Aufgabe 1 – *Lateinisches Rechteck*

Ein lateinisches $r \times n$-Rechteck ($r \leq n$) ist eine Anordnung der Zahlen $1, \ldots, n$ in r Zeilen und n Spalten, so dass in jeder Zeile und jeder Spalte jede Zahl höchstens einmal vorkommt. Ein lateinisches n-Quadrat ist ein lateinisches $n \times n$-Rechteck.

$$\begin{bmatrix} 4 & 1 & 2 & 3 \\ 2 & 3 & 1 & 4 \\ 3 & 2 & 4 & 1 \end{bmatrix}$$

Beispiel eines lateinischen 3×4-Rechtecks.

Angenommen wir haben ein lateinisches $r \times n$-Rechteck mit $r < n$ gegeben. Wir wollen sehen, ob wir es zu einem lateinischen n-Quadrat erweitern können. Das heisst, wir wollen $n - r$ weitere Zeilen mit Zahlen aus $1, \ldots, n$ zu dem Rechteck hinzufügen, ohne eine Spalte oder eine Zeile in der eine Zahl mehrfach vorkommt zu erhalten.

(a) Angenommen wir haben ein lateinisches $r \times n$-Rechteck wie oben beschrieben. Wir wollen zeigen, wann man dieses zu einem $(r + 1) \times n$-Rechteck erweitern kann. Beschreiben Sie, wie man dieses Problem mit einem bipartiten Graphen $G = (A \uplus B, E)$ modellieren kann und zeigen Sie, dass die Erweiterung genau dann möglich ist, wenn G ein perfektes Matching hat.

(b) Zeigen Sie, dass der in (a) konstruierte Graph regulär ist. Das heisst, zeigen Sie, dass es eine ganze Zahl k gibt, sodass alle Knoten (sowohl die Knoten in A als auch die Knoten in B) Grad genau k haben.

(c) Benutzen Sie ihr Ergebnis aus (b) um zu beschreiben, welche $r \times n$-Rechtecke man zu einem lateinischen n-Quadrat erweitern kann.

(d) Geben Sie einen Algorithmus an, der als Eingabe ein Lateinisches $r \times n$-Rechteck nimmt und es zu einem lateinischen n-Quadrat erweitert (falls ein solches existiert) und ansonsten "Nicht möglich" ausgibt.

Hinweis: In (a) kann es hilfreich sein, die Knoten in A mit 'Spalte 1', 'Spalte 2',... zu labeln und die Knoten aus B mit 'Nummer 1', 'Nummer 2',.... Was sind dann die Kanten? Was ist die entsprechende Bedeutung eines perfekten Matchings in Bezug auf das lateinische Quadrat?

Hinweis: Für (d) können Sie die Ergebnisse der vorherigen Teilaufgaben verwenden. Beachten Sie aber, dass zu einem Algorithmus auch immer eine Laufzeitanalyse und ein Korrektheitsbeweis gehören – auch wenn dies nur ein kurzer Hinweis auf eine der vorherigen Teilaufgaben ist!

Lösung zu Aufgabe 1 – *Latin Rectangle*

(a) After a bit of pondering, we can see that this is an assignment problem; the task is to assign each of the numbers $1, 2, \ldots, n$ to one of the n positions in the new row of the latin rectangle, with some additional restrictions about which assignments would be allowed (the rectangle should still be latin). Let us try to construct a graph, and see if we can capture these restrictions.

We construct the graph $G = (A \uplus B, E)$ as follows. We let $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$, where in the back of our heads we think of a_i and the "column i", and b_j as the "number j"[1]. We add an edge between vertices a_i and b_j iff the value j is not already present in the i:th column of the given latin $r \times n$ rectangle.

It remains to show that we can extend the latin rectangle by one more row iff G has a perfect matching.

$\Rightarrow$: Suppose the latin rectangle can be extended by adding the row $(x_1, x_2, \ldots, x_n)$. We claim that $M = \{\{a_1, b_{x_1}\}, \{a_2, b_{x_2}\}, \ldots \{a_n, b_{x_2}\}\}$ (i.e. we connect column i to the number x_i, as the back of our heads would expect) is a perfect matching. Indeed M clearly has the size of a perfect matching, so it only remains to show that it is a matching.

As $(x_1, x_2, \ldots, x_n)$ was a valid extension, x_i was not already present in column i, so reading off our construction above $\{a_i, b_{x_i}\}$ are all edges in G, i.e. $M \subseteq E$. It is clear from our definition of M above that every vertex in A is incident to at most (in fact exactly) one edge in M. Moreover, as by definition of latin rectangle, $(x_1, x_2, \ldots, x_n)$ has no repeating elements, the same holds for B. So we have shown that M is a matching, which concludes the argument.

$\Leftarrow$: Any perfect matching in a bipartite graph is a one-to-one mapping between A and B. In this case, between columns and the numbers $1, 2, \ldots n$. Given a perfect matching M in G we add one more row to the latin $r \times n$ rectangle where for each column we write the assigned value from the matching. That is, in column i, we write the number j where j is the unique value satisfying $\{a_i, b_j\} \in M$. As $\{a_i, b_j\} \in M \subseteq E$, it follows that j was not already present in column i, so the extension does not create a repeating element in a column. Moreover as b_j is only incident to one edge in M, no number will appear twice in the new row. Hence we have successfully extended the rectangle.

(b) By our definition of G, a vertex $a_i \in A$ is connected to all b_j where $j \in [n]$ is not present in the i:th column of the latin $r \times n$ rectangle. As there are r unique elements in each column, a_i has degree $n - k$. Furthermore a vertex $b_j \in B$ is connected to all a_i such that the value j is not present in the i:th column. As each j appears once in every row, it is present in the rectangle precisely r times. And as it never appears twice in the same column, as it is a latin $r \times n$ rectangle, j appears in precisely r columns. Thus, it is not present in $n - r$ columns and hence b_j has degree $n - k$. We conclude that G is $(n - r)$-regular.

(c) From (b) we have that any graph G constructed as in (a) is $n - r$-regular. As $n - r > 0$ and G is bipartite by construction, it follows by Frobenius theorem that it always has a perfect matching. Thus by (a) we can always extend a latin $r \times n$ rectangle by one more row, as long as $r < n$. We conclude that all latin $r \times n$ rectangles can be extended to latin squares.

(d) As extension is always possible, the algorithm should never return "Not possible".

Given any latin $r \times n$ square we proceed as follows: For each $t = r+1, r+2, \ldots, n$, we construct the bipartite graph G_t as described in (a), apply the Hopcroft-Karp algorithm to find a perfect matching M_t and extend the latin rectangle with one more row by writing in the i:th column the value j where b_j is the unique neighbor of a_i in the matching and repeat. Once $t = n$, we return the resulting $n \times n$ matrix.

[1] What is going on in the back of our minds has no formal significance for the proof; it is just there to help us think up the model.

Proof of correctness: As we have seen in (c), the graphs G_t have perfect matchings, so any maximum matching algorithm will inedeed find a perfect matching. As we saw in the $\Leftarrow$ part of (a), the extension described above does indeed preserve latin-ness. Clearly after step t, the rectangle consists of t rows, so after reaching $t = n$ this is indeed a latin square that extends the initial $r \times n$ rectangle, as desired.

Runtime analysis: The procedure is ran for at most n iterations. Constructing the graph G and storing it in a sensible format takes $O(n^2)$ time. Running Hopcroft-Karp on this graph and returning a perfect matching in a sensible format (e.g. a list of length n where the i:th element indicates which vertex a_i is matched with) takes $O(n^{2.5})$ time. Finally, reading off the matching to extend the rectangle can be done in $O(n)$ time. We see that the dominating term is computing the matching. Thus we get a total runtime of $O(n \cdot n^{2.5}) = O(n^{3.5})$.