LOUDS++

UNIVERSITÄT OSNABRÜCK
AG Theoretische Informatik

Finn Stutzenstein, Levin Nemesch, Joshua Sangmeister

November 30, 2020

Algorithm Engineering - Übung 2

Überblick

- LOUDS: Level-Order Unary Degree Sequence
- *succinct data structure*: $2n + o(n)$ Bits Speicherbedarf
- Geordneter Baum
- Unterstützt folgende Operationen in $\mathcal{O}(1)$:
    - parent(x)
    - first-child(x)/last-child(x)
    - prev-sibling(x)/next-sibling(x)
    - degree(x): Anzahl der Kinder von x
    - childrank(x): Rang von x unter seinen Geschwistern
    - child(x, i): i-tes Kind von x
- Quelle: "Engineering the LOUDS Succinct Tree Representation" von O'Neil Delpratt, Naila Rahman, and Rajeev Raman (`https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.4250&rep=rep1&type=pdf`)

Bitvektor

- Bitfeld von n Bits
- $\texttt{rank}_b(x)$ ($b \in \{0,1\}$): Anzahl der Vorkommnisse von b bis x
- $\texttt{select}_b(x)$ ($b \in \{0,1\}$): Position des x-ten Vorkommens von b
- Z.B. von Kim Et Al.: Speicher $n + o(n)$ Bits und Laufzeiten in $\mathcal{O}(1)$

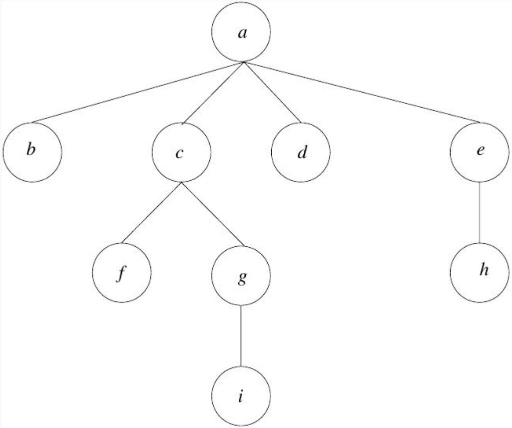
Bitvektor

- Bitfeld von n Bits
- $\texttt{rank}_b(x)$ ($b \in \{0,1\}$): Anzahl der Vorkommnisse von b bis x
- $\texttt{select}_b(x)$ ($b \in \{0,1\}$): Position des x-ten Vorkommens von b
- Z.B. von Kim Et Al.: Speicher $n + o(n)$ Bits und Laufzeiten in $\mathcal{O}(1)$
- Beispiel: `0011011100`
  - $\texttt{rank}_1(7) = 4$
  - $\texttt{rank}_0(7) = 3$
  - $\texttt{select}_1(3) = 6$
  - $\texttt{select}_0(5) = 10$

LOUDS bit string (LBS)

- Speichert unär Baumstruktur
- Jeder Knoten repräsentiert durch 1^d0 Bitstring, d entspricht Anzahl Kinder
- Nummerierung der Knoten ebenenweise (entspricht Breitensuche)
- Beginn markiert durch 10

LOUDS



(a)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**

(b)

Vertex	a	b	c	d	e	f	g	h	i
BFS	1	2	3	4	5	6	7	8	9
1-based	1	3	4	5	6	9	10	13	16
0-based	7	8	11	12	14	15	17	18	19

(c)

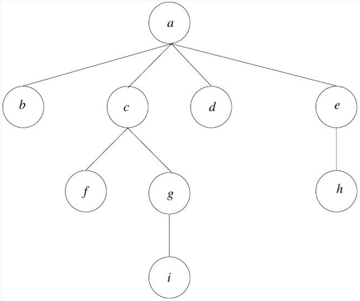
Vertex	$a\ b\ c\ d\ e\ f\ g\ h\ i$
$R0$	**1 0 1 0 1 0 1 0 0 1**
$R1$	**1 0 0 0 1 0 1 1 1**

(d)

Eigenschaften LBS

- Besteht aus $n+1$ 0-en und n 1-en
- i-ter Knoten:
- *Ones-based numbering*
  - i-te 1 in Kodierung des Elternknotens
  - Knoten i wird Zahl $x = \texttt{select}_1(i) \in \{0, \ldots, 2n+1\}$ zugewiesen
  - Knoten mit Zahl x: $i = \texttt{rank}_1(x) \in \{1, \ldots, n\}$
- *Zero-based numbering*: $i+1$-te 0: Ende der Kodierung des i-ten Knotens
  - $i+1$-te 0: Ende der Kodierung des i-ten Knotens
  - Mit $\texttt{select}_0$ und $\texttt{rank}_0$ kann analog zum Ones-based numbering eine Zahl zugeordnet werden

Beispiel: Operation $\texttt{parent}(x)$



(a)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**

(b)

Vertex	a	b	c	d	e	f	g	h	i
BFS	1	2	3	4	5	6	7	8	9
1-based	1	3	4	5	6	9	10	13	16
0-based	7	8	11	12	14	15	17	18	19

(c)

Vertex	$a\ b\ c\ d\ e\ f\ g\ h\ i$
$R0$	**1 0 1 0 1 0 1 0 0 1**
$R1$	**1 0 0 0 1 0 1 1 1**

(d)

- *Ones-based numbering:*
  $x = \texttt{select}_1(\texttt{rank}_0(x))$
- *Zero-based numbering:*
  $\texttt{select}_0(\texttt{rank}_0(\texttt{select}_1(\texttt{rank}_0(x) - 1) + 1))$

Beispiel $\texttt{parent(g)}$ (Ones-based):

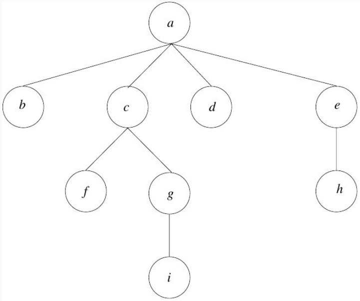
Knoten g (BFS 7) $\rightarrow \texttt{select}_1(7) = 10$

$\texttt{rank}_0(10) = 3$

$\texttt{select}_1(3) = 4$

$\texttt{rank}_1(4) = 3 \rightarrow$ Knoten c (BFS 3)

Beispiel: Operation `parent`(x)



(a)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**

(b)

Vertex	a	b	c	d	e	f	g	h	i
BFS	1	2	3	4	5	6	7	8	9
1-based	1	3	4	5	6	9	10	13	16
0-based	7	8	11	12	14	15	17	18	19

(c)

Vertex		a	b	c	d	e	f	g	h	i
$R0$	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**1**
$R1$		**1**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**1**

(d)

- *Ones-based numbering:*
  $x = \texttt{select}_1(\texttt{rank}_0(x))$
- *Zero-based numbering:*
  $\texttt{select}_0(\texttt{rank}_0(\texttt{select}_1(\texttt{rank}_0(x) - 1) + 1))$

Beispiel `parent(g)` (Zero-based):

$\texttt{rank}_0(17) - 1 = 8 - 1 = 7$

$\texttt{select}_1(7) + 1 = 10 + 1 = 11$

$\texttt{rank}_0(11) = 4$

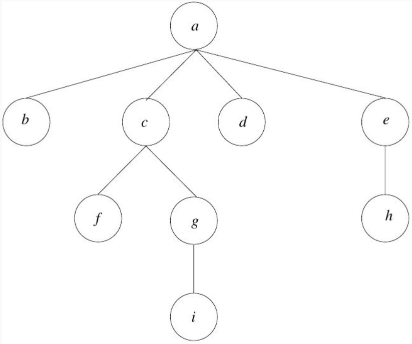
$\texttt{select}_0(4) = 11$

Double-Numbering

- $y = \texttt{select}_0(x) \implies \texttt{rank}_0(y) = x \land \texttt{rank}_1(y) = y - x$
- analog für $\texttt{select}_1$
- Punkte können somit als Paar $\{x, y\}$ gespeichert werden
- Aufrufe der Form `rank(select(...))` können durch einfache `select`s ersetzt werden
- Beispiel für `parent`:

```
parent({x, y}):
    rzerox = y - x
    newy = select_1(rzerox)
    newx = newy - rzerox
    return {newx, newy}
```

Partitioned Representation



(a)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**

(b)

Vertex	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*
BFS	1	2	3	4	5	6	7	8	9
1-based	1	3	4	5	6	9	10	13	16
0-based	7	8	11	12	14	15	17	18	19

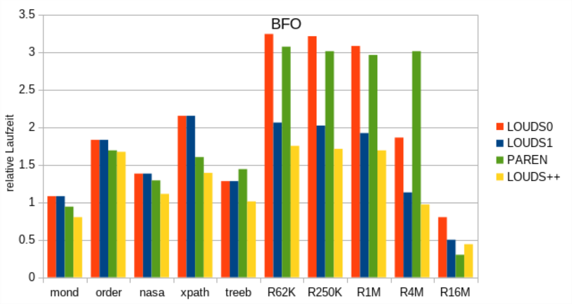
(c)

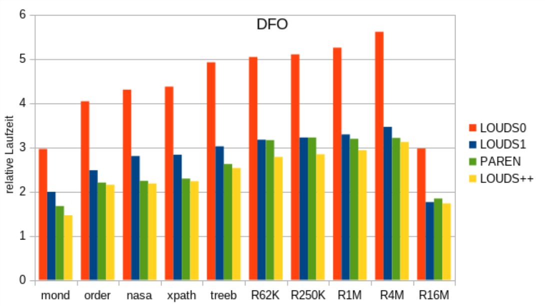
Vertex	*a b c d e f g h i*
$R0$	**1 0 1 0 1 0 1 0 0 1**
$R1$	**1 0 0 0 1 0 1 1 1**

(d)

- R_0: Nullfolgen (mit Längen $\ell_1, \ell_2 \dots \ell_z$) kombinieren zu $R_0 = 0^{\ell_1}10^{\ell_2}1\dots0^{\ell_z}1$
- R_1: Einsfolgen (mit Längen $\ell_1, \ell_2 \dots \ell_z$) kombinieren zu $R_1 = 1^{\ell_1}01^{\ell_2}0\dots1^{\ell_z}0$
- Alle `select` Operationen auf LBS durchführbar durch $\texttt{select}_1$ und $\texttt{rank}_1$ auf R_0 oder R_1
- LOUDS++ nutzt R_0 und R_1

Ergebnisse


BFO
relative Laufzeit
3.5
3
2.5
2
1.5
1
0.5
0
mond
order
nasa
xpath
treeb
R62K
R250K
R1M
R4M
R16M
LOUDS0
LOUDS1
PAREN
LOUDS++


DFO
relative Laufzeit
6
5
4
3
2
1
0
mond
order
nasa
xpath
treeb
R62K
R250K
R1M
R4M
R16M
LOUDS0
LOUDS1
PAREN
LOUDS++