CS 466: Transitive Closure vis-à-vis Matrix Multiplication

Arash Farzan

September 23, 2008

In this lecture, we build on the Strassen method and see how problems are reduced to one another.

Boolean Matrix Multiplication

In the previous lecture, we learned the Strassen method to compute the product of two $n \times n$ matrices in $o(n^3)$ (more precisely, $O(n^{2.81})$).

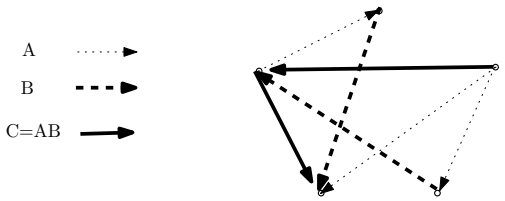
Matrix multiplication over boolean matrices is defined as follows. Given two $n \times n$ boolean matrices A, B, The product is a $n \times n$ boolean matrix $C := AB$ such that

$$C_{ij} = \bigvee_{1 \leq k \leq n} (a_{ik} \wedge b_{kj}).$$

It is not possible to adapt Strassen's method to compute boolean matrix multiplication (as there is no logic operation corresponding to subtraction). However, one can work over integers. We can put zeros and ones in A, B, and compute $C = AB$. We only have to scan the matrix and replace non-zeros by ones to get obtain the boolean product. This is all true assuming integers up to value n can be multiplied, added and subtracted in constant time per operation.

Realization of matrix multiplication in graphs. Let A, B be adjacency matrices of two graphs over the same set of vertices ($\{1, 2, \ldots, n\}$. We define an (A, B)-path as a path of length two whose first edge belongs to A and its second edge belongs to B.

We now form the product matrix $C = AB$. One can easily see that $c_{ij} = 1$ if and only if there is an (A, B)-path from vertex i to vertex j. Therefore, matrix C is the adjacency matrix with respect to (A, B)-paths:



Transitive Closure

Given a directed graph $G = (V, E)$, the *Trnasitive closure* of G is defined as the graph $G^* = (V, E^*)$ where $E^* = \{(i, j) : \text{there is path from vertex } i \text{ to vertex } j \text{ in } G\}$.

Cubic solution

Refer to page 633 of the [CLRS] for a discussion of two algorithms which take $O(n^3)$ time. The second algorithm can be thought of as a dynamic programming with the following recurrence relation:

$$t_{ij}^{(0)} = \begin{cases} 0 & \text{if } i \neq j \text{ and } (i,j) \notin E \\ 1 & \text{if } i = j \text{ or } (i,j) \in E \end{cases}$$

and for $k \geq 1$,

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}).$$

Beating the cubic solution

We observe that by squaring an adjacency matrix, we get in position (i, j), the number of ways of getting from i to j in 2 steps. If we make the diagonal of the matrix all 1's, this gives the number of ways of going from i to j in "at most 2 steps". As we are interested in only existence of paths and not their enumeration, we work with the corresponding boolean matrices. If we keep squaring, say $\lceil \lg n \rceil$ times, we will get one values in location i, j if there is a path from i to j.

Since there are $\lceil \lg n \rceil$ iterations to this algorithm and each step is squaring a boolean matrix (that is, matrix multiplication by itself), we derive a $O(n^{2.81} \lg n)$ time algorithm.

Better still: getting rid of the $\lg n$ factor

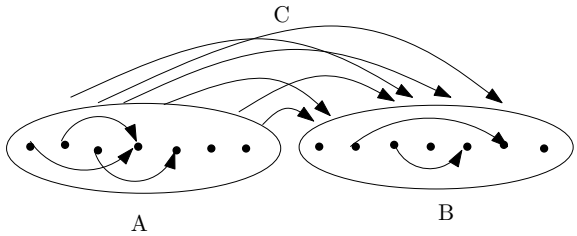
We can drop the extra $\lg n$ in the previous solution by following these steps:

- Determine the "strongly connected components" of the graph (We learned efficient $O(n^2)$ algorithms for this matter in CS 341). Collapse each component into a single vertex. Now given a solution to the new graph, it is easy to recover the solution and extend it to the original graph. Therefore, we have reduced to the problem to the one in the new graph.
- Do a topological sort of the vertex of the graph such that edges are oriented from lower numbered vertices to higher numbered ones.
- Now the adjacency matrix of the graph is an upper triangular matrix:

  $$G = \left(\begin{array}{c|c} A & C \\ \hline 0 & B \end{array}\right),$$

  where A, B are themselves upper triangular matrices with 1's on the diagonal.

  A deals with the edges in the first half of the graph and B deals with the edges in the second half of the graph. C deals with edges from section A to section B and there are no back edges in the opposite direction:



  We can claim that

  $$G^* = \left(\begin{array}{c|c} A^* & A^*CB^* \\ \hline 0 & B^* \end{array}\right).$$

The reason why is that connections within A are independent of B as if we exit from section A in the graph, there is no chance of getting back in. Similarly, connections within B are independent of A. Therefore, the top-lefthand quarter matrix in G^* is actually A^* and the bottom-righthand quarter matrix is B^*. As for the top-righthand quarter matrix, Connection from section A to section B of the graph are found by "taking a few steps" internal to section A and then using an edge as described by C from A to B and then "taking a few steps" internal to section B and therefore the top-righthand quarter matrix is A^*CB^*.

Hence, G^* can be found with determining the transitive closure of two graph with half-size (A^*, B^*), and two boolean matrix multiplication (to compute A^*CB^*). Therefore, one can write the following recurrence relation for the running time:

$$T(n) = 2T(n/2) + O(n^{2.81\cdots}).$$

Using the Master Theorem, this solves to $T(n) = O(n^{2.81\cdots})$.

This means that essentially the problem of computing the transitive closure reduces to the problem of boolean matrix multiplication.

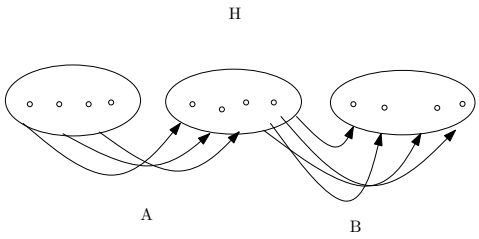
Reduction in the other direction

We showed that the transitive closure computation reduces to boolean matrix multiplication. We now show the other way of the reduction which concludes that these two problems are essentially the same.

Given boolean matrices A, B to compute the product $C = AB$, we form the following matrix:

$$H = \left(\begin{array}{c|c|c} I & A & 0 \\ \hline 0 & I & B \\ \hline 0 & 0 & I \end{array}\right).$$

This matrix is the adjacency matrix of a tripartite graph with edges going from the first to the second and the second to the third part:



It is not hard to see that the transitive closure of such graph is formed by adding the edges from the first part to the third part. These edges are described by the product of matrices A, B. Therefore, the transitive closure of H has the following form:

$$H^* = \left(\begin{array}{c|c|c} I & A & AB \\ \hline 0 & I & B \\ \hline 0 & 0 & I \end{array}\right).$$

This fact implies that the product AB can be determined from the transitive closure of matrix H.