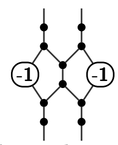
EITQ TD: Counting Complexity

Renaud Vilmart
renaud.vilmart@inria.fr

October 23, 2024

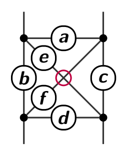
1 Perfect Matchings in Planar Graphs

Question 1. Give a tensor that represents $|0\rangle$ (i.e. a tensor with 1 output edge, which should **never** be in a perfect matching). Give a tensor that represents $|1\rangle$ (i.e. a tensor with 1 output edge, which should **always** be in a perfect matching).

Question 2. Check that  =  indeed has interpretation $|i, j\rangle \mapsto (-1)^{ij}|j, i\rangle$.

Hint: you can check all 4 cases where the input edges are assumed to be in a perfect matching or not.

Question 3. What is the interpretation of the usual swap? Is it a matchgate? Is it a *planar* matchgate?

Question 4. Compute the interpretation of  . Show that <u>almost</u> any **planar** matchgate can be represented this way (up to a global scalar). Show that $X \otimes X$ (with X the usual Pauli gate) is a planar matchgate. Show that we can use $X \otimes X$ to deal with the “almost” above.

2 Detecting Cycles

Let $G = (V, E)$ be a **directed** graph.

Question 1. Consider a vertex $v \in V$ with only outgoing edges. How many cycles can go through this node? Same question for a vertex with only incoming edges.

Question 2. Suppose all the vertices in G have at least 1 incoming edge, and at least 1 outgoing edge. Does G necessarily have a cycle?

Question 3. From the previous observations, suggest an algorithm to detect the presence of a cycle in a directed graph. What is the complexity of this algorithm?

3 Permanent and Cycle Covers

Let $G = (V, E, w)$ be a weighted **directed** graph, and M_G an adjacency matrix of G. A cycle cover of G is a subset C of edges E of G, such that for each vertex $v \in V$, there is exactly one edge $(v, \star) \in C$ and exactly one edge $(\star, v) \in C$.

We call #CC the function that computes the (weighted) number of cycle covers of a graph. Our goal in this exercise is to prove that:

$$\#\mathrm{CC}(G) = \mathrm{PERM}(M_G)$$

Recall the definition of the permanent:

$$\mathrm{PERM}(M) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} m_{i,\sigma(i)}$$

for all $n \times n$ matrices $M = (m_{i,j})$, with S_n the set of permutations over $\{1, ..., n\}$.

Question 1. Show the Laplace expansion formula: for any i:

$$\text{PERM}(M) = \sum_{j=1}^{n} m_{i,j}\, \text{PERM}(M_i^j)$$

where M_i^j is matrix M without row i and column j (called minor of M).

Question 2. Consider the outgoing edges of a vertex v of G. Reasoning on the (number of) cycles that go through v, determine a way to express the number of cycle covers of G from the number of cycle covers of graphs built from G, with one fewer vertex.

Question 3. Relate this with the Laplace expansion formula on M_G, and show that $\#\text{CC}(G) = \text{PERM}(M_G)$.

4 Permanent and Bipartite Perfect Matchings

Let $G = (V_0, V_1, E, w)$ be a weighted **bipartite** graph with $|V_0| = |V_1|$, and M_G be a biadjacency matrix of G. Our goal here is to prove that:

$$\#\text{PM}(G) = \text{PERM}(M_G)$$

Question 1. Considering a vertex $v \in V_0$, and reasoning on the different ways to cover v by a perfect matching, determine a way to compute $\#\text{PM}(G)$ from the number of perfect matchings of simpler graphs derived from G.

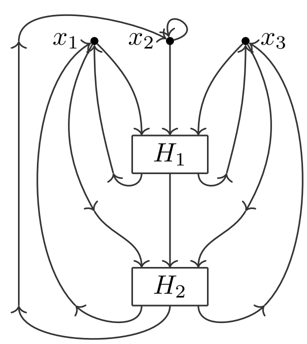
Question 2. Relate this to the Laplace expansion formula on M_G, and show that $\#\text{PM}(G) = \text{PERM}(M_G)$.

5 Cycle Covers and SAT Solutions

The objective here is to find a polynomial reduction from #3-SAT to #CC. For every 3-SAT formula φ, we want to build a graph G_φ such that it is easy to recover $\#\text{3-SAT}(\varphi)$ from $\#\text{CC}(G_\varphi)$. Consider we have a gadget (piece of graph) H that represents a clause. It has 3 incoming edges, and 3 outgoing edges. We build G_φ as follows:

- create one vertex v_i for each variable x_i of φ
- create one clause-gadget H_i for each clause C_i of φ
- for each variable x_i, create two "cycles" that go through v_i. One represents the "True" valuation for x_i, and the other the "False" valuation. If $C_i = \ell_{i_1} \vee \ell_{i_2} \vee \ell_{i_3}$, then if $\ell_{i_1} = x_{i_1}$ the **"True"** cycle of v_{i_1} will enter gadget H_i through the first incoming edge, and exit the gadget through the 1st outgoing edge; if $\ell_{i_1} = \neg x_{i_1}$ the **"False"** cycle of v_{i_1} will enter gadget H_i through the first incoming edge, and exit the gadget through the 1st outgoing edge. We do the same for all clauses, and all literals in the clauses.

For instance, if $\varphi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$, graph G_φ looks as follows:



Suppose the gadget H has adjacency graph M, with incoming wires to vertices (columns) 1, 2 and 3, and outgoing edges from vertices i_1, i_2 and i_3. We ideally want that incoming edge 1 is in the cycle cover iff outgoing edge 1 is in the cycle cover (and similarly for incoming/outgoing edges 2 and 3). But we also want that at least one of the incoming/outgoing edges is in the cycle cover (to represent clauses). We will use negative weights to make sure that the unwanted configurations cancel out.

Question 1. When none of the incoming/outgoing edges are in the cycle cover, you want the contribution to be 0. What does that mean for the permanent of M? When only the first incoming/outgoing pair of edges is in the cycle cover, the contribution is some non-zero constant c. What does that mean for the permanent of the matrix $M^1_{i_1}$? Express similarly all constraints on the permanent of minors of M.

Question 2. Show that the following matrix satisfies the above constraints with constant $c = 12$, and with $i_1 = 5$, $i_2 = 4$ and $i_3 = 3$:

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 & 1 \\ 0 & 0 & -1 & 2 & -1 & 1 & 1 \\ 0 & 0 & -1 & -1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 2 & -1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Question 3. Conclude by showing that #3-SAT $\preccurlyeq_\#$ $\#\mathrm{CC}_{\{-1,1,2,3\}}$.

6 #P-completeness of Counting Perfect Matchings

Question 1. Assuming that #3-SAT is **#P-complete**, putting everything back together, show that #PM is **#P-complete**.