Succint Graph Structures
Lecture 4 - Labeling Schemes

Introduction

High-level question: How to represent a graph?

Conventional approach:

- Give the vertices arbitrary $O(\log n)$-bit identifiers.
  Say, a unique num in $[n]$
- Use some usually centralized representation (adjacency matrix/list...)

Vertex names are meaningless

Today: Labeling Schemes: "Meaningful names"

("Distributed representation")

Adjancency Labeling Schemes

Pair of algorithms:

1. Encoder gets a graph $G=(V,E)$, $|V|=n$. should assign a short label $L_G(v)$ to each $v \in V$.

$$G \longrightarrow \boxed{\text{Enc}} \longrightarrow L_G : V \to \{0,1\}^{\ell}$$

ℓ is the label length
want to minimize it

2. Then, Decoder gets two labels $L(u)$, $L(v)$ and should determine if u,v are adjacent in G

$$\begin{matrix} L_G(u) \\ L_G(v) \end{matrix} \longrightarrow \boxed{\text{Dec}} \longrightarrow \begin{cases} 1 & \text{if } (u,v) \in E \\ 0 & \text{if } (u,v) \notin E \end{cases}$$

Encoder	Decoder
- knows the graph	- Doesn't know the graph [but knows what is the encoder alg]
- Constructs labels	- Anwers queries given labels

General adjacency labels

Claim: A adjency labeling scheme for general n-vertex graphs must have length $\ell = \Omega(n)$.

Proof: Let $\mathcal{G}$ - family of graphs with $V = [n]$

Define $\varphi : \mathcal{G} \longrightarrow \{0,1\}^{\ell \cdot n}$

$$\varphi(G) = \left(L_G(1), L_G(2), \dots, L_G(n)\right)$$

Given $\varphi(G)$, can reconstruct G using Decoder

$\Longrightarrow$ φ is 1-1

$\Longrightarrow$ $2^{\ell n} \geq |\mathcal{G}| = 2^{\binom{n}{2}}$

$\Longrightarrow$ $\ell = \Omega(n)$ $\blacksquare$

Exer: Show $O(n)$-bit adjacency labels.

Tree adjacency labels

What if restrict attention to family of trees?

There are n^{n-2} trees on $[n]$ (Cayley's formula)

$\Rightarrow$ label length LB of $\frac{\log(n^{n-2})}{n} = (1-o(1))\log n$

What about UB?

Claim: There is an adj. labeling scheme for trees with label length $2\lceil \log_2 n \rceil$.

Proof: Encoder: gets $T=([n],E)$ and chooses arbitrary root.

- For vertex $i \in [n]$, let $p(i)$ be its parent. (for the root r we can define $p(r)=r$).
- Label of i is $L(i) = (i, p(i))$

Decoder: given $L(i), L(j)$

return "adjancent" $\Longleftrightarrow$ $i=p(j)$ or $j=p(i)$ ∎

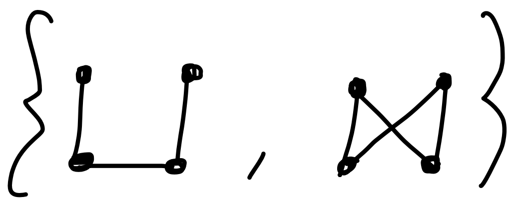
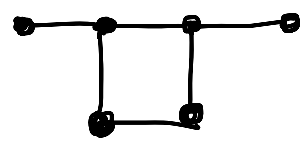
Why pay attention to constants?

Universal graphs

Def: Graph $U=(V,E)$ is universal for graph family $\mathcal{F}$ if:

$\forall G \in \mathcal{F}$, $\exists$ induced subgraph of U that is isomorphic to G.

induced subgraph: choosing some vertex-subset $X \subseteq V$
take all X-to-X edges from U
(can't discard X-to-X edges)

Example: $\mathcal{F} =$  $U =$ 

Thm [Kannan, Naor, Rudich, '88]

Let $\mathcal{F}$ be a graph family.

$\mathcal{F}$ has adj. labeling scheme with length ℓ $\iff$ $\mathcal{F}$ has a universal graph U with 2^ℓ vertices

[that assigns unique labels to the vertices of any $G \in \mathcal{F}$]

The arboricity of planar graphs is 3

Corollaries:

- $O(n^2)$-vertex universal graph for n-vertex trees
- Fact: the edges of a planar graph can be partitioned to 3 forests
  $\Rightarrow$ $4\lceil \log n \rceil$-bit adj. labels (Exercise: why?)
  $\Rightarrow$ $O(n^4)$-vertex universal graph for planar graphs
- $O(n^{a+1})$-vertex universal graph for n-vertex graph of arboricity a

Proof:

($\Rightarrow$) vertices of U are $\{0,1\}^{\ell}$

put edge between $x, y \in \{0,1\}^{\ell}$ iff the decoder outputs "adjacent" when supplied with x, y as input.

For $G \in F$, the subgraph induced on $\{L_G(v) \mid v \in V(G)\}$ is isom.

($\Leftarrow$) Encoder: given $G \in F$

- find isomorphic induced subgraph G' of U
- label each vertex of G by the name of its corresponding vertex from G'

Decoder: given $L_G(u)$, $L_G(v)$

- return adjacent $\Leftrightarrow$ vertices $x = L_G(u)$, $y = L_G(v)$ are adjacent in U.

Note: Decoder knows U (part of the scheme) doesn't know G!

Note: G' being induced subgraph is crucial for correctness!

x, y adj. in U $\underset{\text{$G'$ induced subgraph}}{\Leftrightarrow}$ x, y adj. in G' $\underset{\text{$L_G$ isomorphism}}{\Leftrightarrow}$ u, v adj. in G

∎

Informative Labeling Schemes

Suppose we have graph $G = (V, E)$ and some function of interest $f_G : V^k \to \{0,1\}^*$

<u>Examples</u>: $k = 2$

- distance: $f_G(u,v) = dist_G(u,v)$
- adjacency: $f_G(u,v) = \begin{cases} 1 & (u,v) \in E \\ 0 & (u,v) \notin E \end{cases}$
- flow: $f_G(u,v) :=$ max # of edge-disjoint u-v paths

  [:= min # edges in a uv-cut]

Labeling scheme for function f:

<u>Encoder</u>:

$$G,\ f_G \to \boxed{Enc} \to L_{G,f} : V \to \{0,1\}^\ell$$

<u>Decoder</u>:

$$L_{G,f}(v_1), \ldots, L_{G,f}(v_k) \to \boxed{Dec} \to f(v_1, \ldots, v_k)$$

Decoder doesn't know G

Distance labeling in trees

Thm: There is an $O(\log^2 n)$-bit distance labeling scheme for trees

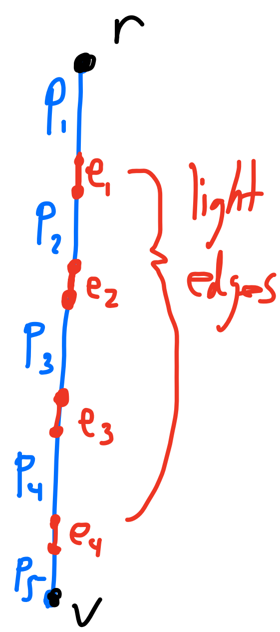
Reminder: Heavy-Light Decomp. of rooted tree T

- The heavy child $h(v)$ of vertex v is the child with most vertices in its subtree
- Edges of the form $(v, h(v))$ are heavy edges. The remaining edges are light edges
- Key property: every (simple) tree path contains only $O(\log n)$ light edges

Encoder: for vertex v, let $\pi(r, v)$ be the path from the root r to v.

- Partition it to segments of only heavy edges with interleaving light edges

$$\pi(r,v) = P_1 \circ e_1 \circ P_2 \circ e_2 \circ P_3 \circ e_3 \circ \cdots \circ P_k$$



The label $L_T(v)$ is the "compressed" $\pi(r, v)$:

$\rightarrow$ each P_i is replaced with its length $|P_i|$

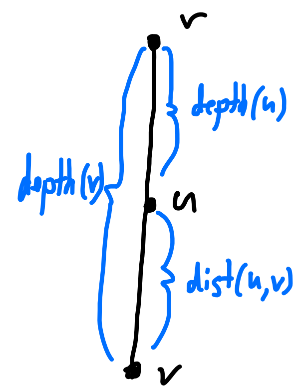
$$L_T(v) = \left(|P_1|, e_1, |P_2|, e_2, \dots, |P_k| \right)$$

By "key property", $k = O(\log n) \Rightarrow$ length $O(\log^2 n)$

Decoder: given $L_T(u)$, $L_T(v)$

Obs 1: can compute $depth(u) = |\pi(r,u)|$ and $depth(v) = |\pi(r,v)|$

Obs 2: u is an ancestor of v iff



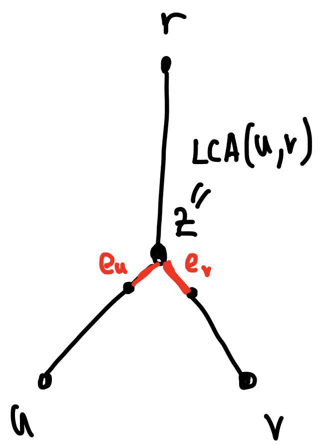
(*) $depth(u) \le depth(v)$, and

(*) $\{$light edges in $\pi(r,u)\} \subseteq \{$light edges in $\pi(r,v)\}$

→ Can detect if u,v have ancestry relations

→ if they do, return $|depth(v) - depth(u)|$

Assume now that u,v <u>don't</u> have ancestry relations



⇒

① $dist(u,v) = depth(u) + depth(v) - 2 \cdot \underbrace{depth(z)}$

depth of LCA(u,v)
also called SepLevel(u,v)

② at least one of e_u, e_v is light!

Say its e_u

→ Can detect e_u in $L_T(u)$ (look for first point of difference from $L_T(v)$)

→ Can compute the length of the common prefix of $\pi(r,v)$ and $\pi(r,u)$:

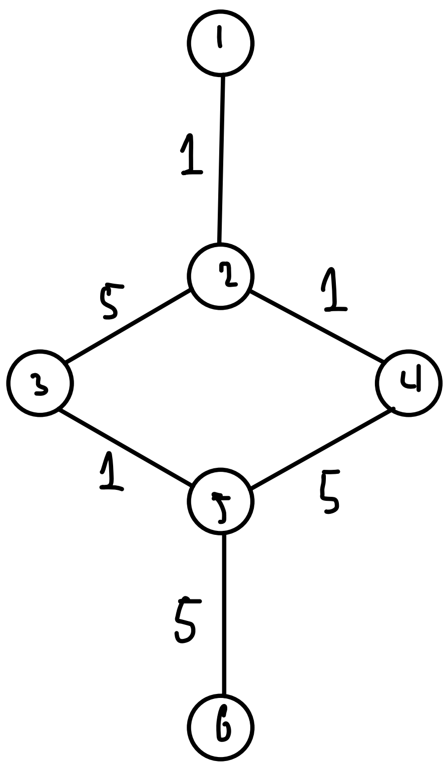
This is SepLevel(u,v)!

Corollary: An $O(\log^2 n)$-bit labeling scheme for separation level in trees

Flow labeling scheme

We now focus on n-vertex graphs G with integral edge capacities (multiplicities) in $\{1, 2, \dots, \bar{W}\}$

Recall: the flow between two vertices u, v
$$= \max \# \text{ edge-disj. } u\text{-}v \text{ paths}$$



	1	2	3	4	5	6
1	∞	1	1	1	1	1
2		∞	6	2	2	2
3			∞	2	2	2
4				∞	6	5
5		Symmetric			∞	5
6						∞

Thm: There is an $O(\log^2(n\bar{W}))$ labeling scheme for flow.

Lemma: For any $k \geq 0$,

$$R_k = \{(x,y) \in V \times V \mid \text{flow}_G(x,y) \geq k\}$$

is an equivalence relation.

Proof: Reflexive + Symmetric: trivial.

Transitive: suppose $\text{flow}(x,y) \geq k$ and $\text{flow}(y,z) \geq k$.

By way of contradiction, assume $\text{flow}_G(x,z) < k$.

By min-cut max-flow thm, there is a cut $(S, V \setminus S)$ of capacity $< k$ with $x \in S$, $z \in V \setminus S$.

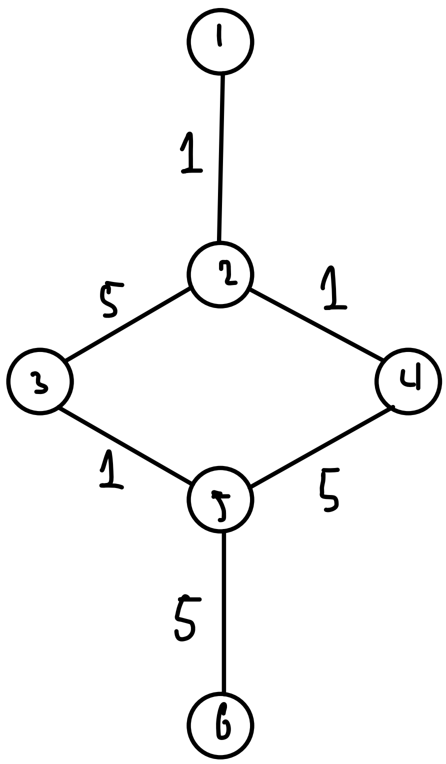
- if $y \in S$, then $(S, V \setminus S)$ is an yz-cut, so by min-cut max-flow thm, $\text{flow}(y,z) < k$ – Contradiction.
- the case $y \in V \setminus S$ is similar.. ∎

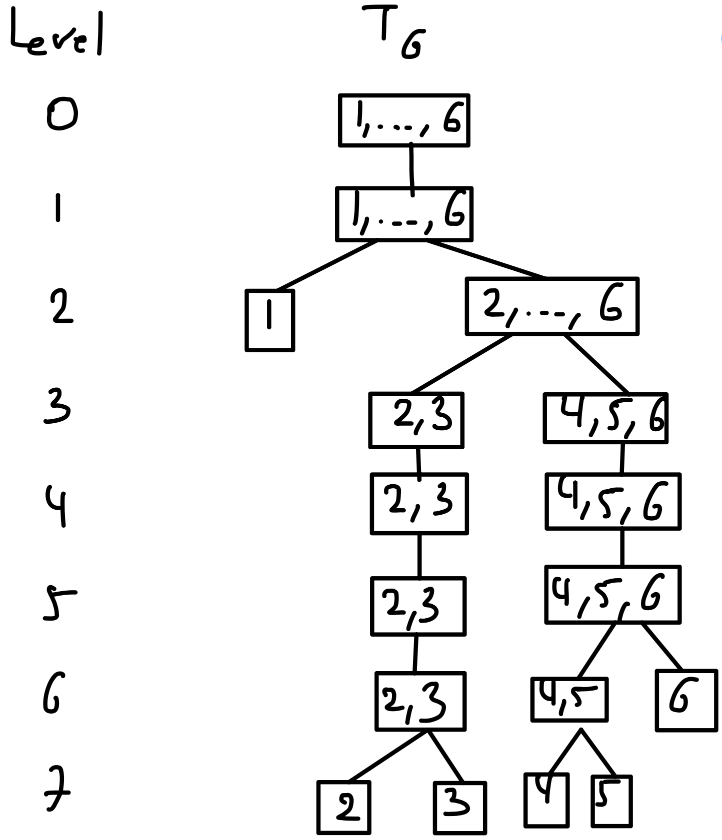
Let $\mathcal{C}_k$ be the equivalence classes of R_k

Note that $\mathcal{C}_{k+1}$ is a <u>refinement</u> of $\mathcal{C}_k$

Construct tree T_G:

- nodes in level k are the classes in $\mathcal{C}_k$
- parent of node $C \in \mathcal{C}_{k+1}$ is $C' \in \mathcal{C}_k$ s.t. $C \subseteq C'$.
- leaves correspond to singletons
- only $O(n^2 W)$ levels (as we cannot have larger flows)





<u>Obs</u>: $\mathrm{flow}_G(x,y) = \mathrm{SepLevel}_{T_G}(\mathrm{leaf}_{T_G}(x), \mathrm{leaf}_{T_G}(y))$

$\Rightarrow$ Can use labels $L_G(x) = L_{T_G}(\mathrm{leaf}_{T_G}(x))$

where $L_{T_G}(\cdot)$ are SepLevel labels!

<u>length</u>: $O(\log^2 |V(T_G)|) = O(\log^2(nw))$

Distance Labeling and Separators

Def: A separator of a graph $G=(V,E)$, $|V|=n$ is a vertex subset $S \subseteq V$ such that removing S from G breaks it into connected components each of size $\le \frac{2}{3}n$.

Claim: Every tree has a separator of size 1.

Proof: The following alg finds the separator:

```
v ← the root
while h(v) has > n/2 vertices in its subtree:
    v ← h(v)
return v
```

heavy child

Thm (Planar Separator): Every planar graph on n vertices has a separator of size $O(\sqrt{n})$.

Def: An family of graphs $\mathcal{F}$ has $r(n)$-separators if every $G \in \mathcal{F}$ with n vertices has a separator of size $\le r(n)$.

- Trees: $r(n)=1$
- planar graphs: $r(n) = O(\sqrt{n})$

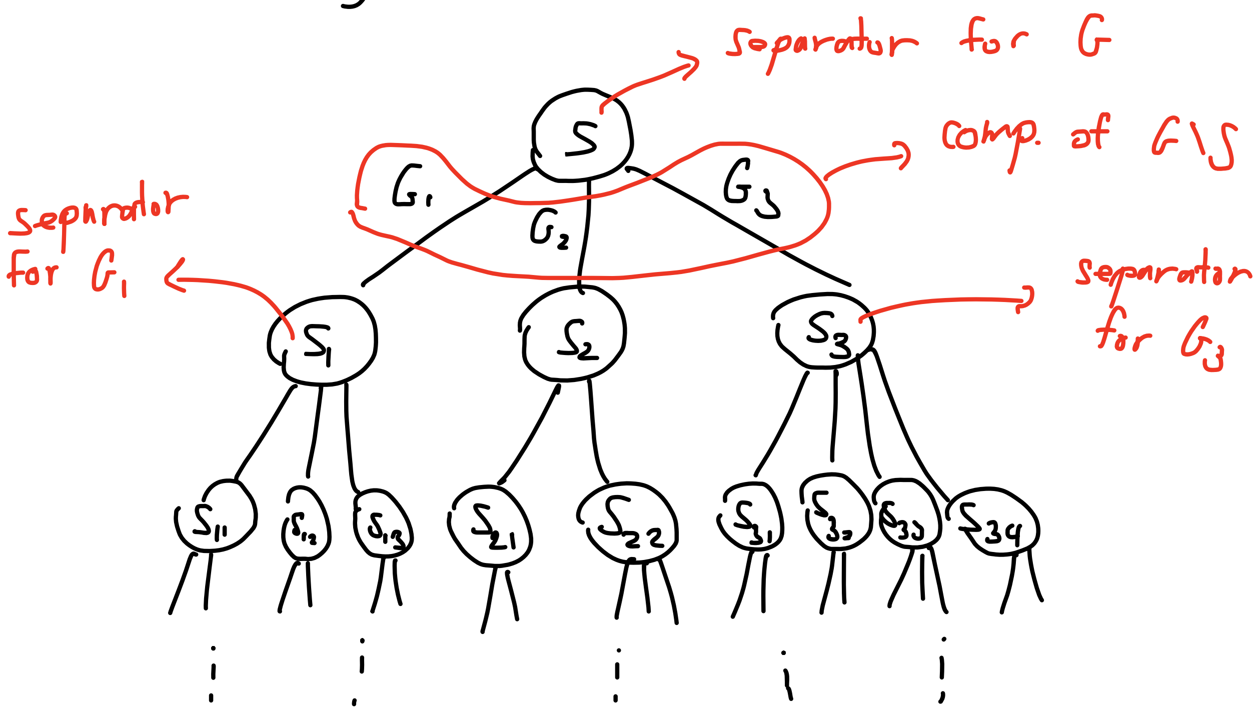
Thm: Let $\mathcal{F}$ be a subgraph-closed family with $r(n)$ separators. Then $\mathcal{F}$ has a distance labeling scheme with length

$$O\left(R(n)\log n + \log^2 n\right), \quad R(n) = \sum_{i=0}^{\log_{3/2}(n)} r\left(\left(\tfrac{2}{3}\right)^i n\right)$$

$R(n) = 1$ for $\mathcal{F} = \{\text{trees}\}$
$R(n) = O(\sqrt{n})$ for $\mathcal{F} = \{\text{planar graphs}\}$

Idea: construct a "tree of separators":

- root node contains separator S for G
- create an edge for each connected comp. of $G \setminus S$
- recursively apply tree construction on each such component



(*) Nodes partition V;
Denote $\text{node}(v)$ for the one containing $v \in V$

Encoding: label $L(v)$ of v contains:

1. SepLevel label of node(v) : $O(\log^2 n)$ bits
2. Let $S^{(0)}, S^{(1)}, \dots, S^{(k)}$ be the root-to-node(v) path

   store $dist_G(v,s)$ for every $s \in \bigcup_{i=1}^{k} S^{(i)}$

   $|S^{(i)}| \leq r\left(\left(\frac{2}{3}\right)^i n\right) \Rightarrow O(R(n) \log n)$ bits

Decoding: given labels $L(u)$, $L(v)$

- if v appears in the ancestors of node(u),

  then $dist(u,v)$ is found in $L(u)$

- if u ———— 〃 ———— node(v)

  ———— 〃 ———— $L(v)$

- Otherwise: Let $q = \text{SepLevel}(\text{node}(u), \text{node}(v))$

  $S^{(0)}, \dots, S^{(q)}$ - common ancestors of node(u), node(v)

  By construction, u and v are in different components of $G \setminus \bigcup_{i=0}^{q} S^{(i)}$

  $\Rightarrow$ any u-v path in G goes through $\bigcup_{i=0}^{q} S^{(i)}$

  $\Rightarrow$ return $\min \left\{ \underbrace{dist(u,s)}_{\text{in } L(u)} + \underbrace{dist(s,v)}_{\text{in } L(v)} \;\middle|\; s \in \bigcup_{i=0}^{q} S^{(i)} \right\}$