CS 38, PS 4

Codename: Jambul

May 23, 2023

1 Problem 1

1.1 Part a

Proof. From lecture, we know that a pair $(U, \mathcal{F})$, for U a finite set and $\mathcal{F}$ a collection of subsets of U, is a matroid if (1) $\mathcal{F} \neq \emptyset$, (2) $I \subseteq J$ and $J \in \mathcal{F} \implies I \in \mathcal{F}$, and (3) $I, J \in \mathcal{F}, |I| > |J| \implies \exists x \in I \backslash J$ such that $J \cup \{x\} \in \mathcal{F}$.

Now we take $U = S$ the set of columns of an $m \times n$ matrix $T \neq 0$ over $\mathbb{R}$ or $\mathbb{F}_2$ as defined, and $\mathcal{F} = \mathcal{I}$ the collection of sets of S such that $A \in \mathcal{I} \iff A$ has linearly independent columns.

Trivially, then, $\mathcal{I} \neq \emptyset$, assuming $m, n > 0$. In this case, since $T \neq 0$, it will be the case that $\mathcal{I}$ will have at least n elements. Indeed, in the worst case, every column in T will be linearly dependent with respect to every other column, but then each column on its own will be an element in $\mathcal{I}$, so $|\mathcal{I}| \geq n$.

We now seek to show that $I \subseteq J, J \in \mathcal{I} \implies I \in \mathcal{I}$. Suppose J is some element of $\mathcal{I}$. Then J is some set of columns from T such that each of the columns in J are linearly independent of one another. But then clearly if *every column* in J is linearly independent of every other, then any subset of the columns in J will also be linearly independent, since it must consist of columns already in J to begin with.[1] Thus indeed if $I \subseteq J$ for $J \in \mathcal{I}$, then $I \in \mathcal{I}$.

Finally, we wish to show that $I, J \in \mathcal{I}, |I| > |J| \implies \exists x \in I \backslash J$ such that $J \cup \{x\} \in \mathcal{I}$. Suppose I and J are elements of $\mathcal{I}$, so I and J are some subsets of linearly independent columns from S, with I having more columns than J. Then is must be the case that $I \backslash J$ is a set of at least one column. From our reasoning in the third paragraph above, we know that any single column will be an element of $\mathcal{I}$, so we can be certain that there will exist at least one column $x \in I \backslash J$ such that $x \in \mathcal{I}$. □

1.2 Part b

Proof. ($\Longleftarrow$) We will prove the contrapositive of this direction of the biconditional. Suppose a set of columns of M is not linearly independent over $\mathbb{F}_2$. Then

[1] This reasoning applies for both $\mathbb{R}$ as well as $\mathbb{F}_2$, since we are told that linear independence is defined in the same way over both spaces.

it must be the case that, for $M = (v_1, ..., v_k)$, $c_1 v_1 + ... + c_k v_k = 0, c_i \in \{0, 1\}$, where not all $c_i = 0$. But since $c_i \in \{0, 1\}$, and $0v_i = 0 \forall i$, then there exists at least one i such that $c_i = 1$. Furthermore, since we know $1 + 0 = 1, 1 + 1 = 0 + 0 = 0$, then either it must be the case that for at least one i, $c_i = 1$ for $v_i = 0$, or for some index set A, $c_i = 1 \forall i \in A$.

The former case would correspond to an edge with no starting or ending point, so we consider only the latter possibility. Since $1 + 1 = 0 + 0 = 0$ are the only summation operations on elements in $\mathbb{F}_2$ that yield 0, we require that every vertex in the subgraph G' induced by the edges $e_i, i \in A$ has even degree. However, if each vertex in the subgraph induced by $\{e_i : i \in A\}$ has even degree, then we recall that this subgraph must have an Eulerian circuit. That is, the subgraph must have a circuit which starts and ends at vertex u and touches each edge in its traversal only once. Of course, this would necessarily imply that the subgraph (and thus the entire graph) is cyclic.

($\Longrightarrow$) Now suppose G is cyclic. Then it must be the case that G contains some Eulerian circuit $C = \{e_k, ..., e_m\}$, $m > k$ so that C starts and ends at some vertex u. But we know that if a subgraph (like that induced by C) has an Eulerian circuit, it must be the case that every vertex in that subgraph has even degree. Then since $1 + 1 = 0 + 0 = 0$ in $\mathbb{F}_2$, we will have that

$$v_k + ... + v_m = 0,$$

for v_i the column vector corresponding to the edge e_i. Then one may simply set $c_i = 0$ whenever $e_i \in C$, and $c_i = 0$ otherwise to see that $c_1 v_1 + ... + c_n v_n = 0$. Since we have assumed C is non-empty ($m > k$, since otherwise G will not be cyclic), it must be the case that this implies M has linearly dependent columns.

We have now shown both directions of the contrapositive to the biconditional given, completing the proof. □

1.3 Part c

Proof. Let $\mathcal{I} \subset 2^E$ be such that $A \in \mathcal{I} \Longleftarrow$ the subgraph formed by edges from A is acyclic. Then from part (b), we know that the subgraph formed by edges from A is acyclic $\Longleftrightarrow$ it is the case that the column vectors of the incidence matrix $M(A)$ of A are linearly independent. Thus by part (a), we have that $(E, \mathcal{I})$ is a matroid, since E holds the same information as the set of columns of $M(A)$, and $A \in \mathcal{I} \Longleftrightarrow$ the columns of $M(A)$ are linearly independent. □

2 Problem 2

Proof. Suppose there exists some other minimum spanning tree T_2 of G such that $L(T_2) = (w_1^{(2)}, ..., w_n^{(2)}) \neq L(T_1) = (w_1^{(1)}, ..., w_n^{(1)})$, for $L(T_2)$ the sorted list of edge weights of T_2 (with repetition).

Then it must be the case that $w_i^{(1)} \neq w_i^{(2)}$ for some $1 \leq i \leq n$. WLOG (we can always relabel which MST is which), assume that $w_i^{(1)} < w_i^{(2)}$, and denote the edge in T_1 corresponding to $w_i^{(1)}$ by (u, v).

Now suppose we add (u, v) to T_2 to create a new graph T_2'. Since T_2 is a minimum spanning tree (and thus has the property that all its vertices are connected), it must be the case that T_2' has a cycle C. Suppose now that there exists some edge $e \in E(C)$, for $E(C)$ the edge set of C, such that the weight of e is greater than that of (u, v). But then we may simply delete e to break the cycle and obtain a connected tree $T_2' \backslash e$ on all vertices in T_2 but of weight less than T_2, a contradiction. We therefore assume that all edges in C have weight less than or equal to (u, v).

We will now show that some edge in $C \backslash (u, v)$ has the same weight as (u, v). Suppose instead that all edges in $C \backslash (u, v)$ have weight strictly less than (u, v). Then it must be the case that, if we were to remove (u, v) from T_1, we would be able to find an edge $e' \in C \backslash (u, v)$ such that $e' + (T_1 \backslash (u, v))$ would be a tree of lower weight than T_1, a contradiction. Thus it must be the case that there is some edge in $C \backslash (u, v)$ that is of the same weight as (u, v).

Replace this edge in T_2 of weight equal to (u, v) by (u, v). Then $L(T_1)$ and $L(T_2)$ should remain the same (so $L(T_1) \neq L(T_2)$), but T_1 and T_2 will have one more edge in common (this will be (u, v)). We can continue this process, updating T_2 to share another edge with T_1, until T_2 shares all its edges with T_1, in which case we must have that $L(T_1) = L(T_2)$, a contradiction. Thus indeed $L(T_1) = L(T_2)$ for T_1, T_2 any two MSTs of G. $\square$

3 Problem 3

Algorithm 1. We will provide a greedy solution to the problem. Our input will be a list L of intervals, in which each entry is of the form $(s_v, t_v), s_v < t_v$. We will use 1-indexing on all arrays.

```
1. Sort the intervals in $L$ in-place by starting point (that is, by $s_v$) in increasing order. Let $L$ be of length $n$.
2. Create an empty min heap $q$. The min heap will hold 2-tuples of the form $(t, i)$, where $t$ will be the element that dictates order within the heap.
3. Create an empty output array $o$.
4. For $v = 1$ to $n$:
   (a) If $q$ is empty: Push $(L[v][2], 1)$ onto $q$. Set $o[v] = 1$.
   (b) Else:
       i. If $q[1][1] \leq L[v][1] = t_v$: Pop the smallest element $(t, i)$ from $q$. Set $o[v] = o[i]$.
       ii. Else: Set $o[v] = |q| + 1$.
       iii. Push $(L[v][2], o[v])$ onto $q$.
5. Return $o$.
```

Proof of correctness. Let k be the maximum number of intervals in L that intersect at any point on the real line. We will prove the correctness of the algorithm given above by contradiction.

Suppose that the algorithm provides a coloring that uses more than k colors. Then consider the first time that the algorithm sets some node to be of color $k + 1$. When this occurs, it must be the case that there are k intervals that intersect over some interval $I = (s_I, t_I)$, because the algorithm will only opt to use a new color when there is no older color that may be reused. If there is in fact an interval that has not been popped from q (and thus whose color has not been reused yet) with an ending point before the current interval's starting point, then the node corresponding to the current interval will assume the color of the other interval, avoiding the use of color $k + 1$.

But then, if k intervals intersect in I, then each of those intervals must contain the starting point s_I of I. Furthermore, since the algorithm is coloring the current interval $k+1$, it must be the case that the current interval intersects with all k of those other intervals. Since we have sorted the intervals by starting points, it follows that the current interval could start later than s_I, but it must start before t_I, since otherwise we would pop from the queue and set the current interval to be the color of the earliest ending, non-overlapping interval with start time before the current interval. Thus there must in fact be $k + 1$ intervals intersecting over some sub-interval I' of I, a contradiction.

Complexity. We sort the intervals in $\mathcal{O}(n \lg n)$ time. We then iterate over the n intervals corresponding to the n nodes of the interval graph in $\mathcal{O}(n \lg n)$ time, as popping the minimum element from a min-heap is done in constant time and pushing a new element onto a min-heap is done in $\mathcal{O}(\lg n)$ time. Thus the total runtime of the algorithm is indeed $\mathcal{O}(n \lg n)$.