CS101 Algorithms and Data Structures
Fall 2022
Homework 8

Due date: 23:59, November 20th, 2022

1. Please write your solutions in English.
2. Submit your solutions to gradescope.com.
3. Set your FULL name to your Chinese name and your STUDENT ID correctly in Account Settings.
4. If you want to submit a handwritten version, scan it clearly. `CamScanner` is recommended.
5. When submitting, match your solutions to the problems correctly.
6. No late submission will be accepted.
7. Violations to any of the above may result in zero points.

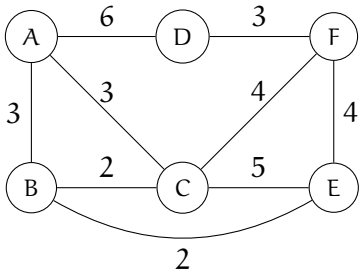
1. (?? points) Multiple Choices

Each question has **one or more** correct answer(s). Select all the correct answer(s). For each question, you will get 0 points if you select one or more wrong answers, but you will get 1 point if you select a non-empty subset of the correct answers.

Write your answers in the following table.

(a)	(b)	(c)	(d)
ABD	C	C	AB

(a) (3') Suppose we use the Prim's algorithm to find the minimum spanning tree of the following graph. Choose all possible sequences of edges added to the minimum spanning tree.



A. {A, C}, {C, B}, {B, E}, {C, F}, {F, D}

B. {A, B}, {B, C}, {B, E}, {C, F}, {F, D}

C. {A, C}, {C, B}, {C, F}, {B, E}, {F, D}

D. {A, B}, {B, E}, {B, C}, {E, F}, {F, D}

(b) (3') Which of the following statements is/are true?

A. The time complexity of the Prim's algorithm with a Fibonacci heap is always asymptotically better than that with a binary heap.

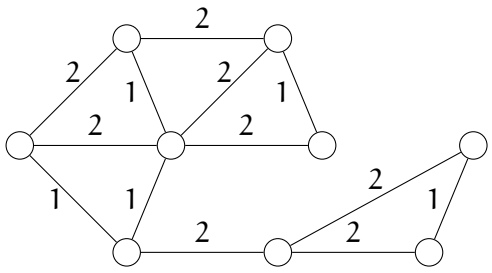
B. The time complexity of the Prim's algorithm using adjacency list and binary heap is always better than that using adjacency matrix without a priority queue.

C. The minimum spanning tree of a graph is unique if all the edges have distinct weights.

D. The time complexity of the Kruskal's algortihm is $O(|E|\alpha(|V|))$ if we use the disjoint-sets with union-by-rank optimization and path-compression optimization.

E. If T is a minimum spanning tree obtained by performing the Prim's algorithm starting with vertex v, then for any vertex u the path on the tree T connecting u and v is the shortest path from u to v in the graph.

(c) (3') How many different minimum spanning trees does the following graph have?



A. 4 B. 5 **C. 6** D. 7

(d) (3') Suppose $G = (V, E)$ is an undirected connected graph and that T is a minimum spanning tree of G. Define $w(e)$ to be the weight of e for $e \in E$. Which of the following statements is/are true?

A. If $C \subseteq E$ is a cycle in G and $e \in C$ is an edge on the cycle such that

$$\forall f \in C \setminus \{e\}, \quad w(e) > w(f),$$

then e does not belong to T.

B. Let $V = X \cup Y$ be a partition of V such that $X \cap Y = \varnothing$. Define

$$C(X, Y) = \{\{u, v\} \in E \mid u \in X, v \in Y\}.$$

If $e \in C(X, Y)$ is an edge such that

$$\forall f \in C(X, Y) \setminus \{e\}, \quad w(e) < w(f),$$

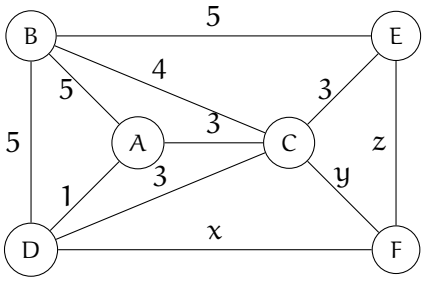
then e must belong to T.

C. Suppose $T' \neq T$ is another minimum spanning tree of G. Let $w_0 \in \{w(e) \mid e \in T\}$ be the weight of some edge in T. Let m be the number of edges weighted w_0 in T. Then T' may contain less than m edges weighted w_0.

D. If $e \in E$ is an edge that has the largest weight among all edges in E, then e cannot belong to T.

2. (?? points) Minimum Spanning Tree

Consider the following weighted undirected graph.

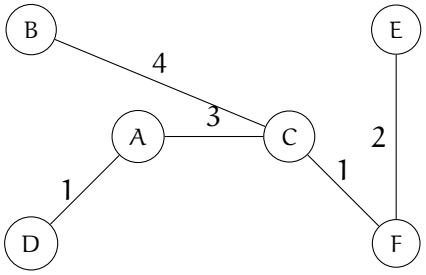


(a) (3') Suppose $(x, y, z) = (4, 1, 2)$ and that we use the Kruskal's algorithm to find a minimum spanning tree of the graph. To make your answer unique and clear, please follow the rules below.

- Use (u, v) to represent an undirected edge $\{u, v\}$, where $u < v$.
- Edges with same weight are sorted in alphabetical order. If two edges $e_1 = (u, v)$ and $e_2 = (w, t)$ have the same weight, e_1 appears before e_2 in the edge list if $(u < w) \vee ((u = w) \wedge (v < t))$.

Write down the sequence of edges added into the minimum spanning tree, and draw the tree.

Solution:
(A, D)
(C, F)
(E, F)
(A, C)
(B, C)



(b) (3') If $(x, z) = (2, 3)$, for what values of y is the edge $\{C, F\}$ guaranteed to be contained in a minimum spanning tree? Give a sufficient and necessary condition and briefly justify your answer.

Solution:
the sufficient and necessary condition is $y < 3$
proof:
cosider the cycle made up by the edges $(C, D), (C, F), (D, F)$ the edge with biggest

weight must not be in the MST, so we have to make sure y is not the biggest weight in the cycle. and since $weight(C, D) = 3, wight(D, F) = x = 2$, so $y < 3$
similarly, cosider the cycle made up by the edges $(C, E), (C, F), (E, F)$ the edge with biggest weight must not be in the MST, so we have to make sure y is not the biggest weight in the cycle. and since $weight(C, E) = 3, wight(E, F) = z = 2$, and $(C, F) mustbeinthecycle$,so $y < 3$
so above all, the values of y is $y < 3$.

(c) (2') If $5 \notin \{x, y, z\}$, is it possible for an edge weighted 5 to appear in a minimum spanning tree? Briefly justify your answer.

Solution:
It is impossible.
proof:
we can make the graph into two partitions.
one set is {A,B,C,D,E}, and the other one is {F}
for the first set, there exist an unique MST combined with edges
$(A, C), (A, D), (B, C), (C, E)$ and there weights are $1, 3, 4, 3$, so the MST for the first set do not include edge weighted 5.
since the second set only include one vertex, so we do not need to consier its MST.
to connected the two sets, and generate the MST of the whole graph, we need to take some edges which are connecting two sets.
which means we will take the some of the weights in {x,y,z}.
and since $5 \notin \{x, y, z\}$, so the wight(s) we will take $\neq 5$
so above all, it is impossible to have an edge weighted 5 in the MST.

3. (?? points) Algebraic Geometry

Liu Big God, who loves pure math, has bought n books on algebraic geometry, the i-th of which has price a_i, $i = 1, \cdots, n$. He will give his students some books to arouse their interest in pure math. For each student, Liu Big God is going to give him/her **one or two** books with total price not exceeding P.

Liu Big God is not going to keep any of these books, because he has read all of them. He wants to send all these books to students. What is the minimum number of students that can receive books?

It is guaranteed that $0 \leqslant a_i \leqslant P$ for every $i = 1, \cdots, n$. You should come up with a greedy algorithm with time complexity $O(n \log n)$.

(a) (3') Description of your algorithm in **pseudocode** or **natural language**.

(b) (4') Proof of correctness of your algorithm.

(c) (2') Time complexity.

Solution: (a) algorithm:
firstly sort the price array in the decending order.
we can use the two pointers to get the answer. So we need to set a right pointer, which story the right-most book that has not been given out. So the initial of the right pointer is at n
And we use the left pointer to go through the sorted sequence from left to right.
If the value of the book that the left pointer pointing to plus the value of the book that the right pointer pointing to is not exceeding P, then we will give the two books to one student, and move backward the left pointer, move forward the right point.
otherwise, we will just give the book that left pointer pointing to to the student, and move the backward the left pointer.
At the time the left pointer and the right pointer are pointing to the same book, we will give the book to a student and finish giving books.
If the right pointer is on the left of the left pointer, we will immediately stop giving books because that means all books have been given out.

(b) correctness:
mark the answer we get is A, suppose there exist an optimal solution O which can give the books to less students than A.
and we can write down the solution A as a pair sequence
$(a_1, b_1), (a_2, b_2), \cdots, (a_k, b_k)$
where a_i, b_i are the value of the books that the ith student has recived.
and let a_i be the more expensive book. If ith student only recived one book, then $b_i = 0$.
similarly, we can also write the solution of O as
$(a'_1, b'_1), (a'_2, b'_2), \cdots, (a'_l, b'_l)$
they have the same explanation with A.
since we consider the O is optimal, then $l \leq k$.
Assume the first $m - 1$ pairs are the same of the 2 algorithms, which means
$(a_m, b_m) \neq (a'_m, b'_m)$
there are three possibilities:

1. $a_m = a'_m, b_m \neq b'_m$
2. $a_m \neq a'_m, b_m = b'_m$
3. $a_m \neq a'_m, b_m \neq b'_m$

based on our A's description, a_m is the most value in the rest of the books, b_m is the least value in the rest of the books(if $b_m \neq =$).

1. according to the analize above, $b_m < b'_m$, so we can just let b'_m change to b_m, the sequence is still legal, then $(a_1, b_1), (a_2, b_2), \cdots, (a_m, b_m)$ is exactly same as $(a'_1, b'_1), (a'_2, b'_2), \cdots, (a'_m, b'_m)$

2. according to the analize above, $a_m > a'_m$, so we can just let a'_m change to a_m, the sequence is still legal, then $(a_1, b_1), (a_2, b_2), \cdots, (a_m, b_m)$ is exactly same as $(a'_1, b'_1), (a'_2, b'_2), \cdots, (a'_m, b'_m)$

3. according to the analize above, $a_m > a'_m$ and $b_m < b'_m$, so we can just let a'_m change to a_m and b'_m change to b_m, the sequence is still legal, then $(a_1, b_1), (a_2, b_2), \cdots, (a_m, b_m)$ is exactly same as $(a'_1, b'_1), (a'_2, b'_2), \cdots, (a'_m, b'_m)$

Also, if there exist two pairs in O such as $(a_{k_1}, b_{k_1}), (a_{k_2}, b_{k_2})$
and $a_{k_1} > a_{k_2}$, but $b_{k_1} > b_{k_2}$
we can just swap the b_{k_1} and b_{k_2}, since $a_{k_1} > a_{k_2}$
so pairs $(a_{k_1}, b_{k_2}), (a_{k_2}, b_{k_1})$ are still legal, and that is same as what A is doing.

so above all, no matter which case above, the optimal O is the same with the our solution A.
so the greedy solution A is the optimal.

(c) time complexity:
the sort of the price array a takes the time of $\Theta(n \log n)$
and during the processing of giving out the books, each book has only been visit by one pointer once, so it takes time of $\Theta(n)$
so the whole time complexity is $\Theta(n \log n + n) = O(n \log n)$
so above all, the time complexity is $O(n \log n)$

4. (?? points)

Given a set of $n \geqslant 3$ distinct positive numbers $S = \{s_1, s_2, \cdots, s_n\}$, we want to find a permutation $A = \langle A_1, \cdots, A_n \rangle$ of S, where $A_i \in S$ for all $i \in \{1, \cdots, n\}$, such that

$$f(A) = A_1^2 + \sum_{i=2}^{n} (A_i - A_{i-1})^2$$

is maximized.

(a) (3') Describe your algorithm that finds the permutation A for which $f(A)$ is maximized. Use **pseudocode** or **natural language**.

(b) (4') Prove the correctness of your algorithm by showing that your choice on the value of A_1 is optimal, i.e. any other choice would not lead to a better solution.

(c) (2') Time complexity. Your algorithm should be $O(n \log n)$.

Solution:

(a) algorithm:

sort the sequence S and we can get an ascending sequence a.

and since S's elements are distinct, so $a_1 < a_2 < \cdots < a_n$.

and let $A_1 = a_n, A_2 = a_1, A_3 = a_{n-1}, A_4 = a_2, \cdots$

let A be staggered put the maximum and minimum values of a. in this way, we can get the maximum $f(A)$.

(b) proof:

if we swap a_1 with any ohter elements in the sequence $a_i (i \neq 1)$ and get a new permutation A'

if $i = 2$, then $f(A) - f(A') = 2A_3(A_1 - A_2)$, since $A_3 > 0, A_1 = a_n = max\{a_1, \cdots, a_n\}$, so $f(A) > f(A')$

if $i \neq 1$ and $i \neq 2$ and $i \neq n$, then $f(A) - f(A') = 2(A_1 - A_i)(A_{i-1} + A_{i+1} - A_2)$, since $A_1 = a_n = max\{a_1, \cdots, a_n\}, A_2 = a_1 = min\{a_1, \cdots, a_n\}$, so $A_1 - A_i > 0, A_{i-1} + A_{i+1} - A_2 > 0$ so $f(A) > f(A')$

if $i = n$, then $f(A) - f(A') = (A_1 - A_n)(A_1 + A_n - 2A_2 + 2A_{n-1})$, since $A_1 = a_n = max\{a_1, \cdots, a_n\}, A_2 = a_1 = min\{a_1, \cdots, a_n\}$, so $A_1 - A_n > 0, 2A_{n-1} - 2A_2 > 0$ so $f(A) > f(A')$

so above all, if the rest of the permutation is the same, swap any of the elements with A_1, the $f(A)$ cannot be bigger.

so $A_1 = a_n$ is optimal.

(c) time complexity:

to sort the sequence S takes the time complexity of $\Theta(n \log n)$

and calculating the function $f(A)$ takes time of $\Theta(n)$ if we can regard that the numbers are small enough to compute the multiplication and addition operations in $O(1)$ time.(i.e. high precision calculation is not required)

so the whole time complexity is $\Theta(n \log n + n) = O(n \log n)$

so above all, the time complexity is $O(n \log n)$

5. (?? points) Discovery

(a) (1') Let $G = (V, E)$ be an unweighted undirected graph where $V = \{v_1, \cdots, v_n\}$ and $E = \{e_1, \cdots, e_m\}$. For simplicity we assume there are no multiple edges (i.e. two or more edges incident to the same two vertices). Let $D \in \mathbb{R}^{n\times n}$ be the *degree matrix* whose (i, j)-th entry is

$$d_{ij} = \begin{cases} \deg(v_i), & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

Let $A \in \mathbb{R}^{n\times n}$ be the *adjacency matrix* of G, whose (i, j)-th entry is

$$a_{ij} = \begin{cases} 1, & \text{if } \{v_i, v_j\} \in E, \\ 0, & \text{if } \{v_i, v_j\} \notin E. \end{cases}$$

Note that $\deg(v_i) = \sum_{j=1}^{n} a_{ij}$. The matrix $L = D - A$ is the *Laplacian matrix*. Prove that L is positive semidefinite. (Hint: Try to show that $x^T Lx \geqslant 0$ holds for every $x \in \mathbb{R}^n$.)

Solution: since the graph is undirected, so the matrix have the property that
$D^T = D$, $A^T = A$
and $L^T = (D - A)^T = D^T - A^T = D - A = L$
so $L^T = L$
$\forall x \in \mathbb{R}^n$, we can write $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$
so $x^T Lx = x^T(D - A)x = x^T Dx - x^T Ax$
and $x^T Ax = (x_1, \cdots, x_n) \begin{pmatrix} a_{11}, \cdots, a_{1n} \\ \vdots \ddots \vdots \\ a_{n1}, \cdots, a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j a_{ij}$
so $x^T Lx = \sum_{i=1}^{n} d_{ii} x_i^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j a_{ij}$
$= \frac{1}{2}(\sum_{i=1}^{n} d_{ii} x_i^2 - 2\sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j a_{ij} + \sum_{j=1}^{n} d_{jj} x_j^2)$
and since $d_{ii} = deg(v_i) = \sum_{j=1}^{n} a_{ij}$
so $x^T Lx = \frac{1}{2}(\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i^2 + 2\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j + 2\sum_{j=1}^{n} \sum_{i=1}^{n} a_{ji} x_j^2)$
$= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}(x_i - x_j)^2$
≥ 0
so above all, $x^T Lx \geq 0$ $\forall x \in \mathbb{R}^n$.
so L is positive semidefinite.

(b) (0') STFW (Search The Friendly Web) about how the Laplacian matrix is related to the number of spanning trees of a graph.

Solution:
After searching the friendly web, we could discover that there exist a theorem called *Matrix-tree Theorem*.
With a long and complicated proof.
It is a wonderful proof, but the blank space is too little, and I am lazy to copy it.
And it is described as:
the undirected graph $G = (V, E)$ with its Laplacian matrix L.
the number of G's spanning tree is $\det(L_0)$
where L_0 is the submatrix of L obtained by removing the ith row and the ith column of L.(i could be any number).

furthermore, since the Laplacian matrix of the graph is mentioned, I will take some small notes by the way.

- The number of eigenvalues which is equal to 0 is the number of connected regions of the graph.
- the complete graph K_n with n vertices. Then the number of its spanning trees is n^{n-2}
- The sum of each line of L is 0.
- the eigenvalues of L are nonnegetive, and at least have one 0.