Advanced Data Structures

Lecture 02: Succinct Trees

Florian Kurpicz
Recap: Rank Queries on Bit Vectors (1/2)

\( \text{rank}_\alpha(i) \) # of \( \alpha \)s before \( i \)

\( \text{select}_\alpha(j) \) position of \( j \)-th \( \alpha \)

# of 0s w.r.t. super-block

# of 0s w.r.t. BV
Lemma: Binary Rank- and Select-Queries

Given a bit vector of size $n$, there exist data structures that can be computed in time $O(n)$ of size $o(n)$ bits that can answer rank and select queries on the bit vector in $O(1)$ time.

Word RAM

- unlimited memory
- words of size $w$ \( w = \Theta \log n \)
- constant time load and store
- constant time bit operations on words
Plan for Today

- represent tree with $n$ nodes using $2n$ bits
- make succinct tree fully-functional using additional $o(n)$ bits

- trees are important
  - searching for keys
  - maintaining directories
  - representations of parsings
  - . . .

- different representations
- supporting different operations

Handout
Preliminaries

- A tree is an acyclic connected graph \( G = (V, E) \) with a root \( r \in V \).
- Degree \( \delta \) is the number of children.
- Leaves have degree 0.
- Depth of a node is the length of the path from the root to that node.
Level Ordered Unary Degree Sequence (1/2) [Jac88]

- represent tree level-wise
- use \( \leq 2 \) bits per node

**Definition: LOUDS**

Starting at the root, all nodes on the same depth
- are visited from left to right and
- for node \( v \), \( \delta(v) \) 1’s followed by a 0 are appended to the bit vector that contains an initial 10

**Lemma: Space Usage of LOUDS**

Representing a tree with \( n \) nodes requires \( 2n + 1 \) bits using LOUDS

- write down the LOUDS representation of this example tree
Level Ordered Unary Degree Sequence (2/2)

```
ab ch id ejkfg
1011100110011001100000
```

- node start at pertinent 0
What is Fully-Functional?

Operations

- degree is leaf
- i-th child
- parent
- subtree size

- depth
- lowest common ancestor
- rank (pre- or post-order)
- ...

![Tree diagram](image)
Making LOUDS Fully-Functional

- degree of $p$: $p - \text{select}_0(\text{rank}_0(p)) - 1$
- $i$-th child of $p$:
  $\text{select}_0(\text{rank}_1(\text{select}_0(\text{rank}_0(p)))) + i + 1$
- parent of $p$:
  $\text{select}_0(\text{rank}_0(\text{select}_1(\text{rank}_0(p)))) + 1$

- explanation on the board 📣

- subtree size 📆 PINGO
Definition: Balanced String of Parentheses

A string of parentheses is balanced, if for each left parenthesis there exist unique right parenthesis to its right.

- \textit{findclose}(i)\textsuperscript{:} find the right parenthesis matching the left parenthesis at position \( i \)
- \textit{findopen}(i)\textsuperscript{:} find the left parenthesis matching the right parenthesis at position \( i \)
- \textit{excess}(i)\textsuperscript{:} find the difference between the number of left and right parentheses before position \( i \)
- \textit{enclose}(i)\textsuperscript{:} given a parentheses pair with the left parenthesis at position \( i \), return the position of the closest left parenthesis belonging to the parentheses pair enclosing it

- how can we answer \textit{excess} queries

- instead of 0 and 1
- use \((\text{and})\)

- requires the same space
- can add relation between parentheses

From Bit Vectors to Parentheses

2022-05-02 Florian Kurpicz | Advanced Data Structures | 02 Succinct Trees

Institute of Theoretical Informatics, Algorithm Engineering
From Bit Vectors to Parentheses

- all parentheses operations can be answered in $O(1)$ time using $o(n)$ bits space
- here, a little bit simpler

- $\text{excess}(i) = \text{rank}^\text{"("}(i) - \text{rank}^\text{")"}(i)$
- $\text{fwd\_search}(i, d) = \min\{j > i : \text{excess}(j) - \text{excess}(i - 1) = d\}$
- $\text{bwd\_search}(i, d) = \max\{j < i : \text{excess}(i) - \text{excess}(j - 1) = d\}$

- $\text{findclose}(i) = \text{fwd\_search}(i, 0)$
- $\text{findopen}(i) = \text{bwd\_search}(i, 0)$
- $\text{enclose}(i) = \text{bwd\_search}(i, 2)$

- can be answered with a min-max-tree later in this course
Balanced Parentheses (1/2) [MR01]

- represent tree as depth-first traversal
- using balanced parentheses

**Definition: BP**
Starting at the root, traverse the tree in depth-first order and append a

- left parenthesis if a node is visited the first time
- right parenthesis if a node is visited the last time
to the bit vector

**Lemma: Space Usage of BP**
Representing a tree with \( n \) nodes requires \( 2n \) bits using BP

- write down the BP representation of this example tree
Balanced Parentheses (2/2)

- node starts at first parenthesis
- subtree structure is encoded in parentheses

- Diagram showing a balanced parentheses structure

ab cd ef g h ij k

(((()))(()))((()())())
Making BP Fully-Functional

- subtree size of $p$: $\left(\text{findclose}(p) - p + 1\right)/2$
- parent of $p$: $\text{enclose}(p)$

- explanation on the board

Complicated Constant Time [NS14]
- degree
- $i$-th child

Complicated Constant Time [NS14]
Advantages and Disadvantages of Both Approaches

- LOUDS cannot answer subtree size
- BP cannot easily answer $i$-th child and degree
- all other operations can be done easily
Definition: DFUDS
Starting at the root, traverse tree in depth-first order and append
- for node \(v\), \(\delta(v)\) left parentheses and
- a right parenthesis if \(v\) is visited the first time
to the bit vector that initially contains a left parenthesis to make them balanced.

Lemma: Space Usage of DFUDS
Representing a tree with \(n\) nodes requires \(2n\) bits using DFUDS.

Write down the DFUDS representation of this example tree.
Depth First Unary Degree Sequence (2/2)

- node starts at first parenthesis
- subtree structure is encoded

a bc de fghi jk
((((()))(())))()())
Making DFUDS Fully-Functional

- **degree of** $p$: $\text{select}^\omega\left(\text{rank}^\omega(p) + 1\right) - p$
- **$i$-th child of** $p$: $\text{findclose}(\text{select}^\omega(\text{rank}^\omega(p) + 1) - i) + 1$
- **parent of** $p$: $\text{select}^\omega(\text{rank}^\omega(\text{findopen}(p - 1))) + 1$
- **subtree size of** $p$: $(\text{findclose}(\text{enclose}(p)) - p)/2 + 1$

- **explanation on the board** 📚
Conclusion and Outlook

This Lecture
- three succinct tree representations
- different advantages and disadvantages
- outlook to min-max-trees

Next Lecture
- dynamic bit vectors and succinct trees
- maybe succinct graphs

Advanced Data Structures
- BV
- succ. trees

