Solutions to Assignment 1 of CPSC 368/516 (Spring'23)

February 15, 2023

1 Extension of Problem 2.22

Problems 1.1. *Let $G = (V, E)$ (with $n \coloneqq |V|$ and $m \coloneqq |E|$) be an undirected, connected graph with a weight vector $w \in \mathbb{R}^E$. Consider the following algorithm for finding a maximum weight spanning tree in G.*

- *Sort the edges in nondecreasing order:*

$$w(e_1) \le w(e_2) \le \cdots \le w(e_m).$$

- *Set $T \coloneqq E$.*
- *For $i = 1, 2, \ldots, m$*
  - *If the graph $(V, T \setminus \{e_i\})$ is connected, set $T \coloneqq T \setminus \{e_i\}$.*
- *Output T.*

What is the running time of the algorithm? Is the algorithm correct? If yes prove its correctness, otherwise provide a counterexample. Assume m integers can be sorted in $O(m \log m)$ time and, given a graph $G'(V', E')$ as input, one can check if it is connected in $O(|V'| + |E'|)$ time.

This is a typical example of a greedy algorithm. There are many examples where such algorithms are not optimal, but in this case the algorithm correctly outputs an MST. We will prove this by induction. For each $i \in \{0, 1, \ldots, m\}$, let T_i denote the set T after the i-th step.

Lemma 1.2. *For every $i = 0, 1, 2, \ldots, m$, the set T_i contains some MST of G as a subset.*

Proof. We proceed by induction on i. The base $i = 0$ holds because $T_0 = E$.

Suppose the hypothesis holds for all $0 \le i < m$. Let us establish the hypothesis for $i + 1$. If the edge e_{i+1} is not removed from T_i, then there is nothing to prove. Otherwise, take any MST $T_{\text{MST}} \subseteq T_i$. If $e_{i+1} \notin T_{\text{MST}}$, then again we are done. Suppose $e_{i+1} \in T_{\text{MST}}$ and e_{i+1} gets removed from T_i.

Let $S, \overline{S} \subseteq V$ the two non-empty components obtained after removing e_{i+1} from T_{MST}. We know that $T_i \setminus \{e_{i+1}\}$ is connected, hence there is at least one edge $e' \in T_i \setminus \{e_{i+1}\}$ going between S and $\overline{S}$. This means that e_{i+1} lies on a cycle $C \subseteq T_{\text{MST}} \cup \{e'\}$. Further, note that $C \subseteq T_i$ and every edge $e \in C$ can be removed from T_i without affecting it's connectivity. This implies that $e' = e_k$ for some $m \ge k > i + 1$. (Otherwise it would have been removed earlier). Hence,

$$w(e') \ge w(e_{i+1}). \tag{1}$$

We will show that the set of edges

$$T' \coloneqq T_{\text{MST}} \setminus \{e_{i+1}\} \cup \{e'\}$$

forms a spanning tree: First, note that T' is connected because it joins S and $\overline{S}$ by e' and any path in T_{MST} only passing through vertices in S (respectively $\overline{S}$) also exists in T'. Then, since T' has the same number of edges as T_{MST} (which is a spanning tree) it follows that T' is a spanning tree.

Further, by Equation (1) it follows that T' has weight at least the weight of T_{MST}:

$$w(T') = w(T_{\text{MST}}) - w(e_{i+1}) + w(e') \ge w(T_{\text{MST}}).$$

Thus, T' is also an MST. Further, because $e' \in T_{i+1}$, $T' \subseteq T_{i+1}$. $\square$

To conclude the proof of correctness of the algorithm, it remains to prove that T_m is indeed a tree, not a bigger set of edges. First note that due to the algorithm T_m is a set of edges connecting all vertices in V. Next, suppose that T_m has a cycle $C \subseteq T_m$ and e_i is some edge in C. Then, e_i should have been removed at step i of the algorithm, thus we have a contradiction. Since T_m is acyclic set of edges connecting all vertices in V, T_m is a spanning tree.

Running time. In Step 1, the algorithm sorts m integers which by the given assumption takes $O(m \log m)$ time. The time taken by Step 2 depends on the specific implementation of the pseudo-code. Suppose that T is a copy of E, and hence, in Step 2, the algorithm copies each entry of E, since E has a bit complexity of $O(m \log n)$ (why?) this takes $O(m \log n)$ time. In the i-th iteration of Step 4, for $1 \leq i \leq m$, the algorithm checks if a graph with n vertices and $m - i$ edges is connected, which by the given assumption, takes $O(n + m - i)$ time. Hence, Step 4 in total takes $O\left(\sum_{i=1}^{m} n + m - i\right) = O(m(n + m))$ time. In Step 5, the algorithm outputs a tree T with n vertices and $n - 1$ edges and, since T has a bit complexity of at most $O(n \log n)$ (why?), this takes $O(n \log n)$ time. Hence, the total running time is $\widetilde{O}(m(n + m))$ where $\widetilde{O}$ hides a factor of $\log n$.[1]

2 Problem 2.23

Problems 2.1. *Let $T = (V, E)$ be a tree on n vertices V and let L denote its Laplacian. Given a vector $b \in \mathbb{R}^n$, design an algorithm for solving linear systems of the form $Lx = b$ in time $O(n)$.* Hint: *use Gaussian elimination but carefully consider the order in which you eliminate variables.*

Proof. We present a recursive algorithm that given a tree $T = (V, E)$ with n vertices V and Laplacian L and a vector $b \in \mathbb{R}^n$, after $O(n)$ arithmetic operations, either outputs a solution to the linear system $Lx = b$ or (correctly) outputs that there is no solution. To simplify the notation index the vertices in V by $\{1, 2, \ldots, n\}$.

Pre-processing. Before executing the recursive algorithm we identify a leaf $v \in V$ of the tree T. (Note that can be performed $O(n)$ arithmetic operations, and hence, does not affect the asymptotic running time.)

Recursive algorithm. The recursive algorithm takes a tree $T = (V, E)$, vector b, and a leaf $v \in V$ of T as input.

Base case: The base case is when T has a unique vertex, namely, v. In this case, the algorithm outputs $x = [1]$[2] if $b_1 = 0$ and outputs `infeasible` otherwise.

Recursive step: Suppose T has $1 < k \leq n$ vertices. Suppose v is connected to the tree T via the edge $\{u, v\}$, where without loss of generality, $u = k - 1$ and $v = k$. Given T, b, and v, the algorithm queries a solution to the sub-problem defined by the following parameters

- the tree T' formed by deleting the vertex v from T, i.e., $T' = (V \setminus \{v\}, E \setminus \{\{u, v\}\})$,
- the vector b' formed by subtracting the v-th entry of b from its u-th entry and then deleting the v-th entry, i.e., $b' = (b_1, b_2, \ldots, b_{n-2}, b_{k-1} - b_k) \in \mathbb{R}^{k-1}$, and
- the leaf u of T'.

If the sub-problem is infeasible, the algorithm outputs `infeasible` and otherwise, given a solution $x' \in \mathbb{R}^{k-1}$ to the sub-problem, the algorithm outputs the solution $(x'_1, x'_2, \ldots, x'_{k-1}, x'_{k-1} + b'_k)$.

Step 1 (Upper bound on running time). For any $1 \leq k \leq n$, let $R(k)$ be the maximum number of arithmetic operations the algorithm performs on given a tree of size k. In the base case, where the given tree has size $k = 1$, the algorithm performs a constant number of arithmetic operations to check if $b_1 = 0$, and hence, $R(k) = O(1)$. In the recursive step, given a tree of size $k \geq 2$, the algorithm performs $O(1)$ arithmetic

[1] $f = \widetilde{O}(g)$ is equivalent to $f = O(g \cdot (\log g)^k)$ for some constant k.

[2] In fact, it suffices to output $x = [c]$ for any number $c \in \mathbb{R}$.

operations and queries the solution to a sub-problem with a tree of size $k-1$. Hence, we have the following recursive relation for all $k \geq 2$

$$R(k) \leq O(1) + R(k-1).$$

Which, since $R(1) = O(1)$, implies that $R(k) = O(k)$ for any $k \geq 1$. Thus, the algorithm, after at most $O(n)$ arithmetic operations, either outputs a solution to the linear system $Lx = b$ or outputs `infeasible`.

Step 2 (Correctness).

Base case: In the base case, the Laplacian is the 1×1 matrix containing 0; hence, the system $Lx = b$ has a solution if and only if $b_1 = 0$. The solution is not unique: in fact, for any constant $c \in \mathbb{R}$, the vector $[c]$ is a solution. This proves the correctness of the base case.

Recursive step: Suppose v is connected to the tree T via an edge $\{u, v\}$ where, without loss of generality, the indices of u and v are $k-1$ and k respectively. Let L and L' be the Laplacians of tree T and the tree T' respectively. Let E be the $k \times k$ matrix encoding the elementary row operation that subtracts the k-th row from the $(k-1)$-th row. We claim that

$$EL = \begin{bmatrix} L' & 0_{k-1\times 1} \\ -e_{k-1}^\top & 1 \end{bmatrix} \quad \text{and]} \quad Eb = b' \tag{2}$$

where $0_{k-1\times 1}$ is the $(k-1) \times 1$ matrix containing all zeros and e_{k-1} is the $(k-1)$-dimensional vector of the form $[0, 0, \ldots, 0, 1]$. Since all elementary row operations are invertible, the set of solutions of $Lx = b$ is the same as the set of solutions of $ELx = Eb$. The correctness of the recursive step follows from the above claim and the observation that if $L'x' = b'$, then $x = \begin{bmatrix} x'^\top & x'_{k-1} + b'_k \end{bmatrix}$ is a solution to $ELx = Eb$. It remains to prove the claim in Equation (2).

Proof of the claim. The equality $Eb = b'$ holds by construction. In the other equality, the first $(k-2)$ rows of the LHS and the RHS are identical because the degrees and adjacency lists of all vertices in V except u and v are identical. The last row of the LHS and the RHS are identical due to construction. The second last row of the LHS and the RHS can be verified to be identical due to the definition of the elementary row operation E, the fact that the adjacency list u in T' is the same as its adjacency list in T except omitting the vertex v. $\square$

3 Problem 2.24

Problems 3.1. *Let $G = (V, E, w)$ be a connected, weighted, undirected graph. Let $n := |V|$ and $m := |E|$. We use L_G to denote the corresponding Laplacian. For any subgraph H of G we use the notation L_H to denote its Laplacian.*

1. *Let $T = (V, F)$ be a connected subgraph of G. Let P_T be any square matrix satisfying*

$$L_T^+ = P_T P_T^\top.$$

   Prove that

$$x^\top P_T^\top L_G P_T x \geq x^\top x$$

   for all $x \in \mathbb{R}^n$ satisfying $\langle x, 1 \rangle = 0$.

2. *Prove that*

$$\mathrm{Tr}\left(L_T^+ L_G\right) = \sum_{e \in G} w_e b_e^\top L_T^+ b_e,$$

   where b_e is the column of B corresponding to the edge e.

3. *Now let $T = (V, F)$ be a spanning tree of G. For an edge $e \in G$, write an explicit formula for $b_e^\top L_T^+ b_e$.*

4. *The weight of a spanning tree $T = (V, F)$ of $G = (V, E, w)$ is defined as*

$$w(T) \coloneqq \sum_{e \in F} w_e.$$

Prove that if T is the maximum weight spanning tree of G, then $\mathrm{Tr}\left(L_T^+ L_G\right) \leq m(n-1)$.

5. *Deduce that if T is a maximum weight spanning tree of G and P_T is any matrix such that $P_T P_T^\top = L_T^+$, then the* **condition number** *of the matrix $P_T L_G P_T^\top$ is at most $\leq m(n-1)$. The condition number of $P_T L_G P_T^\top$ is defined as*

$$\frac{\lambda_n(P_T L_G P_T^\top)}{\lambda_2(P_T L_G P_T^\top)}.$$

Here $\lambda_n(P_T L_G P_T^\top)$ denotes the largest eigenvalue of $P_T L_G P_T^\top$ and $\lambda_2(P_T L_G P_T^\top)$ is the smallest nonzero eigenvalue of $P_T L_G P_T^\top$. How large can the condition number of L_G be?

3.1 Subproblem 1

Let I^+ be the matrix that maps every vector v orthogonal to the all-ones vector $1 \in \mathbb{R}^n$ to itself and that maps the all-ones vector to 0. Concretely, define

$$I^+ \coloneqq I - \frac{1}{n} 1 1^\top.$$

We prove this result under the additional assumption that $P_T 1 = 0$. Note that if P_T is the vertex-edge incidence matrix B_T of the tree T (as in the subsequent subproblems), then $P_T 1 = 0$ holds.

Lemma 3.2. *Given any graph $T = (V, F)$ and any square matrix P_T such that $L_T^+ = P_T P_T^\top$ and $P_T 1 = 0$, it holds that*

$$P_T^\top L_T P_T = I^+.$$

Proof. Fix any graph $T = (V, F)$ and any square matrix P_T such that $L_T^+ = P_T P_T^\top$. Observe that

$$\begin{aligned}(P_T^\top L_T P_T) \cdot (P_T^\top L_T P_T) &= P_T^\top L_T L_T^+ L_T P_T && \text{(Using that } L_T^+ = P_T P_T^\top\text{)}\\ &= P_T^\top L_T P_T. && \text{(Using the definition of the pseudo-inverse)}\end{aligned}$$

Hence, $P_T^\top L_T P_T$ is an orthogonal projection matrix which implies that all of its eigenvalues are either 0 or 1. (This is because $H = P_T^\top L_T P_T$ satisfies $H^2 = H$. Since the eigenvalues of H^2 are squares of eigenvalues of H, any eigenvalue λ of H must satisfy $\lambda = \lambda^2$.) Moreover, it holds that

$$\begin{aligned}\mathrm{Tr}\left(P_T^\top L_T P_T\right) &= \mathrm{Tr}\left(P_T P_T^\top L_T\right) && \text{(Using that } \mathrm{Tr}(AB) = \mathrm{Tr}(BA) \text{ for any matrices } A \text{ and } B\text{)}\\ &= \mathrm{Tr}\left(L_T^+ L_T\right) && \text{(Using that } L_T^+ = P_T P_T^\top\text{)}\\ &= \mathrm{Tr}\left(I^+\right) && \text{(Using that } I^+ \coloneqq I - \tfrac{1}{n} 1 1^\top\text{)}\\ &= n - 1.\end{aligned}$$

Since the eigenvalues of $P_T^\top L_T P_T$ are either 0 or 1 and $\mathrm{Tr}\left(P_T^\top L_T P_T\right)$ is the sum of its eigenvalues, $P_T^\top L_T P_T$ has exactly $n-1$ eigenvalues equal to 1 and one eigenvalues equal to 0. We claim that the 0 eigenvalue corresponds to the all-ones eigenvector. This implies that $P_T^\top L_T P_T = I^+$, as required. $\square$

Fix any vector x such that $\langle x, 1 \rangle = 0$. The result follows due to the following sequence of inequalities.

$$\begin{aligned}
x^\top P_T^\top L_G P_T x &= x^\top P_T^\top \left(\sum_{e \in E} w_e b_e b_e^\top \right) P_T x \\
&= \sum_{e \in E} w_e (x^\top P_T^\top b_e)^2 \\
&\geq \sum_{e \in F} w_e (x^\top P_T^\top b_e)^2 && \text{(Using that } F \subseteq E \text{ and } w_e \geq 0 \text{ for all } e \in E) \\
&= x^\top P_T^\top L_T P_T x \\
&= x^\top I^+ x && \text{(Using Lemma 3.2)} \\
&= x^\top x. && \text{(Using that } x \text{ satisfies } \langle x, 1 \rangle = 0)
\end{aligned}$$

3.2 Subproblem 2

We can prove the required inequality as follows

$$\begin{aligned}
\operatorname{Tr}\left(L_T^+ L_G\right) &= \operatorname{Tr}\left(L_T^+ B W B^\top\right) && \text{(Using the weighted version of Lemma 2.22 from the book [1])} \\
&= \operatorname{Tr}\left(L_T^+ \left(\sum_{e \in E} w_e b_e b_e^\top \right) \right) \\
&= \operatorname{Tr}\left(\sum_{e \in E} L_T^+ w_e b_e b_e^\top \right) \\
&= \sum_{e \in E} \operatorname{Tr}\left(w_e b_e^\top L_T^+ b_e \right) && \text{(Using the linearity of trace)} \\
&= \sum_{e \in E} w_e b_e^\top L_T^+ b_e. && \text{(Using the fact that for any scalar } x \in \mathbb{R},\ \operatorname{Tr}(x) = x)
\end{aligned}$$

3.3 Subproblem 3

In this section, we give the following formula for $b_e^\top L_T^+ b_e$ in terms of certain paths on the tree T.

Lemma 3.3. *If $e \in E$ is an edge with endpoints $i, j \in V$ and $P \subseteq F$ is the unique path from vertex i to vertex j in the tree T, then*

$$b_e^\top L_T^+ b_e = \sum_{f \in P} \frac{1}{w_f}.$$

The proof of the above lemma relies on the following observation, which can be verified by explicit computation.

Lemma 3.4. *For any spanning tree $T = (V, F)$ of the graph $G = (V, E)$, any edge $e = \{i, j\} \in F$ that is a part of the spanning tree T, and the following vector b_e*

$$v \in V, \quad (b_e)_v \coloneqq \begin{cases} -1 & \text{if } v = i, \\ +1 & \text{if } v = j, \\ 0 & \text{otherwise,} \end{cases}$$

one solution to the linear system $L_T x = b_e$ is as follows

$$\forall v \in V, \quad x_v^{(e)} \coloneqq \begin{cases} 0 & \text{if } v \in T_{i,e} \\ \frac{1}{w_e} & \text{if } v \in T_{j,e}. \end{cases} \tag{3}$$

Where $T_{i,e}$ and $T_{j,e}$ are the two connected subgraphs of T formed by removing edge e, where $T_{i,e}$ contains i and $T_{j,e}$ contains j.

Proof of Lemma 3.3. Fix any orientations of the edges such that all edges in the path P point from "vertex i to vertex j." Suppose the edge e also points from vertex i to vertex j. For this orientation, the following equality holds

$$b_e = \sum_{f \in P} b_f. \tag{4}$$

This equality and Lemma 3.4, imply the following

$$\begin{aligned}
b_e^\top L_T^+ b_e &= b_e^\top L_T^+ \left(\sum\nolimits_{f \in P} b_f\right) && \text{(Using Equation (4))}\\
&= \sum_{f \in P} b_e^\top L_T^+ b_f \\
&= \sum_{f \in P} b_e^\top x^{(f)} && \text{(Using Lemma 3.4 and the fact that } f \text{ is part of the spanning tree } T\text{)}\\
&= \sum_{f \in P} \frac{1}{w_f}
\end{aligned}$$

(Using that for any $f \in P$, $i \in T_{i,f}$, and $i \in T_{j,f}$, the definition of b_e, and the definition of $x^{(f)}$)

□

3.4 Subproblem 4

We need to upper bound the following quantity

$$\operatorname{Tr}\left(L_T^+ L_G\right) = \sum_{e \in E} w_e b_e^\top L_T^+ b_e.$$

If the edge e is a part of the tree T, then from the previous subproblem it follows that

$$w_e b_e^\top L_T^+ b_e = 1. \tag{5}$$

Suppose now that edge e is not a part of the tree T. Let $P \subseteq T$ be the path in T between the endpoints of e. Since T is a maximum weight spanning tree, no edge f in the path P can have a weight smaller than w_e (otherwise, one can obtain a spanning tree of a larger weight by removing the edge f from T and adding the edge e), and hence,

$$w_e b_e^\top L_T^+ b_e \overset{\text{Lemma 3.3}}{=} \sum_{f \in P} \frac{w_e}{w_f} \leq |P| \leq n-1. \tag{6}$$

Thus, it follows that

$$\operatorname{Tr}\left(L_T^+ L_G\right) = \sum_{e \in E} w_e b_e^\top L_T^+ b_e \overset{(5),(6)}{=} \sum_{e \in E} (n-1) \overset{|E|=m}{\leq} m(n-1). \tag{7}$$

3.5 Subproblem 5

Note that in Subproblem 1, we showed smallest non-zero eigenvalue of $P_T^\top L_G P_T$ is at least 1, i.e.,

$$\lambda_2(P_T^\top L_G P_T) \geq 1. \tag{8}$$

Moreover, since $P_T^\top L_G P_T$ is a PSD matrix, all of its eigenvalues are non-negative, and hence, $\operatorname{Tr}\left(P_T^\top L_G P_T\right)$ is an upper bound on the largest eigenvalue of $P_T^\top L_G P_T$. This gives us the following upper bound on $\lambda_n(P_T^\top L_G P_T)$

$$\begin{aligned}
\lambda_n(P_T^\top L_G P_T) &= \operatorname{Tr}\left(P_T^\top L_G P_T\right)\\
&= \operatorname{Tr}\left(P_T P_T^\top L_G\right) && \text{(Using that } \operatorname{Tr}(AB) = \operatorname{Tr}(BA) \text{ for any matrices } A \text{ and } B\text{)}\\
&= \operatorname{Tr}\left(L_T^+ L_G\right)\\
&\overset{(7)}{\leq} m(n-1).
\end{aligned} \tag{9}$$

Thus, Equations (8) and (9) imply that

$$\frac{\lambda_n(P_T^\top L_G P_T)}{\lambda_2(P_T^\top L_G P_T)} \leq m(n-1). \tag{10}$$

Note that this condition number is independent of the weights of the edges in G. In contrast, the condition number of L_G can be a function of the edge weights and its magnitude can be as large as the ratio $\frac{\max_e w_e}{\min_e w_e}$. As a concrete example, consider a graph G with three vertices $V = \{1, 2, 3\}$ and two edges $\{\{1, 2\}, \{2, 3\}\}$. Let edge $e_1 \coloneqq \{1, 2\}$ have weight ε and edge $e_2 \coloneqq \{2, 3\}$ have weight $\frac{1}{\varepsilon}$. One can verify that the condition number of G, is $\Omega\left(\varepsilon^{-2}\right)$–where as the upper bound in Equation (10) is 4 for any $\varepsilon > 0$.

4 Problem 2.26

Problems 4.1. *A matrix is said to be* **totally unimodular** *if each of its square submatrices has determinant* 0, +1, *or* −1. *Prove that, for a graph $G = (V, E)$, any of its vertex-edge incidence matrices B is totally unimodular.*

Proof. Fix any graph $G = (V, E)$ and any vertex-edge incidence matrix B of G. Let $n = \min\{|V|, |E|\}$. We will prove by induction that B every square submatrix S of B of size $1 \leq s \leq n$ has determinant 0, +1, or −1. Note that this proves the required claim.

The base case is holds as when $s = 1$, the determinant of S is equal to the corresponding entry of B and all entries of B are 0, +1, or −1.

Suppose that the induction hypothesis holds for some $1 \leq k \leq n-1$, i.e., all square submatrices of B of size k have determinant 0, +1, or −1. Consider any $(k+1) \times (k+1)$ square submatrix S of B. Since each column of S has at most 2 non-zero entries, one of the following three cases holds.

- **Case I (S has a column with all zeros):** In this case, the determinant of S is 0.
- **Case II (S has a column with exactly one non-zero):** Suppose the j-th column of S has exactly one non-zero entry S_{ij} at the i-th row. Let T be the $k \times k$ matrix formed by deleting the i-th row and j-th column in S. By the induction hypothesis, the determinant of T is either 0, +1, or −1. Since the determinant of S is S_{ij} times the determinant of T and $S_{ij} \in \{0, -1, +1\}$, and hence, the determinant of S is either 0, +1, or −1.
- **Case III (All columns of S have exactly two non-zeros):** In this case, all columns of S have exactly one +1 entry and one −1 entry, and hence, the sum of all rows of S is 0 which, in turn, implies that the determinant of S is 0.

It follows that the induction hypothesis holds at $k+1$. Hence, if the induction hypothesis holds at k, then the induction hypothesis also holds at $k+1$. Thus, since the induction hypothesis holds at $k = 1$, by the principle of mathematical induction it follows that every square submatrix of B has determinant 0, +1, or −1, and hence, B is totally unimodular.

□

References

[1] Nisheeth K. Vishnoi. *Algorithms for Convex Optimization.* Cambridge University Press, 2021.