Advanced Data Structures

Lecture 04: Succinct Planar Graphs and Range Min-Max Trees
Florian Kurpicz
Recap: Succinct (Dynamic) Graphs

- dynamic bit vector
- dynamic succinct trees
- which was the easiest representation for dynamic trees

Advanced Data Structures

- static/dynamic BV
- static/dynamic succinct trees
Today’s Plan

- preliminaries planar graph
- succinct planar graph representation
- range min-max trees
- project
Planar Graphs (1/2)

Definition: Planar Graph

A graph $G = (V, E)$ is planar, if it
- can be drawn on the plane such that
- no edges cross each other

- drawing (planar) embedding of the graph
- not unique

a graph is planar if it has no minor $\kappa$
- $K_{3,3}$
- $K_5$
Planar Graphs (2/2)

- embedding is defined by order of neighbors
- this defines faces
- must specify outer face

Now Consider Only
- connected planar graphs with embedding,
- multi-edges, and
- self-loops appear twice in list of edges
Definition: Dual Graph

Given an embedding of a planar graph $G$, the dual graph $G^*$ of $G$ has:

- one node for each face of $G$ and
- one edge $e'$ for each edge $e$ in $G$ such that $e'$ crosses $e$ and is incident to the faces separated by $e$

- dual graph is unique for the embedding
- dual graph is planar
Spanning Trees

Definition: Spanning Tree

Given a connected graph $G = (V, E)$, a spanning tree is a tree $T = (V, E')$ with $E' \subseteq E$.

- Consider spanning tree of planar graph and
- its dual graph
- trees can be represented succinctly
Recap: 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

- $ab\ cd\ ef\ g\ \ h\ \ ij\ \ k$
- $(()(()(()()))()(()()))$

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

- $findclose(i) = fwd\_search(i, 0)$
- $findopen(i) = bwd\_search(i, 0)$
- $enclose(i) = bwd\_search(i, 2)$
Succinct Planar Graph: General Idea [Fer+20; Tur84]

- given connected planar graph $G$ and its dual $G^*$
- let $T$ be spanning tree of $G$
- construct complementary spanning tree $T^*$ of $G^*$ using only edges not crossing edges in $T$

- edges are stored in adjacency lists

**Definition: Incidence**

Given a face $f$ and a vertex $v$, an incidence of $f$ in $v$ is a pair of edges $e$, $e'$, such that $v$ is part of $f$ and $e$, $e'$ are incident of $f$ and consecutive in the adjacency list of $v$
Lemma: Graph-Tree-Traversal

Given an embedding of $G$, a spanning tree $T$ of $G$, and its complementary spanning tree $T^*$ of the dual of $G$. When

- traversing $T$ depth-first, starting at any node on the outer face
- processing edges in counter-clockwise order
- (for the root choose an arbitrary incidence of the outer face),

each edge not in $T$ corresponds to the next edge visited in a depth-first traversal of $T^*$.
Traversing the Graph gives Traversal of Trees (2/2)

Proof Graph-Tree-Traversal
- proof by induction
- correct in the beginning
- processed $i$ edges, $(i + 1)$-th edge is $(v, w)$
- if $(v, w)$ is in $T$, nothing changes
- example on the board

Proof Graph-Tree-Traversal
- proof by induction
- correct in the beginning
- processed $i$ edges, $(i + 1)$-th edge is $(v, w)$
- if $(v, w)$ is in not $T$, then
- visit new edge in $T'$
- due to counter-clockwise visiting of nodes in $G$, going deeper in $T^*$
- example on the board

12/20 2022-05-16 Florian Kurpicz | Advanced Data Structures | 04 Succinct Planar Graphs & Range Min-Max Trees Institute of Theoretical Informatics, Algorithm Engineering
Succinct Graphs \((n = |V| \text{ and } m = |E|)\)

- bit vector \(A[0..2m]\) with \(A[i] = 1 \iff \text{the } i\text{-th edge processed is in } T\)
- bit vector \(B[0..2(n - 1)]\) with \(B[i] = "(" \iff \text{i-th time an edge in } T \text{ is processed is the first time that edge is processed}\)
- bit vector \(B^*[0..2(m - n + 1)]\) with \(B^*[i] = "(" \iff \text{i-th time an edge not in } T \text{ is processed is the first time that edge is processed}\)

\[
A = 0110110101110010110100010100
\]

\[
B = ((((()))))((()))((()))
\]

\[
B^* = ()(((()))))((()))
\]
Simple Planar Succinct Graph Operations (1/2)

- \( \text{first}(v) \) return \( i \) such that the first edge processed when visiting \( v \) is processed \( i \)-th during traversal
- \( \text{next}(i) \) return \( j \) such that next edge that is processed when visiting \( v \) by \( i \)-th edge is processed \( j \)-th during traversal
- \( \text{mate}(i) \) return \( j \) such that edge is processed \( i \)-th and \( j \)-th during traversal
- \( \text{vertex}(i) \) return node \( v \) that is visited when processing \( i \)-th edge during traversal
Simple Planar Succinct Graph Operations (2/2)

- All operations work in $O(1)$ time
- Using rank and select queries on $A$
- Using BP representation of $T$ and $T^*$

$A = 01101101011100101100010100$

$B = ((()))(())(())$

$B^* = ()(())(())(())$

$first(0) = 0 \quad mate(0) = 3 \quad vertex(3) = 2$

$next(0) = 1 \quad mate(1) = 9 \quad vertex(9) = 1$

$next(1) = 10 \quad mate(10) = 16 \quad vertex(16) = 4$

$next(10) = 17 \quad mate(17) = 25 \quad vertex(25) = 6$

- Example on the board
Getting the Degree

- while node has next
- increase counter and go to next
- return counter

- running time depends of degree of node
- better running time preferable

- speed up queries using $o(m)$ additional bits
- let $f(m) \in \omega(m)$
- mark in $D[0..m]$ nodes with degree $> f(m)$
  1. at most $m/f(m)$ ones (sparse)
- for these nodes store degree unary in $E[0..2m]$
  1. also sparse
- compressed sparse bit vectors require $o(m)$ space

- degree queries require only $O(f(m))$ time
- example on the board 📚
Lemma: Succinct Planar Graphs

Storing an embedding of a connected planar graph with $m$ edges requires $4m + o(m)$ bits and all nodes incident to a node can be iterated over in (counter-)clockwise order in constant time per edge. Finding the degree of a node can be done in $O(f(m))$ time for any function $f(m) \in \omega(1)$. 
Definition: Range Min-Max Tree

Given a bit vector $B$ of length $n$ and a block size $b$, store for each consecutive block (from $s$ to $e$) of $BV$

- total excess in block: $excess(e) - excess(s - 1)$
- minimum left-to-right excess in block: $\min\{excess(p) - excess(s - 1): p \in [s, e]\}$

and build a binary tree over these blocks, where each node stores the same total information for blocks in all its leaves.

Lemma: Range Min-Max Tree Space

A range min-max tree with block size $b$ for a bit vector of size $n$ requires $n + O((n/b) \log n)$ bits of space.
fwdsearch in a Range Min-Max Tree

- scan block
- if not found traverse tree
- identify block in tree
- scan block

- process \( c \) bits at a time
- first align with next \( c \) bits
- requires \( O(c + b/c) \) time

- going up and down tree in \( O(\log(n/b)) \) time
- scanning last block requires \( O(c + b/c) \) time

by choosing \( b = c \log n \) this requires

\[ O(\log n) \text{ time and } n + O(n/(c \log n)) = n + o(n) \text{ bits space} \]

Improvements

- two level approach
- build range min-max trees for chunks of size \( \Theta(\log^3 n) \)
- \( O(\log \log n) \) query time inside a chunk
- can result in total query time of \( O(\log \log n) \)
Conclusion and Outlook

This Lecture
- succinct planar graphs
- range min-max trees

- no live lecture next week
- video only
- will start half an hour earlier on 30.05. for questions

Next Lecture
- predecessor data structures
- introduction to range minimum queries

Advanced Data Structures
- static/dynamic BV
- static/dynamic succ. trees
- range min-max tree
- succ. graphs
Project

- detailed information on the homepage
- implement dynamic bit vectors and BP
- deadline: 15.07.2022
- present results in 5 minutes on 25.07.2022