-SELECT( $x.\text{right}, i - r$ )

[NB: denne pseudo-kode fra bogen antager, at der bruges et eksplikt knude-objekt (med  $\text{size} = 0$ ) til at repræsentere NIL (blade).]

## Vedligehold den ekstra information i knuderne

Antag følgende egenskab gælder for den ekstra information:

Hvis en knudes to børns værdier  $k_1$  og  $k_2$  allerede er korrekte, kan knudes egen værdi  $k$  beregnes i  $O(1)$  tid. Et blads værdi kan beregnes i  $O(1)$  tid.

I eksemplet ovenfor:  $k$  kan findes som  $1 + k_1 + k_2$ .

# Vedligehold den ekstra information i knuderne

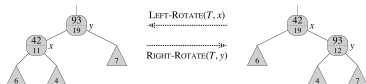
Antag følgende egenskab gælder for den ekstra information:

Hvis en knudes to børns værdier  $k_1$  og  $k_2$  allerede er korrekte, kan knudes egen værdi  $k$  beregnes i  $O(1)$  tid. Et blads værdi kan beregnes i  $O(1)$  tid.

I eksemplet ovenfor:  $k$  kan findes som  $1 + k_1 + k_2$ .

Af antagelsen følger, at værdierne kan vedligeholdes under updates i rød-sort søgetræer, og dette uden at ændre  $O(\log n)$  køretiden:

- ▶ Knuder uden for sti fra indsat/slettet knude til rod skal ikke have deres værdi ændret (bottom-up beregning giver uændret resultat).
- ▶ Under rebalancering: genberegner for berørte knuder efter hver rotation og efter hvert skridt opad. Til sidst: genberegner på sti fra sidste rebalanceringssted til rod.





## Andre eksempler

Vedligeholdelsessystemet ovenfor vil altid fungere når følgende gælder:

Hvis en knudes to børns værdier  $k_1$  og  $k_2$  allerede er korrekte, kan knudes egen værdi  $k$  beregnes i  $O(1)$  tid. Et blads værdi kan beregnes i  $O(1)$  tid.

Flere eksempler på information som opfylder dette (vi antager, at hvert element udover søgenøglen har noget associeret data, som er et tal):

- ▶ Maksimum af data-værdier i undertræet.
- ▶ Minimum af data-værdier i undertræet.
- ▶ Sum af data-værdier i undertræet.

Via sådan information kan man tilføje yderligere funktionalitet til træet. F.eks. kan man søge efter elementer med den maksimale dataværdi. Eller man kan finde gennemsnittet af dataværdierne i en knudes undertræ (gennemsnit = sum af dataværdier / antal knuder).