Lexicographic Breadth First Search

Lexicographical Order

Lexicographical Order

A sequence of vectors is in *lexicographical order* if, for each vector (a_1, a_2) and (b_1, b_2),

$$(a_1, a_2) \leq (b_1, b_2) \text{ if and only if } a_1 < b_1 \vee (a_1 = b_1 \wedge a_2 \leq b_2).$$

Examples

- Words in a dictionary
- `0815`
  `156`
  `18`
  `22`
  `734`
  `9`

LexBFS

Idea

- Modified BFS
- Each vertex has a label (updated during iteration).
- Next vertex is vertex with lexicographically largest label.

Applications

- Recognition of special graph classes.
- Diameter approximation for special graph classes.

LexBFS – Algorithm

Input: A graph $G = (V, E)$ with $|V| = n$ and a start vertex $v \in V$.

Output: A vertex ordering σ.

Set $\text{label}(v) := \langle n \rangle$.

For each vertex $u \neq v$, set $\text{label}(u) := \langle\,\rangle$.

For $i := 1$ **To** n

  Pick an unvisited vertex u with a lexicographically largest label.

  Set u as visited and $\sigma[i] := u$, i. e., u is the i-th vertex in order σ.

  For Each unvisited $w \in N(u)$

    Append i to $\text{label}(w)$.

How can we implement this in linear time?

Partition Refinement

Idea

- "Data structure" representing a partition of a set.
- It allows refining the partition by splitting its sets into smaller sets.

Refinement

- Given: Family of disjoint sets $\mathcal{S} = \{S_1, S_2, \ldots\}$ and set X.
- Replace each set S_i by $S_i \cap X_i$ and $S_i \setminus X_i$.
- Remove a set if empty.

Implementation

- $\mathcal{S}$ is doubly linked list.
- Constant time function $f\colon X \to \mathcal{S}$ with $f(x) = S_i$ if and only if $x \in S_i$.

LexBFS using Partition Refinement

Input: A graph $G = (V, E)$ with $|V| = n$ and a start vertex $v \in V$.

Output: A vertex ordering σ.

```
Initialise 𝒮 := {{v}, V − v} = {S_1, S_2}.
For i := 1 To n
    Pick first vertex u ∈ S_i, set u as visited, and set σ[i] := u.
    𝒮 := Refine(𝒮, {u})
    𝒮 := Refine(𝒮, { x ∈ N(u) | x is unvisited})
```