Math 262B Lecture Note 2

Prof. Fan Chung Graham

April 24, 2003

Note: this borrows heavily from Van Lint and Wilson's *A course in Combinatorics* - pages 79-83

1 An Addressing Problem

Theorem (Winkler '83) Let G = (V,E). Then q(G) $\leq$ | V(G) | - 1

Proof
First, pick a vertex, x_0, then create a spanning tree T by a breadth-first search, and then number the vertices by a depth-first search.

Let $n := |V(G)| - 1$.
For i $\leq$ n, define

$P(i) := \{j : x_j$ is on a path from x_0 to x_i in T$\}$.

$i \triangle j := max(P(i) \bigcap P(j))$

i' := max(P(i) $\backslash \{i\}$)

$i \sim j \iff$ P(i) $\subseteq$ P(j) or P(j) $\subseteq$ P(i)

Denote distances in G by d_G, and distances in T by d_T.

Def: *discrepancy function* $c(i, j) := d_T(x_i, x_j) - d_G(x_i, x_j)$

Lemma 1.
(i) $c(i, j) = c(j, i) \geq 0$
(ii) if $i \sim j$, then $c(i, j) = 0$
(iii) if $i \nsim j$, then $c(i, j') \leq c(i, j) \leq c(i, j') + 2$

Proof. (i) is trivial; (ii) follows from the definition of T since

$$d_G(x_i, x_j) \geq |d_G(x_j, x_0) - d_G(x_i, x_0)| = d_T(x_i, x_j)$$

(iii) follows from the fact that $|d_G(x_i, x_j) - d_G(x_i, x_{j'})| \leq 1$ and that $d_T(x_i, x_j) = 1 + d_T(x_i, x_{j'})$ $\square$

Now the addressing. For $0 \leq i \leq n$ the vertex x_i is given the address $\mathbf{a}_i \epsilon \{0, 1, *\}^n$, where

$$\mathbf{a}_i = (a_i(1), a_i(2), ..., a_i(n))$$

and

$$a_i(j) := \begin{cases} 1 & \text{if j } \epsilon \text{ P(i)} \\ * & \begin{cases} \text{c(i,j) - c(i,j') = 2}, & \text{or} \\ \text{c(i,j) - c(i,j')} = 1, i < j, \text{c(i,j)} even, & \text{or} \\ \text{c(i,j) - c(i,j')} = 1, i > j, \text{c(i,j)} odd \end{cases} \\ 0 & \text{otherwise} \end{cases}$$

Lemma 2. $d(\mathbf{a}_i, \mathbf{a}_k) = d_G(x_i, x_k)$

Proof. WLOG, assume that $i < k$.

If $i \sim k$, then $d_G(x_i, x_k) = |P(k) \backslash P(i)|$. $j \in P(k) \backslash P(i)$ iff $a_k(j) = 1$ and $a_i(j) \neq 1$. For these values of j, we see that $c(i, j) = 0$, hence $a_i(j) = 0$ and we're done.

If $i \nsim k$, then let $n_1 \leq n_2 \leq ... \leq n_l$ be a nondecreasing sequence of integers such that $|n_{i+1} - n_i| \leq 2$ for all i. If m is an even integer between n_1 and n_l that does not occur in the sequence, then there is an i such that $n_i = m - 1, n_{i+1} = m + 1$. Now consider the sequence

$$c(i, k) \geq c(i, k') \geq c(i, k'') \geq ... \geq c(i, i \triangle k) = 0$$

So, by the definition of $a_i(j)$ and the comments above, $a_i(j) = *$ and $a_k(j) = 1$ exactly as many times as there are even integers between $c(i, i \triangle k)$ and $c(i, k)$. Similarly, $a_k(j) = *$ and $a_i(j) = 1$ as many times as there are odd integers between $c(i, i \triangle k)$ and $c(i, k)$. So

$$\begin{aligned} d(\mathbf{a}_i, \mathbf{a}_k) &= |P(k) \backslash P(i)| + |P(i) \backslash P(k)| - c(i, k) \\ &= d_T(x_i, x_k) - c(i, k) \\ &= d_G(x_i, x_k). \end{aligned} \tag{1}$$

$\square$

Thus, we have proven the theorem.

2 Distance and Diameter

There are many notions of distance. Distance answers the question "how far are 2 things(points/vertices) apart?"

In a more rigorous setting, we speak of metric instead of distance. A metric, g, is defined to have the following three properties:
(1) g(x,z) $\leq$ g(x,y) + g(y,z)

(2) g(x,x) = 0
(3) g(x,y) = g(y,x)

Examples of metrics:

Graph(undirected) distance:
$d(u, v)$ = the length of the shortest path from u to v.
Directed graphs typically violate property 2 and 3, and thus is not considered a metric.

For l_p spaces,
#1 l_1–distance $d_1(x, y) = \Sigma|x_i - y_i|$
#2 l_2–distance $d_2(x, y) = [\Sigma|x_i - y_i|^2]^{1/2}$
#3 l_p–distance $d_p(x, y) = [\Sigma|x_i - y_i|^p]^{1/p}$
#4 l_∞–distance $d_\infty(x, y) = max|x_i - y_i|$

What does the l stand for? Man may never know.

2.1 Computing graph Diameter

Graph diameter $D = max_{u,v} d(u, v)$, the length of the "longest shortest path".

To compute the diameter of a tree, pick any vertex, and find its furthest point (computing a BFS will do this). Then pick that furthest point, and find the furthest point from it.

A more general way to compute the graph diameter is to compute the shortest path between each pair of vertices, and take the max. This is known as *all pairs shortest path*, which could be done using a BFS on each vertex, which takes $O(n^3)$ time. This is little worse than the current best all pairs shortest path algorithm. Due to the large overlap of data in the BFS trees, this bound is very unsatisfactory.