Discrete Mathematics **Lent 2009**

MA210

Solutions to Exercises 9

(1) There are 5 cities. The cost of building a road directly between i and j is the entry $a_{i,j}$ in the matrix below. An indefinite entry indicates that the road cannot be built. Determine the least cost of making all the cities reachable from each other.

$$\begin{pmatrix} 0 & 3 & 5 & 11 & 9 \\ 3 & 0 & 3 & 9 & 8 \\ 5 & 3 & 0 & \infty & 10 \\ 11 & 9 & \infty & 0 & 7 \\ 9 & 8 & 10 & 7 & 0 \end{pmatrix}$$

Solution. We order the edges according to the weights: 12, 23, 13, 45, 25, 15, 24, 35, 14. Kruskal's Algorithm accepts edges $12, 23$, then rejects 13, then accepts 45, 25, and then it stops. Thus, the least cost to build the road network is $3 + 3 + 7 + 8 = 21$. □

(2) For natural numbers n and p, let $G(n, p)$ be the complete graph with vertex set $\{1, 2, \ldots, n\}$, and let the weight of the edge ij be given by $c_{ij} = |i - j| \mod p$. (So $c_{ij} \in \{0, 1, \ldots, p - 1\}$.) For every n and p, determine the minimum weight of a spanning tree in $G(n, p)$.

Solution. We must distinguish two cases: $p \geq n$ and $p < n$.

• When $p \geq n$, we have that

$$1 \leq |i - j| < n \leq p$$

for every $i \neq j$, $1 \leq i, j \leq n$. From this, we obtain $c_{ij} = |i - j| \geq 1$ for every two vertices $i \neq j$ of $G(n, p)$. Since every spanning tree of $G(n, p)$ must have $n-1$ edges and each edge has weight at least 1, it follows that every spanning tree of $G(n, p)$ must have total weight at least $n - 1$.

On the other hand, the edges $12, 23, \ldots, (n - 1)n$ have all weight 1 and form a spanning tree of $G(n, p)$. Hence, the minimum weight of a spanning tree of $G(n, p)$ is $n - 1$.

• When $p < n$, we must be more careful because some edges have weight 0. Let ij be any edge of $G(n,p)$. Then, $c_{ij} = 0$ if and only if $|i-j|$ is divisible by p. So, when does this happen ?

We write $n = mp + r$, where $0 \leq r < p$. We notice that any edge with both endpoints in $\{p, 2p, 3p, \ldots, mp\}$ or in $\{1, 1+p, 1+2p, \ldots\}$ has weight 0. More generally, for $0 < k \leq r$, we set

$$X_k = \{k, k+p, k+2p, \ldots, k+mp\}$$

and, for $r < k \leq p$, we set

$$X_k = \{k, k+p, k+2p, \ldots, k+(m-2)p, k+(m-1)p\}.$$

Note that $c_{ij} = 0$ if and only if $i, j \in X_k$ for some k.

We order the edges of $G(n,p)$ in the following order: first we list all the edges with both ends X_1, then all the edges with both ends in X_2, etc., until we list all the edges with both ends X_k. After this we put edges 12, 23, $\ldots$, $(p-1)p$, all of which have weight 1. Then, we list all the remaining edges in any non-decreasing order.

So, what happens when we run Kruskal's Algorithm on this order? It first accepts every edge with both endpoints in X_1 as long as it does not create a cycle. The crucial observation is that after Kruskal's Algorithm examines all the edges with both endpoints in X_1, the edges selected form a spanning tree on the vertices in X_1. In other words, Kruskal's Algorithm picked exactly $|X_1| - 1$ edges (each of weight 0).

The same thing remains true for the other sets X_k: after Kruskal's Algorithm examines all the edges with both endpoints in X_k, the edges selected form a spanning tree on the vertices in X_k. Hence, Kruskal's Algorithm picked exactly $|X_k| - 1$ edges.

So, before proceeding to edges 12, 23, etc., Kruskal's Algorith selected

$$(|X_1| - 1) + (|X_2| - 1) + \ldots (|X_p| - 1) = |X_1| + |X_2| + \cdots + |X_p| - p = n - p$$

edges. These edges form a forest with exactly p components (one on every X_k). Now, edge 12 connects components on X_1 and X_2 to a tree on $X_1 \cup X_2$. Adding the edge 23 creates a tree on $X_1 \cup X_2 \cup X_3$. Similarly, when we add the edge $(k-1)k$, we join two components, one on $X_1 \cup X_2 \cup \cdots \cup X_{k-1}$ and the other one on X_k, and obtain a tree on $X_1 \cup X_2 \cup \cdots \cup X_k$. Thus, we never create a cycle. Hence, Kruskal's Algorithm accepts $p-1$ edges 12, 23, $\ldots$, $(p-1)p$ and then stops because it selected $n-1$ edges.

What is the weight of this spanning tree? The only edges with non-zero weight are 12, 23, $\ldots$, $(p-1)p$, hence, the minimum weight of a spanning tree is $p-1$.

each of the edges 12, 23, $\ldots$, $(p-1)p$ connects two of the components (draw a picture to see this!) □

(3) (a) What is the chromatic number of the complete graph K_n?

Solution. Every two vertices of K_n are adjacent, therefore, every two vertices must have different colours in any colouring of K_n. Clearly, if every vertex get a distinct coulour, we obtain a proper colouring of K_n. Thus, $\chi(K_n) = n$. □

(b) What is the chromatic number of the path P_n?

Solution. Clearly $\chi(P_1) = 1$. Since P_n has at least one edge for $n \geq 2$, we have $\chi(P_n) \geq 2$. P_n also has no (odd) cycles, so it is bipartite. Each partite set can be viewed as one colour class, so we have a proper colouring of P_n with two colours. Hence, $\chi(P_n) = 2$ for $n > 1$. □

(c) What is the chromatic number of the cycle C_n?

Solution. Since C_n has at least one edge for $n \geq 3$, we have $\chi(C_n) \geq 2$. As observed in part (b), any graph is colourable with 2 colours if and only if it is bipartite.

For n even, C_n has no odd cycles, so it is bipartite and $\chi(C_n) = 2$. For n odd, C_n is not bipartite, so $\chi(C_n) > 2$. It is easy to see that if C_n has vertices $v_1, \ldots, v_n$ and edges $v_1v_2, v_2v_3, \ldots, v_{n-1}v_n, v_nv_1$, then $c(v_1) = c(v_3) = \cdots = c(v_{n-2}) = 1$, $c(v_2) = c(v_4) = \cdots = c(v_{n-1}) = 2$, $c(v_n) = 3$ is a proper colouring of C_n with 3 colours. Hence, for n odd, $\chi(C_n) = 3$. □

(4) Prove or disprove:

(a) Every k-chromatic graph G has a proper k-colouring in which some colour class has $\alpha(G)$ vertices.

Solution. This is false. Let G be a graph with vertices $\{a, b, c, A, B, C\}$ and edges $\{aA, aB, aC, bA, cA\}$. This graph is bipartite (why?), hence it is 2-chromatic. In every 2-colouring of G, the vertices a and A must be in different colour classes because aA is an edge. However, this force B, C to be in the same colour class as A, and b, c to be in the same colour class as a. Hence, in every 2-colouring of G, every colour class has size 3.

However, $\{b, c, B, C\}$ is an independent set of size 4, so $\alpha(G) > 3$. □

(b) For every n-vertex graph G, $\chi(G) \leq n - \alpha(G) + 1$.

Solution. Let $G = (V, E)$ be a graph and let X be an independent set of size $\alpha(G)$. We assign the same colour 1 to all the vertices of X. Each of the remaining $n - \alpha(G)$ vertices gets a new colour. Except colour 1, all the other colour classes have size 1, i.e., they cannot contain an edge. The colour class for colour 1 is X and it contains no edge because it is an independent set. Hence, we found a proper colouring of G using $n - \alpha(G) + 1$ colours. This means that $\chi(G) \leq n - \alpha(G) + 1$. □

(c) For every two vertex disjoint graphs G and H, $\chi(G + H) = \max\{\chi(G), \chi(H)\}$.

Here, $G + H$ is defined as follows: Let $G = (V, E)$ and $H = (V', E')$ be two graphs with disjoint vertex sets, i.e., $V \cap V' = 0$. The disjoint union of G and H, denoted by $G + H$, is the graph with vertex set $V \cup V'$ and edge set $E \cup E'$.

Solution. $G + H$ contains both G and H as subgraphs. SO, every proper colouring of $G + H$ also produces a proper colouring of G and H. Hence, $\chi(G + H) \geq \max\{\chi(G), \chi(H)\}$.

On the other hand, since $\chi(G), \chi(H) \leq \max\{\chi(G), \chi(H)\}$, we can find a proper colourings of G and H using $\max\{\chi(G), \chi(H)\}$ colours. Since there is no edge between G and H in $G + H$, we can combine these two colourings to a colouring of $G + H$. Hence, $\chi(G + H) \leq \max\{\chi(G), \chi(H)\}$. □

(5) Let G be a graph. Prove that there exists some ordering of the vertices of G such that the greedy algorithm uses exactly $\chi(G)$ colours.

Solution. Take any colouring of G using $\chi(G)$ colours. (**We may not know how this colouring looks like, but we do know it exists!**) We first list the vertices from colour class 1, then from colour class 2, etc., and the end we list the vertices from the colour class $\chi(G)$. So, we have an ordering of the vertices of G. (More precisely, we know that the ordering described above exists.)

So, how will the greedy algorithm (GA from now on) colour graph G using this order? First, remember that each colour class is an independent set. So, the GA will assign colour $1'$ to all the vertices coming from colour class 1. (We use $1'$ to distinguish this colour from colour 1 used in the definition of the ordering.)

Then, the GA will assign colour $1'$ or $2'$ to all the vertices coming from the colour class 2. This is because each such vertex may have neighbours among vertices from the colour class 1 (GA coloured these by $1'$) but none coming from the colour class 2.

In general, when the GA considers a vertex coming from the colour class i, it gets a colour from $\{1', 2', ..., i'\}$ because may have neighbours among vertices from colour classes $1, \ldots, i-1$ (GA coloured these by one of $\{1', \ldots, (i-1)'\}$) but none coming from the colour class i.

Hence, the GA used at most $\chi(G)$ colours $1', 2', \ldots, \chi(G)'$. But no proper colouring of G can use less than $\chi(G)$ colours (this comes from the definition of $\chi(G)$, so the GA used exactly $\chi(G)$ colours. □

(6) Find the minimum distance for the following codes:

(a) $C_1 = \{10000, 01010, 00001\}$;

(b) $C_2 = \{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\}$;

(c) $C_3 = \{000000, 101010, 010101\}$.

Suppose we want to add extra codewords to the codes above. For which of the three of them is that possible without altering the minimum distance?

Solution. A quick examination yields $\delta(C_1) = 2$, $\delta(C_2) = 2$, $\delta(C_3) = 3$.

We can add 11111 to C_1 and 111000 to C_3. (Check!) If we can add any word to C_2, it must have distance at least 2 from every word of C_2. Since $0000, 1111 \in C_2$, every such word must contain at least two 0's and at least two 1's. But the length of the code is 4 so if we want to enlarge C_2 without changing the minimum distance, such a word must have exactly two 0's and exactly two 1's. Number of such words is $\binom{4}{2} = 6$ (Why ?) and we see all of them are already in C_2. □

(7) Prove the triangle inequality: for all $\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{z}} \in \{0,1\}^n$,

$$d_H(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}) + d_H(\bar{\boldsymbol{z}}, \bar{\boldsymbol{y}}) \geq d_H(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}).$$

Solution. Let $\bar{\boldsymbol{x}} = x_1x_2 \ldots x_n$, $\bar{\boldsymbol{y}} = y_1y_2 \ldots y_n$, $\bar{\boldsymbol{z}} = z_1z_2 \ldots z_n \in \{0,1\}^n$ be given. Let A be the set of positions at which $\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}$ differ, that is,

$$A = \{i \mid 1 \leq i \leq n,\ x_i \neq z_i\}.$$

Then, $|A| = d_H(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}})$. Similarly, we define

$$B = \{i \mid 1 \leq i \leq n,\ z_i \neq y_i\},$$

$$C = \{i \mid 1 \leq i \leq n,\ x_i \neq y_i\}.$$

Then, $|B| = d_H(\bar{\boldsymbol{z}}, \bar{\boldsymbol{y}})$ and $|C| = d_H(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$.

We will show that $C \subset A \cup B$. Indeed, let $i \in C$, that is $x_i \neq y_i$. If $i \notin A \cup B$, then $i \notin A$ and $i \notin B$, so $x_i = z_i$ and $z_i = y_i$. Hence, $x_i = z_i = y_i$ and this is not possible. Thus, $i \in A \cup B$.

Consequently, we have $|C| \leq |A \cup B|$ and

$$d_H(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}) = |C| \leq |A \cup B| \leq |A| + |B| = d_H(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}) + d_H(\bar{\boldsymbol{z}}, \bar{\boldsymbol{y}}).$$

□

(8) Construct a binary code C of length 6 such that $|C| = 5$ and C is 1-error-correcting.

Solution. We know that C is 1-error-correcting if and only if $\delta(C) \geq 2(1) + 1 = 3$. So, we need to find binary code C of length 6 such that $|C| = 5$ and $\delta(C) \geq 3$. One such a code is

$$C = \{000000, 101010, 010101, 111100, 001111\}.$$

(Are you sure?) □