Chapter 13

Labeling Schemes

Imagine you want to repeatedly query a huge graph, e.g., a social or a road network. For example, you might need to find out whether two nodes are connected, or what the distance between two nodes is. Since the graph is so large, you distribute it among multiple servers in your data center.

13.1 Adjacency

Theorem 13.1. *It is possible to assign labels with at most* $2 \log n$ *bits to nodes in a tree so that for every pair* u, v *of nodes, it is easy to tell whether* u *is adjacent to* v *by just looking at* u *and* v*'s labels.*

Proof. Choose a root in the tree arbitrarily so that every non-root node has a parent. The label of each node u consists of two parts: The ID of u (from 1 to n), and the ID of u's parent (or nothing if u is the root). □

Remarks:

- What we have constructed above is called a *labeling scheme*, more precisely a labeling scheme for adjacency in trees. Formally, a labeling scheme is defined as follows.

Definition 13.2. *A* labeling scheme *consists of an* encoder e *and a* decoder d. *The encoder* e *assigns to each node* v *a label* $e(v)$. *The decoder* d *receives the labels of the nodes in question and returns an answer to some query. The largest size (in bits) of a label assigned to a node is called the* label size *of the labeling scheme.*

Remarks:

- In Theorem 13.1, the decoder receives two node labels $e(u)$ and $e(v)$, and its answer is YES or NO, depending on whether u and v are adjacent or not. The label size is $2 \log n$.

- The label size is the complexity measure we are going to focus on in this chapter. The run-time of the encoder and the decoder are two other complexity measures that are studied in the literature.

- There is an interesting connection between labeling schemes for adjacency and so-called *induced-universal graphs*: Let $\mathcal{F}$ be a family of graphs. The graph $U(n)$ is called *n-induced-universal for $\mathcal{F}$* if all $G \in \mathcal{F}$ with at most n nodes appear as node-induced subgraphs in $U(n)$. (A node-induced subgraph of $U(n) = (V, E)$ is any graph that can be obtained by taking a subset V' of V and all edges from E which have both endpoints in V'.)

- In the movie Good Will Hunting, the big open question was to find all graphs of the family of homeomorphically irreducible (non-isomorphic, no node with degree 2) trees with 10 nodes, $\mathcal{T}_{10}$. What is the smallest (10-)induced-universal graph for $\mathcal{T}_{10}$?

- For a graph family $\mathcal{F}$, if there is a labeling scheme for adjacency with distinct node labels and label size $f(n)$, then there is an n-induced-universal graph for $\mathcal{F}$ of size at most $2^{f(n)}$. Since the size of $U(n)$ is exponential in f it is interesting to study the label size carefully: If f is $\log n$, the size of $U(n)$ is n, whereas if f is $2\log n$ the size of $U(n)$ becomes n^2!

- What about adjacency in general graphs?

Theorem 13.3. *There is a labeling scheme for adjacency in general graphs with the label size $\frac{n}{2} + \log n$.*

Proof. Let the label of each node comprise:

- A distinct ID i between 0 and $n-1$ ($\lceil \log n \rceil$ bits);
- For $j = i+1 \mod n, \ldots, i + \lfloor \frac{n}{2} \rfloor \mod n$, one bit indicating adjacency to the node with the ID j.

Then, for any pair of nodes, one of the node labels will contain the bit indicating adjacency. The bit can be found based on the ID's. □

Theorem 13.4. *Any labeling scheme for adjacency in general graphs has a label size of at least $\frac{n-1}{2} - 3$ bits.*

Proof. Let $\mathcal{G}_n$ denote the family of graphs with n nodes. A graph in $\mathcal{G}_n$ can have at most $\binom{n}{2}$ edges, so $|\mathcal{G}_n| = 2^{\binom{n}{2}}$.

Assume there is a labeling scheme for adjacency in graphs from $\mathcal{G}_n$ with label size s. For simplicity, let us assume that labels have size *exactly* s bits.

For a graph $G \in \mathcal{G}_n$ with vertex set V_G, write $S(G) = \{\{e(v) \mid v \in V_G\}\}$ for the multiset of labels produced by the encoder for G. Observe that $S(G)$ uniquely determines G up to isomorphism (i.e., up to the ordering of the vertices): the decoder called with every pair of values in $S(G)$ should correctly reconstruct whether there is an edge between the corresponding vertices. Consequently, given two non-isomorphic graphs $G, H \in \mathcal{G}_n$ it should hold that $S(G) \neq S(H)$.

Construct $\mathcal{G}'_n \subseteq \mathcal{G}_n$ by including exactly one graph for each possible graph 'shape' (i.e., one member for each isomorphism class). Because each graph $G \in \mathcal{G}_n$ is isomorphic to at most $n!$ other graphs, it follows that $|\mathcal{G}'_n| \geq |\mathcal{G}_n|/n! = 2^{\binom{n}{2}}/n!$. From the previous, we know that the function S is injective on $\mathcal{G}'_n$, so its codomain must have size at least $|\mathcal{G}'_n| \geq 2^{\binom{n}{2}}/n!$. However, the size of the codomain is bounded from above by the number of multisets with n values from the set $\{0, \ldots, 2^s - 1\}$. The number of such multisets can be shown to be:[1]

$$\binom{2^s + n - 1}{n} = \frac{2^s \cdot (2^s + 1) \cdot \ldots \cdot (2^s + n - 1)}{n!}$$

Hence, we must have that $2^{\binom{n}{2}}/n!$ is at most the right-hand side of the above. Multiplying both sides by $n!$, we get that:

$$2^{\binom{n}{2}} \leq 2^s \cdot (2^s + 1) \cdot \ldots \cdot (2^s + n - 1) \leq (2^s + n - 1)^n$$

Since $\binom{n}{2} = n(n-1)/2$, taking n-th roots on both sides gives $2^{(n-1)/2} \leq 2^s + n - 1$, which is the same as $2^s \geq 2^{(n-1)/2} - (n-1)$. For $n \leq 5$, the right-hand side is non-positive, so we get the trivial bound $s \geq 0$, but we need not worry about small values of n. For $n \geq 6$, rewriting and taking logs:

$$2^s \geq 2^{(n-1)/2}\left(1 - \frac{n-1}{2^{(n-1)/2}}\right) \iff s \geq \frac{n-1}{2} + \log_2\left(1 - \frac{n-1}{2^{(n-1)/2}}\right)$$

One can check that the logarithm is strictly increasing and hence attains its minimum at $n = 6$. Its value there is around -3.106, from which we get that $s \geq (n-1)/2 - 3.2$. Since s is an integer, this can be upgraded for free to $s \geq (n-1)/2 - 3$. Note that this bound also holds (trivially) for $n \leq 5$, so we get our bound in general. □

Remarks:

- The lower bound for general graphs is a bit discouraging; we wanted to use labeling schemes for queries on large graphs!
- The situation is less dire if the graph is not arbitrary. For instance, in degree-bounded graphs, in planar graphs, and in trees, the bounds change to $\Theta(\log n)$ bits.
- What about other queries, e.g., distance?
- Next, we will focus on rooted trees.

13.2 Rooted Trees

Theorem 13.5. *There is a $2\log n$ labeling scheme for ancestry, i.e., for two nodes u and v, find out if u is an ancestor of v in the rooted tree T.*

Proof. Traverse the tree with a depth-first search, and consider the obtained pre-order traversal of the nodes, i.e., enumerate the nodes in the order in which they are first visited. For a node u, denote by $l(u)$ its index in the traversal. Our encoder assigns labels $e(u) = (l(u), r(u))$ to each node u, where $r(u)$ is the largest value $l(v)$ that appears at any node v in the sub-tree rooted at u. With the labels assigned in this manner, we can find out whether u is an ancestor of v by checking if $l(v)$ is contained in the interval $(l(u), r(u)]$. □

Algorithm 13.6 Naïve-Distance-Labeling(T)

```
Let l be the (already assigned) label of the root r of T
Let T_1, ..., T_δ be the sub-trees rooted at each of the δ children of r
for i = 1, ..., δ do
    The root of T_i gets the label obtained by appending i to l
    Naïve-Distance-Labeling(T_i)
end for
```

Theorem 13.7. *There is an $\mathcal{O}(n \log n)$ labeling scheme for distance in trees.*

Proof. Label the root of T with the empty label and then apply the encoder algorithm Naïve-Distance-Labeling(T) to label the rest of the tree. The encoder assigns to every node v a sequence $(l_1, l_2, \ldots)$. The length of a sequence $e(v)$ is at most $n-1$, and each entry in the sequence requires at most $\log n$ bits. A label $(l_1, \ldots, l_k)$ of a node v corresponds to a path from r to v in T, and the nodes on the path are labeled $(l_1), (l_1, l_2), (l_1, l_2, l_3)$, etc. The distance between u and v in T is obtained by reconstructing the paths from $e(u)$ and $e(v)$. □

Remarks:

- We can assign the labels more carefully to obtain a smaller label size. For that, we use the following so-called *heavy-light decomposition*.

Algorithm 13.8 Heavy-Light-Decomposition(T)

```
Node r is the root of T
Let T_1, ..., T_δ be the sub-trees rooted at each of the δ children of r
Let T_max be a largest tree in {T_1, ..., T_δ} in terms of number of nodes
Mark the edge (r, T_max) as heavy
Mark all edges to other children of r as light
Assign the names 1, ..., δ − 1 to the light edges of r
for i = 1, ..., δ do
    Heavy-Light-Decomposition(T_i)
end for
```

Theorem 13.9. *There is an $\mathcal{O}(\log^2 n)$ labeling scheme for distance in trees.*

Proof. For our proof, use Heavy-Light-Decomposition(T) to partition T's edges into heavy and light edges. All heavy edges form a collection of paths, called the *heavy paths*. Moreover, every node is reachable from the root through a sequence of heavy paths connected with light edges. Instead of storing the whole path to reach a node, we only store the information about heavy paths and light edges that were taken to reach a node from the root.

For instance, if node u can be reached by first using 2 heavy edges, then the 7^{th} light edge, then 3 heavy edges, and then the light edges 1 and 4, then we assign to v the label $(\mathbf{2}, 7, \mathbf{3}, 1, 4)$. We wrote in bold font the counts of heavy edges. Implementation-wise, we would tag each value with an extra bit to represent whether it's bold or not. For any node u, the path $p(u)$ from the

[1] `https://en.wikipedia.org/wiki/Multiset#Counting_multisets`

root to u is now specified by the label. The distance between any two nodes can be computed using the paths.

Since every parent has at most $n-1$ children, the name of a light edge uses at most $\log n$ bits. The size (number of nodes in the sub-tree) of a light child is less than half the size of its parent, so all paths use less than $\log n$ light edges. Between any two light edges, there could be a heavy path, so we can have up to $\log n$ heavy paths in a label. The length of such a heavy path can be described with $\log n$ bits as well since no heavy path has more than n nodes. Altogether we therefore need at most $\mathcal{O}(\log^2 n)$ bits. □

Remarks:

- One can show that any labeling scheme for distance in trees needs to use labels of size at least $\Omega(\log^2 n)$.
- The distance encoder from Theorem 13.9 also supports decoders for other queries. To check for ancestry, it therefore suffices to check if $p(u)$ is a prefix of $p(v)$ or vice versa.
- The lowest common ancestor is the node w that is on $p(u)$ and $p(v)$ and on the shortest path between u and v; the separation level of u and v is the distance of w to the root.
- Two nodes are siblings if their distance is 2 but they are not ancestors.
- The heavy-light decomposition can be used to shave off a few bits in other labeling schemes, e.g., ancestry or adjacency.

13.3 Road Networks

Labeling schemes are used to quickly find shortest paths in road networks.

Remarks:

- A naïve approach is to store at every node u the shortest paths to all other nodes v. This requires an impractical amount of memory. For example, the road network for Western Europe has 18 million nodes and 44 million directed edges, and the USA road network has 24 million nodes and 58 million directed edges.
- What if we only store the next node on the shortest path to all targets? In a worst case this stills requires $\Omega(n)$ bits per node. Moreover, answering a single query takes many invocations of the decoder.
- For simplicity, let us focus on answering distance queries only. Even if we only want to know the distance, storing the full table of n^2 distances costs more than 1000TB, too much for storing it in RAM.
- The idea for the encoder is to compute a set S of *hub* nodes that lie on many shortest paths. We then store at each node u only the distance to the hub nodes that appear on shortest paths originating or ending in u.

- Given two labels $e(u)$ and $e(v)$, let $H(u, v)$ denote the set of hub nodes that appear in both labels. The decoder now simply returns $d(u, v) = \min\{\mathrm{dist}(u, h) + \mathrm{dist}(h, v) : h \in H(u, v)\}$, all of which can be computed from the two labels.
- The key in finding a good labeling scheme now lies in finding good hub nodes.

Algorithm 13.10 Naïve-Hub-Labeling(G)

```
Let P be the set of all n^2 shortest paths
while P ≠ ∅ do
  Let h be a node which is on a maximum number of paths in P
  for all paths p = (u, ..., v) ∈ P do
    if h is on p then
      Add h with the distance dist(u, h) to the label of u
      Add h with the distance dist(h, v) to the label of v
      Remove p from P
    end if
  end for
end while
```

Remarks:

- Unfortunately, algorithm 13.10 takes a prohibitively long time to compute.
- Another approach computes the set S as follows. The encoder (Algorithm 13.11) first constructs so-called *shortest path covers*. The node set S_i is a shortest path cover if S_i contains a node on every shortest path of length between 2^{i-1} and 2^i. At node v only the hub nodes in S_i that are within the ball of radius 2^i around v (denoted by $B(v, 2^i)$) are stored.

Algorithm 13.11 Hub-Labeling(G)

```
for i = 1, ..., log D do
  Compute the shortest path cover S_i
end for
for all v ∈ V do
  Let F_i(v) be the set S_i ∩ B(v, 2^i)
  Let F(v) be the set F_1(v) ∪ F_2(v) ∪ ...
  The label of v consists of the nodes in F(v), with their distance to v
end for
```

Remarks:

- The size of the shortest path covers will determine how space efficient the solution will be. It turns out that real-world networks allow for small shortest path covers: The parameter h, the so-called *highway dimension* of G, is defined as $h = \max_{i,v} F_i(v)$, and h is conjectured to be small for road networks.

- Computing S_i with a minimal number of hubs is NP-hard, but one can compute a $\mathcal{O}(\log n)$ approximation of S_i in polynomial time. Consequently, the label size is at most $\mathcal{O}(h \log n \log D)$. By ordering the nodes in each label by their ID, the decoder can scan through both node lists in parallel in time $\mathcal{O}(h \log n \log D)$.

- While this approach yields good theoretical bounds, the encoder is *still* too slow in practice. Therefore, before computing the shortest path covers, the graph is contracted by introducing *shortcuts* first.

- Based on this approach a distance query on a continent-sized road network can be answered in less that 1μs on current hardware, orders of magnitude faster than a single random disk access. Storing all the labels requires roughly 20 GB of RAM.

- The method can be extended to support shortest path queries, e.g., by storing the path to/from the hub nodes, or by recursively querying for nodes that lie on the shortest path to the hub.

Chapter Notes

Adjacency labelings were first studied by Breuer and Folkman [BF67]. A $\log n + \mathcal{O}(\log^* n)$ upper bound for trees is due to [AR02] using a clustering technique. In contrast, it was shown that for general graphs, the size of universal graphs is at least $2^{(n-1)/2}$! Since graphs of arboricity d can be decomposed into d forests [NW61], the labeling scheme from [AR02] can be used to label graphs of arboricity d with $d \log n + \mathcal{O}(\log n)$ bit labels. For a thorough survey on labeling schemes for rooted trees please check [AHR].

Universal graphs were studied already by Ackermann [Ack37], and later by Erdős, Rényi, and Rado [ER63, Rad64]. The connection between labeling schemes and universal graphs [KNR88] was investigated thoroughly. Our adjacency lower bound follows the presentation in [AKTZ14], which also summarizes recent results in this field of research.

Distance labeling schemes were first studied by Peleg [Pel00]. The notion of highway dimension was introduced by [AFGW10] in an attempt to explain the good performance of many heuristics to speed up shortest path computations, e.g., Transit Node Routing [BFSS07]. Their suggestions to modify the SHARC heuristic [BD08] lead to the hub labeling scheme and were implemented and evaluated [ADGW11], and later refined [DGSW14]. A $\Omega(n)$ label size lower bound for routing (shortest paths) with stretch smaller than 3 is due to [GG01].

Bibliography

[Ack37] Wilhelm Ackermann. Die Widerspruchsfreiheit der allgemeinen Mengenlehre. *Mathematische Annalen*, 114(1):305–315, 1937.

[ADGW11] Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato Fonseca F. Werneck. A hub-based labeling algorithm for shortest paths in road networks. In *SEA*, 2011.

[AFGW10] Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato Fonseca F. Werneck. Highway dimension, shortest paths, and provably efficient algorithms. In *SODA*, 2010.

[AHR] Stephen Alstrup, Esben Bistrup Halvorsen, and Noy Rotbart. A survey on labeling schemes for trees. To appear.

[AKTZ14] Stephen Alstrup, Haim Kaplan, Mikkel Thorup, and Uri Zwick. Adjacency labeling schemes and induced-universal graphs. *CoRR*, abs/1404.3391, 2014.

[AR02] Stephen Alstrup and Theis Rauhe. Small induced-universal graphs and compact implicit graph representations. In *FOCS*, 2002.

[BD08] Reinhard Bauer and Daniel Delling. SHARC: fast and robust unidirectional routing. In *ALENEX*, 2008.

[BF67] Melvin A Breuer and Jon Folkman. An unexpected result in coding the vertices of a graph. *Journal of Mathematical Analysis and Applications*, 20(3):583 – 600, 1967.

[BFSS07] Holger Bast, Stefan Funke, Peter Sanders, and Dominik Schultes. Fast routing in road networks with transit nodes. *Science*, 316(5824):566, 2007.

[DGSW14] Daniel Delling, Andrew V. Goldberg, Ruslan Savchenko, and Renato F. Werneck. Hub labels: Theory and practice. In *SEA*, 2014.

[ER63] P. Erdős and A. Rényi. Asymmetric graphs. *Acta Mathematica Academiae Scientiarum Hungarica*, 14(3-4):295–315, 1963.

[GG01] Cyril Gavoille and Marc Gengler. Space-efficiency for routing schemes of stretch factor three. *J. Parallel Distrib. Comput.*, 61(5):679–687, 2001.

[KNR88] Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. In *STOC*, 1988.

[NW61] C. St. J. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *J. London Math. Soc.*, 36:445–450, 1961.

[Pel00] David Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory*, 33(3):167–176, 2000.

[Rad64] Richard Rado. Universal graphs and universal functions. *Acta Arith.*, 9:331–340, 1964.