CS254 Algorithmic Graph Theory - Lecture Notes

Based on lectures by Prof. Maxim Sviridenko
Typeset by Alex J. Best

April 30, 2013

Chapter 1

Basics

1.1 Graphs

Lecture 1

Definition. A graph G is an ordered pair $V(G), E(G))$ consisting of a set $V(G)$ of vertices and a set $E(g)$ of edges. Each edge $e \in E(G)$ is associated with a pair of vertices $\{u, v\}$. We will say that u, v are endpoints of e. We also say u, v are incident to e and vice versa.

Definition. Two vertices which are incident with a common edge are adjacent (or neighbours). Two or more edges with the same pair of endpoints are said to be parallel.

Definition. A simple graph is one without parallel edges.

Notation. By convention, G will denote a graph, n and m will be the number of vertices $|V(G)|$ and the number of edges $|E(G)|$ respectively.

1.2 Special graphs

Definition. A complete graph is a simple graph in which every two vertices are adjacent.
Such a graph on n vertices is denoted K_n.

We can see that $|E(K_n)| = \frac{n(n-1)}{2}$.

Definition. A graph $G = (V, E)$ is called empty if $E = \emptyset$.

Definition. A graph is called bipartite is its vertex set can be partitioned into two subsets X and Y so that every edge has one endpoint in X and one in Y.

Example. The set of all (monogamous) people can be represented by a graph with edges joining married couples, this graph is bipartite.

We will call X, Y the parts of the graph. So a bipartite graph is often denoted as $G = (X, Y, E)$.

Notation. We write $K_{m,n}$ for the complete bipartite graph with $|X| = m$, $|Y| = n$ and E containing all edges between X and Y.

Definition. The graph $K_{1,n}$ is often called a star.

Definition. A path is a simple graph whose vertices can be rearranges into a linear sequence in such a wayy that two adjacent vertices are always consecutive in the sequence and vice versa, non-adjacent vertices are non-consecutive, such a graph on n vertices is often denoted P_n.

Definition. A cycle on n vertices is a graph defined in a similar way but a cyclic sequence of vertices.

Notation. The length of a path or cycle is the number of its edges. A path or cycle of length k is often called a k-path of k-cycle.

1.2.1 Adjacency and incidence matrices

Definition. The incidence matrix of $G = (V, E)$ when $|V| = m$ and $|E| = n$ is the $m \times n$ matrix $M_G = (m_{ve})$ where m_{ve} is the number of times that vertex v and e are incident ($m_{ve} \in \{0, 1\}$).

Example.

Definition. The adjacency matrix of $G = (V, E)$ when $|V| = m$ is the $m \times m$ matrix $A_G = (a_{uv})$ where m_{uv} is the number of edges joining u and v.

Example.

$$= A_{K_3}$$

Definition. An adjacency list is a list $(N(v), v \in V)$ used to represent matrices, $N(v)$ is the set of neighbours of v in arbitary order.

1.2.2 Adjacency and incidence matrices

Definition. The degree of a vertex v in a graph G is denoted $d_G(v)$ or d_v and is the number of edges of G incident with v.

Theorem (Handshake lemma). *For any graph*

$$\sum_{v \in V} d_v = 2|E| \tag{1.1}$$

Theorem. *In any graph, the number of vertices of odd degree is even.*

Proof. Consider the equation 1.1 module 2. We have degree of each vertex $d_v \equiv 1$ if d_v is odd, or 0 is d_v is even. Therefore the left hand side of 1.1 is congruent to the number of vertices of odd degree and the RHS is 0. Therefore the number of vertices of odd degree is congruent to 0 mod 2. $\square$

Definition. A graph G is called k=regular is $d_G(v) = k$ for all $v \in V(G)$. A graph is regular if it is k-regular for some k.

Example. K_n is $(n-1)$-regular.
C_n is 2-regular.
P_n is not regular.

Lecture 2

Theorem. *Let $G = (X, Y, E)$ be a bipartite graph without isolated vertices s.t. $d_x \geq d_y$ for all $x \in X$ and $y \in Y$ s.t. $(x, y) \in E$. Then $|X| \leq |Y|$ with equality iff $d_x = d_y \forall (x, y) \in E$.*

Proof. Take the adjacency matrix B. Define the matrix B' by dividing each row of B correspondiong to vertex u by d_u. consider one column sum corresponding to $y \in Y$.

$$\sum_{x \in X} b'_{xy} = \sum_{(x,y) \in E} \frac{1}{d_x} = \frac{d_y}{d_x} \leq 1$$

Therefore $|X| = \sum_{x \in X} \sum_{y \in Y, (x,y) \in E} \frac{1}{d_x}$

$$= \sum_{x \in X, y \in Y} \sum_{(x,y) \in E} \frac{1}{d_x} \leq \sum_{x \in X, y \in Y}^{(x,y) \in E} \frac{1}{d_y}$$

$$= \sum_{y \in Y} \sum_{x \in X, (x,y) \in E} \frac{1}{d_y} = |Y|$$

If $|X| = |Y|$, then the middle inequality must be an inequality implying that $d_x = d_y$ for all $(x, y) \in E$. $\square$

1.2.3 Isomorphisms

Definition. Two graphs G and H are isomorphic, written $G \cong H$, if there is a bijection $\Theta \colon V(G) \to V(H)$ such that $(u, v) \in E$ iff $(\Theta(u), \Theta(v)) \in E(H)$.

Example.

Remark.
- The mapping Θ is not unique.
- Isomorphic graphs have the same number of vertices and edges.
- Up to isomorphism there is onlt one complete graph on n vertices denoted K_n. So K_n is a class of graphs that are all isomorphic to each other.
- The same as above goes for each of $K_{m,n},\ P_n,\ C_n$.

1.2.4 Directed graphs

Definitions. In a graph $G = (V, E)$ a *walk* from a starting vertex v_0 to a destination vertex v_n is a sequence

$$W = \langle v_0, e_1, v_1, e_2, \ldots, e_n, v_n \rangle \text{ s.t. } e_i = (v_{i-1}, v_i)$$

The *length* of the walk is the number of edges in the sequence W. A *closed walk* has the same starting and ending vertices. A *path* is a walk where every vertex appears at most once.
The *distance* $d(s, t)$ between a vertex s and a vertex t in a graph G is the length of the shortest path between s and t (if one exists) and $d(s, t) = +\infty$ otherwise.
A *weighted graph* $G = (V, E, \omega)$ is a graph and a function $\omega\colon E \to \mathbb{R}$ assigning weights to edges.

We now define some problems that drive our study of algorithms in graph theory.

Problem 1 Given an edge weighted graph $G = (V, E, \omega)$. Find a shortest path between s and t (length is the sum of the edge weights).

Problem 2 Minimum-weight spanning tree: Given an undirected graph $G = (V, E, \omega)$ find an acyclic connected subgraph of G of smallest weight.

Problem 3 Travelling salesman problem: We are given a complete weighted graph G. Find a cycle that visits every vertex exactly once and minimises its length.

Problem 4 Maximum flow problem: You have a directed graph G representing a system of pipes, each pipe has a capacity. We are given also the source S and the sink T, $S, T \in V(G)$. How much water can we pump through the network per time unit?

Definition. A *leaf* is a vertex of degree one.

Proposition. *Every tree T has at least two leaves.*

Proof. Consider $P = \langle v_1, \ldots, v_k \rangle$ a path of maximum length in T. Consider v_1 if v_1 has degree more than one then we can either extend the path or T is not acyclic. □

Lecture 3

Corollary. *If the degree of each node in G is ≥ 2 then G has a cycle.*

Proposition. *Each tree has exactly $|V| - 1$ edges.*

Proof. [By induction] □

Definitions. An acyclic graph is called a *forest.*
A *connected component* of G is a maximal connected subgraph of G. Let $c(G)$ be the number of connected components of G.

Corollary. *Any graph on n vertices has at least $n - c(G)$ edges.*

Proposition. *If G is a simple graph with n vertices then*

$$|E(G)| \leq \frac{(n-k)(n-k+1)}{2} \text{ where } k = c(G)$$

Proof. Let $c_1, \dots, c_k$ be the connected components of G, i.e. $c_i = (V_i, E_i)$ and $E(G) = \bigcup_{i=1}^k E_i$, $V = \bigcup_{i=1}^k V_i$, $n = |V|$. Then $n = \sum_{i=1}^k n_i$ and $n_i \geq 1$. Obviously $|E(G_i)| \leq \frac{n_i^2 - n_i}{2}$. Thus $|E(G)| \leq \sum_{i=1}^k \frac{n_i^2-n_i}{2}$. The maximum of RHS occurs when $|C_1| = |C_2| = \ldots = |C_{k-1}| = 1$ and $|C_k| = n - k + 1$.
Assume the contrary that the maximum occurs when $c_i = k$, and $c_j = k_s$ where $r \geq s \geq 2$. Then $|E(c_i)| + |E(c_j)| = \frac{r^2+s^2-r-s}{2}$. But is $c_i = k_{r+1}$ and $c_j = k_{s-1}$, then we have $\frac{(r+1)^2+(s-1)^2-(r+1)-(s-1)}{2}$ edges. We get a contradiction since $(r+1)^2 + (s-1)^2 > r^2 + s^2$. □

Definition. A *cut-edge* is an edge whose removal increases the number of connected components.

Theorem. *Let T be a graph with n vertices, then the following statements are equivalent:*

1. *T is a tree.*
2. *T contains no cycles and has $(n-1)$ edges.*
3. *T is connected and has $(n-1)$ edges.*
4. *T is connected and every edge is a cut-edge.*
5. *Any two veritces of T are connected by exactly one path.*
6. *T contains no cycles and for any new edges the graph $T + e$ has exactly one cycle.*

Proof. ($1 \implies 2$) [Previous proposition]and definition of tree.
($2 \implies 3$) Suppose T has k connected components. Then each component must be a tree (because they have no cycles) therefore T must contain $n - k$ edges, exactly $\implies k = 1$.
($3 \implies 4$) Let e be an edges of T. Since $T \setminus e$ has $n - 2$ edges by one of the corollaries, we must have $c(T) \geq 2$.
($4 \implies 5$) (By contradiction) Summpose x and y are two vertices of tress T that have two different paths, say p_1 and p_2 connecting them. Let u be the the first vertex where the paths diverge and let v be the first vertex where the paths meet again. Any edge of the cycle containing u and v is not a cut edge which contradicts (4).

($5 \implies 6$) T contains no cycles otherwise we would have two paths between $x, y \in$ cycle. If we add an edge $e = (u, v)$ we will create a cycle since T contains a path between u and v. If $T + e$ contains two cycles then both cycles must contain e. u, v have two different paths containing them in T is a contradiction.
($6 \implies 1$) [By defintion]

□

1.2.5 Huffman Trees and Optimal Prefix Codes

Definitions. A *binary code* is an assignment of symbols or other meanings to a set of bitstrings.
A *prefix code* is a binary code with the property that no codeword is an initial substring of any other codeword.

This allows for transmission without confusion.

Example. A prefix code for an alphabet: construct a binary tree with a leaf corresponding to a letter of your alphabet, every left-hand edge is labelled by 0 and right-hand edge is labelled by 1.

Lecture 4

Definitions. Assume we have a code. Each codeword has a length l_c and each symbol has its frequency p_c. The *code efficiency* is then$\sum p_c l_c$.
Let T be a binary tree with leaves $s_1, \ldots, s_l$ such that each leaf s_i is assigned weight w_i.
The *average weighted depth* of the binary tree T is $\mathrm{wt}(T) = \sum_{i=1}^{l} \mathrm{depth}(s_i) w_i$ where depth is the number of edges to the root.

Algorithm (Constructing Huffman Prefix Codes)**.**
Input $s_1, \ldots, s_l$ of symbols and a list of $w_1, \ldots, w_l$ of weights w_i correspoding to s_i.
Output A binary tree representing a prefix code for a set of symbols whose codewords have minimum average weighted length.
Initialise F to be a forest of isolated vertices, labelled $s_1, \ldots, s_l$ with respective weights $w_1, \ldots, w_l$.
For $i = 1$ to $l - 1$ Choose two trees T and T' from the forest of smallest weights.
Create a binary tree by adding a new vertex with T as a child on the left and T' a child on the right. Label the edges to these new child trees 0 and 1 respectively.
Assign to the new tree the weight $\mathrm{wt}(T) + \mathrm{wt}(T')$.
Replace T and T' in F by the new tree.
When finished return F.

Example.

Lemma. *If the leaves of a binary tree are assigned wights and if each internal vertex (node) is assigned a weight equal to the sum of its children's weights, then the tree's average weighted depth equals the sum of the weights of its internal (non-leaf) vertices.*

Proof. In assignments. □

Theorem. *For a given list of weights $w_1, \ldots, w_l$ a Huffman tree has the smallest average weighted depth among all binary trees whose leaves are assigned these weights.*

Proof. Induction on l. Let T be the Huffman tree. **Base case** $l = 2$. Then $\mathrm{wt}(T) = w_1 + w_2$, because T is the only tree on 3 vertices. **Inductive step** Assume that for some $l \geq 2$ the Huffman algorithm produces a Huffman tree of minimum average weighted depth for any list of l weights.
We are given $w_1 \leq w_2 \leq \ldots \leq w_{l+1}$. The Huffman tree H that is constructed by the algorithm consists of the Huffman tree $\bar{H}$ for the weights $w_1 + w_2, w_3, \ldots, w_{l+1}$ where the leaf with weight $w_1 + w_2$ is replaced by a tree with two children of weights w_1 and w_2. By the previous lemma, we know $\mathrm{wt}(H) = \mathrm{wt}(\bar{H}) + w_1 + w_2$. By the inductive hypothesis, $\bar{H}$ is optimal among all binary trees whose leaves are assigned weights $w_1 + w_2, w_3, \ldots, w_{l+1}$.
Suppose T^* is an optimal binary tree for the weights $w_1, \ldots, w_l$. Let x be an internal vertex of T^* of greatest depth and suppose y and z are its left child and right child respectively. Without loss of generality, y and z have weights w_1 and w_2 otherwise we can swap their weights with w_1 and w_2 and produce a tree with the same or better weights. Delete y and z from T^* and call this tree $\bar{T}$, we have $\mathrm{wt}(T^*) = \mathrm{wt}(\bar{T}) + w_1 + w_2$. We also know that $\mathrm{wt}(\bar{T}) \geq \mathrm{wt}(\bar{H}) \implies \mathrm{wt}(T^*) \geq \mathrm{wt}(H)$. □

Lecture 5

Definition. Given trees $T = (V, E)$ and $T' = (V, E')$ we say $T = T'$ if $E = E'$.

Example.

Theorem (Cayley's Formula)**.** *The number of n vertex labelled trees is n^{n-2}.*

Definition. A *Prüfer sequence* of length $n - 2$ for $n \geq 2$ is any sequence of integers between 1 and n with repetitions allowed. There are n^{n-2} of these.

Algorithm.
Input An n-vertex tree.
Output A Prüfer sequence of length $n - 2$.

```
Initialise $T$ to be a given tree
For $i = 1$ to $n - 2$
    Let $v$ be the vertex of degree 1 with smallest label.
    Let $s_i$ be the only neighbour of $v$.
    Define $T = T \setminus v$
Return the sequence $\langle s_1, \ldots, s_{n-2} \rangle$
```

Example.

Proposition. *Let d_k be the number of occurences of the number k in a Prüfer encoding sequence for a labelled tree T. Then the degree of k in T is $d_k + 1$.*

Proof. By induction:
The assertion is true for any tree on 3 vertices since the Prüfer sequence consists of a single label of the vertex of degree 2.
Assume the assertion is true for all labelled trees on n vertices for some $n \geq 3$. Let T be a labelled tree on $n+1$ vertices. Let v be a leaf with smallest label $l(v)$. Let w be a neighbour of v. Then the Prüfer sequence S for T consists of $l(w)$ followed by S^*, the Prüfer sequence of $T^* = T \setminus v$.
Now by the inductive hypothesis, for each $u \in T^*$ the number of occurences of $l(W)$ in S^* is $d_{T^*}(u) - 1$. But for all $u \neq w$ the number of occurences of the label $l(u)$ in S^* is the same as in S, and $d_T(u) = d_{T^*}(u)$. Moreover, $d_T(w) = d_{T^*}(w) + 1$ and $l(w)$ has one more occurence in S than in S^*. □

Algorithm (for Prüfer decoding)**.**
Input A Prüfer sequence of length $n-2$.
Output An n-vertex labelled tree.