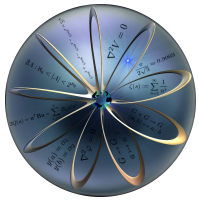

planetmath.org

Math for the people, by the people.

maximal bipartite matching algorithm

Canonical name	MaximalBipartiteMatchingAlgorithm
Date of creation	2013-03-22 12:40:11
Last modified on	2013-03-22 12:40:11
Owner	mathcam (2727)
Last modified by	mathcam (2727)
Numerical id	7
Author	mathcam (2727)
Entry type	Algorithm
Classification	msc 05C70

The maximal bipartite matching algorithm is similar some ways to the Ford-Fulkerson algorithm for network flow. This is not a coincidence; network flows and matchings are closely related. This algorithm, however, avoids some of the overhead associated with finding network flow.

The basic idea behind this algorithm is as follows:

1. Start with some (not necessarily maximal) matching M.
2. Find a path that alternates with an edge $e_1 \notin M$, followed by an edge $e_2 \in M$, and so on, ending with some edge $e_f \notin M$.
3. For each edge e in the path, add e to M if $e \notin M$ or remove e from M if $e \in M$. Note that this must increase $|M|$ by 1.
4. Repeat until we can no longer augment the matching in this manner.

The algorithm employs a clever labeling trick to find these paths and to ensure that the set of edges chosen remains a valid matching.

The algorithm as described here uses the matrix form of a bipartite graph. Translating the matching from a matrix to a graph is straightforward.

There are two phases to this algorithm: labeling and flipping.

Labeling We begin with a matrix with R rows and C columns containing 0s, 1s, and 1$*$s, where a 1$*$ indicates in edge in the matching and a 1 indicates an edge not in the matching. Number the columns $1 \dots C$ and number the rows $1 \dots R$.

Start by labeling each column that contains no 1$*$s with the symbol #.

Now we scan the columns. Scan each column i that has been labelled but not scanned. Find each 1 in column i that is in an unlabelled row; label this row i. Mark column i as scanned.

Next, we scan the rows. Scan each row j that has been labelled but not scanned. Find the first 1$*$ in row j. Label the column in which it appears j, and mark row j as scanned. If there is no 1$*$ in row j, proceed to the flipping phase.

Otherwise, go back to column scanning. Continue scanning and labelling until there are no labelled, unscanned rows or columns; at that point, the set of 1$*$s is a maximal matching.

Flipping We enter the flip phase when we scan some row j that contains no $1*$. This row must have some label c, and in column c, row j of the matrix, there must be a 1; change this to a $1*$.

Now consider column c; it has some label r. If r is #, clear all the labels and mark all rows and columns unscanned, and begin the labeling phase again. Otherwise, change the $1*$ at column c, row r to a 1.

Move on to row r and scan this row.

Notes The algorithm must begin with some matching; we may begin with the empty set (or a single edge), since that is always a matching. However, each iteration through the process increases the size of the matching by exactly one. Therefore, we can make a simple optimization by starting with a larger matching. A naïve greedy algorithm can quickly choose a valid matching that is usually close to the size of the maximal matching; we may initalize our matrix with that matching to give the procedure a head start.