Rainbow Perfect Matchings

Mats Rydberg & Martin Larsson

October 10, 2012

Algorithm

```
1. Fix prime $p \gg n$ [1]
2. For each colour $C \in [n]$
       Construct $n * n$ matrix $m_C$
       For each $uv \in E$ with $c(uv) = C$
           Pick random integer $r \in (0, p]$
           Set $m_C[u, v] = r$
3. Set $B = \sum_{i=0}^{n-1} m_i$
4. Compute $d_B = \det(B) \bmod p$
5. If $d_B = 0$ return "no"
6. Else set $sum = 0$
7. For each $X \subset [n]$
       Initialize $M = \mathbf{0}$ [2]
       For each $C \in X$
           Set $M = M + m_C$
       Set $sum = sum + (-1)^{n-1-|X|} \cdot \det(M) \bmod p$
8. If $d_B - sum = 0$ return "no" else return "yes"
```

[1] In our case, we selected $p = 32749$.

[2] $\mathbf{0}$ is the $n * n$ all-zeroes matrix.

Our algorithm modifies the given Algorithmic Piece 1 on step 2, by creating n matrices, one for each colour. But in step 3 we combine them into the biadjacency matrix B (called A_G in the assignment) and do the same end condition for its determinant.

For Algorithmic Piece 2, we have modified the pseudo code to be defined via matrix sums instead. We construct the biadjacency matrix for the current set of colours by simple addition, and compute the determinant for every such matrix. We are not including det(B) in *sum*, so to get the signs right we subtract an extra 1 in the exponent for -1.[3] This is really just an optimization, as we could remove steps 3 and 4 and have Algorithmic Piece 2 more or less intact. Our algorithm is faster for "no"-instances, however. With this change in mind, the logic is the same as in the assignment for why a non-rainbow perfect matching will eliminate itself in the calculation of *sum*.

[3] Another way to fix this would be to compute $d_B + sum$ in step 7 of the algorithm, but we thought this was cleaner.

Running Time

1. $O(1)$
2. $O(n)$

    $O(1)$

    Worst case n edges of colour C from all n nodes $\Rightarrow O(n^2)$

        $O(1)$

        $O(1)$

3. $T(n) = n^3$ additions. $O(n^3)$
4. $O(1)$
5. $O(1)$
6. There are 2^n subsets to a set of n elements $\Rightarrow T(n) = 2^n -$ 2(the empty set and the full set) $\Rightarrow O(2^n)$

    $O(1)$

    Since $|X| \leq n-1$, this is $O(n)$, right?

        n^2 additions $\Rightarrow O(1)$ or $O(n^2)$?

    $T(n) = 1 + (n-2) + O(det(M))$

7. $O(1)$

$$T(n) = 1 + n(1 + n^2(1+1)) + (n^3|1) + 1 + 1 + (2^n - 2) \cdot (1 + n(n^2|1) + 1 + 1 + (n-2) + O(det(M))$$

$$\leq 3 - n + n^3 - 2T(det(M)) + 2^n(n + n^3 + T(det(M)))$$

$$\leq 2^n(n + n^3 + T(det(M)))$$

$$T(det(M)) = ?$$

The anwer will probably become

$$O(2^n)$$

eventually. But how do we convince ourselves?

I asked on Piazza.

Failure bound

Analysis

Lol

The files are in the data directory are:

three.txt The 4-vertex graph from Fig. 1.

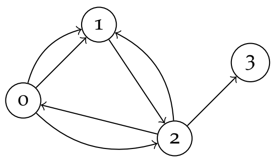

Figure 1: A directed multigraph.

Report part maybe

Transition probabilities

The transition matrix for the graph described in three.txt is[4]

[4] Fill in the right values. Set $\alpha = \frac{85}{100}$.

$$P = \begin{pmatrix} 1 & 6 & \pi & 1 \\ 1 & 1/e & -2 & \cdots \\ 1 & 1 & 0 & \\ \vdots & & & \end{pmatrix},$$

text is text is text is text

three.txt	2 (36.6%)	1 (27.5%)	0 (18.4%)	3 (17.3%)
tiny.txt	[...]			
medium.txt				
wikipedia.txt				
p2p-Gnutella08-mod.txt				