Lecture 5 Shortest Paths and Transitive Closure

5.1 Single-Source Shortest Paths

Let $G = (V, E)$ be an undirected graph and let ℓ be a function assigning a nonnegative length to each edge. Extend ℓ to domain $V \times V$ by defining $\ell(v, v) = 0$ and $\ell(u, v) = \infty$ if $(u, v) \notin E$. Define the *length*[2] of a path $p = e_1 e_2 \ldots e_n$ to be $\ell(p) = \sum_{i=1}^{n} \ell(e_i)$. For $u, v \in V$, define the *distance* $d(u, v)$ from u to v to be the length of a shortest path from u to v, or ∞ if no such path exists. The *single-source shortest path problem* is to find, given $s \in V$, the value of $d(s, u)$ for every other vertex u in the graph.

If the graph is unweighted (*i.e.*, all edge lengths are 1), we can solve the problem in linear time using BFS. For the more general case, here is an algorithm due to Dijkstra [28]. Later on we will give an $O(m + n \log n)$ implementation using Fibonacci heaps. The algorithm is a type of greedy algorithm: it builds a set X vertex by vertex, always taking vertices closest to X.

[2] In this context, the terms "length" and "shortest" applied to a path refer to ℓ, not the number of edges in the path.

Algorithm 5.1 (Dijkstra's Algorithm)

$X := \{s\}$;
$D(s) := 0$;
for each $u \in V - \{s\}$ do
    $D(u) := \ell(s, u)$;
while $X \neq V$ do
    let $u \in V - X$ such that $D(u)$ is minimum;
    $X := X \cup \{u\}$;
    for each edge (u, v) with $v \in V - X$ do
        $D(v) := \min(D(v), D(u) + \ell(u, v))$
end while

The final value of $D(u)$ is $d(s, u)$. This algorithm can be proved correct by showing that the following two invariants are maintained by the while loop:

- for any u, $D(u)$ is the distance from s to u along a shortest path through only vertices in X;
- for any $u \in X$, $v \notin X$, $D(u) \leq D(v)$.

5.2 Reflexive Transitive Closure

Let E denote the adjacency matrix of the directed graph $G = (V, E)$. Using Boolean matrix multiplication, the matrix E^2 has a 1 in position uv iff there is a path of length exactly 2 from vertex u to vertex v; *i.e.*, iff there exists a vertex w such that $(u, w), (w, v) \in E$. Similarly, one can prove by induction on k that $(E^k)_{uv} = 1$ iff there is a path of length exactly k from u to v.

The reflexive transitive closure of G is

$$\begin{aligned} E^* &= I \vee E \vee E^2 \vee \cdots \\ &= I \vee E \vee E^2 \vee \cdots \vee E^{n-1} \\ &= (I \vee E)^{n-1} . \end{aligned}$$

The infinite join is equal to the finite one because if there is a path connecting u and v, then there is one of length at most $n - 1$.

Suppose that two $n \times n$ Boolean matrices can be multiplied in time $M(n)$. Then $E^* = (I \vee E)^{n-1}$ can be calculated in time $O(M(n) \log n)$ by squaring E $\log n$ times. We will show below how to calculate E^* in time $O(M(n))$. Conversely, if there is an algorithm to compute E^* in time $T(n)$, then $M(n)$ is $O(T(n))$ (under the reasonable assumption that $M(3n)$ is $O(M(n))$): to multiply A and B, place them strategically into a $3n \times 3n$ matrix, then take its reflexive transitive closure:

$$\begin{bmatrix} 0 & A & 0 \\ 0 & 0 & B \\ 0 & 0 & 0 \end{bmatrix}^* = \begin{bmatrix} I & A & AB \\ 0 & I & B \\ 0 & 0 & I \end{bmatrix} .$$

The product AB can be read off from the upper right-hand block.

Here is a divide and conquer algorithm to find E^* in time $M(n)$.

Algorithm 5.2 (Reflexive Transitive Closure)

1. Divide E into 4 submatrices A, B, C, D of size roughly $\frac{n}{2} \times \frac{n}{2}$ such that A and D are square.

$$E = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

2. Recursively compute D^*. Compute

$$F = A + BD^*C \ .$$

   Recursively compute F^*.

3. Set

$$E^* = \left[\begin{array}{c|c} F^* & F^*BD^* \\ \hline D^*CF^* & D^* + D^*CF^*BD^* \end{array}\right] .$$

Essentially, we are partitioning the set of vertices into two disjoint sets U and V, where A describes the edges from U to U, B describes edges from U to V, C describes edges from V to U, and D describes edges from V to V. We compute reflexive transitive closures on these sets recursively and use this information to describe the reflexive transitive closure of E. Note that we compute two reflexive transitive closures, a few matrix multiplications (whose complexity is given by M) and a few matrix additions (whose complexity is assumed to be quadratic) of matrices of roughly half the size of E. This gives the recurrence

$$T(n) = 2T(\frac{n}{2}) + cM(\frac{n}{2}) + d(\frac{n}{2})^2$$

where c and d are constants. Under the quite reasonable assumption that $M(2n) \geq 4M(n)$, the solution to this recurrence is $O(M(n))$.

5.3 All-Pairs Shortest Paths

Let E denote the adjacency matrix of a directed graph with edge weights. Replace the 1's in E by the edge weights and the 0's by ∞. Apply Algorithm 5.2 to calculate E^*, except use $+$ instead of $\wedge$ and min instead of $\vee$. We will show next time that this solves the all-pairs shortest path problem.