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.

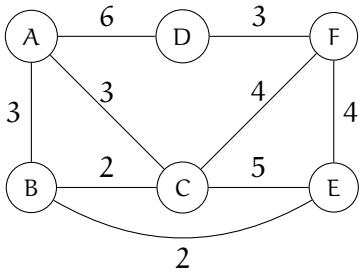
1. (12 points) Multiple Choices

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.

Solution: Time complexity of the Prim's algorithm:

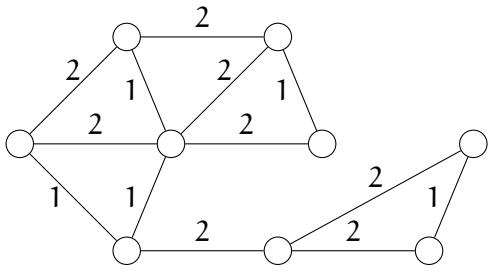
- Adjacency matrix without priority queue: $\Theta(|V|^2 + |E|)$
- Adjacency list with binary heap: $\Theta(|E| \log |V|)$
- Adjacency list with Fibonacci heap: $\Theta(|E| + |V| \log |V|)$

When $|E| = \Theta\left(|V|^2\right)$ the second one is less preferable than the first one. When $|E| = \Theta(|V|)$ the second one and the third one are asymptotically equal.

The Kruskal's algorithm requires sorting the edges first, which is the bottleneck in terms of time complexity.

Minimum spanning tree vs "shortest-path tree".

(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.

Solution:

A. The cycle property.

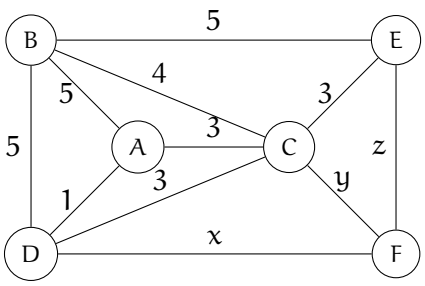
B. The cut property.

C. A graph may have multiple minimum spanning trees, but the multiset of weights of edges in minimum spanning trees is unique. STFW for a proof of this proposition.

D. Consider the case when $|E| = |V| - 1$, i.e. the graph is a tree itself.

2. (8 points) Minimum Spanning Tree

Consider the following weighted undirected graph.

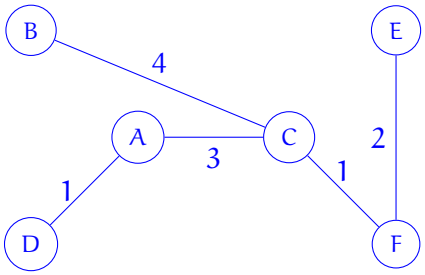


(a) (3') Suppose $(x, y, z) = (4, 1, 2)$ and that we use the Kruskal's algorithm to find a minimum spanning tree of the graph. To make your answer unique and clear, please follow the rules below.

- 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: Sequence: $(A, D), (C, F), (E, F), (A, C), (B, C)$.



(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: $y < 3$. When $y < 3$, it follows from the *cut property* if we look at the cut induced by $\{C\} \cup \{A, B, D, E, F\}$. When $y = 3$, the tree $\{\{A, D\}, \{D, F\}, \{F, E\}, \{C, E\}, \{B, C\}\}$ is a minimum spanning tree that does not contain $\{C, F\}$. When $y > 3$, the *cycle property* on the cycle $C - F - E$ ensures that $\{C, F\}$ is not in any MST.

(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: No. $\{B, D\}$ is the unique heaviest edge on the cycle $B - C - D$. $\{A, B\}$ is the unique heaviest edge on the cycle $A - B - C$. $\{B, E\}$ is the unique heaviest edge on

the cycle $\mathsf{B} - \mathsf{C} - \mathsf{E}$. From the *cycle property* these edges are impossible to appear in a minimum spanning tree.

3. (9 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:

```
Sort the array a in ascending order
ans ← 0
(i, j) ← (1, n)
while i ⩽ j do
    ans ← ans + 1
    if a_i + a_j ⩽ P then                ▷ The student gets the i-th and j-th books.
        (i, j) ← (i + 1, j − 1)
    else                                 ▷ The student gets the j-th book.
        j ← j − 1
    end if
end while
return ans
```

Proof of correctness. Consider the subproblem of minimizing the number of students receiving books from $\{a_i, \cdots, a_j\}$, where $1 \leqslant i \leqslant j \leqslant n$.

- If $a_i + a_j \leqslant P$, we consider the following cases and show that a strictly better solution cannot be achieved if the choice of our algorithm at this step is not followed.
  1. A student receives the book indexed i and another student receives the books indexed j and k, for some $k \in (i, j)$. For any solution obtained in this way, we can transform it by giving the first student the k-th book and the second student the books indexed i and j, leading to a solution that can be obtained via our algorithm.
  2. A student receives the book indexed j and another student receives the books indexed i and k, for some $k \in (i, j)$. This is similar to case 1.

3. A student receives the books indexed i, k and another student receives the books indexed j, ℓ, for some $k, \ell \in (i, j)$. By swapping the k-th and the j-th book, any solution in this case can be transformed into an equally preferable one that can be obtained by our algorithm.
4. A student receives the i-th book, and another student receives the j-th book. This is obviously worse than a solution that our algorithm can yield.

- If $a_i + a_j > P$, then $a_k + a_j > P$ holds for every $k \in [i, j)$ since the elements in a are in ascending order. This means that the j-th book can only be given alone to a student, which is the choice of our algorithm.

Hence we conclude that our algorithm eventually yields the optimal solution. □

Time complexity: $O(n \log n)$ if we use an $O(n \log n)$ sorting algorithm.

4. (9 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: W.L.O.G. suppose $s_1 < s_2 < \cdots < s_n$. We can show that

$$A = \langle s_n, s_1, s_{n-1}, s_2, s_{n-2}, \cdots \rangle$$

is the maximizer of $f(A)$.

Proof of correctness. We will prove by contradiction that $A_1 = s_n$ leads to an optimal solution. Assume that $A_1 = s_k \neq s_n$ for some $k \in \{1, \cdots, n-1\}$.

- If $s_n = A_n$, the sequence A is

  $$A = \langle A_1 = s_k, A_2, A_3, \cdots, A_{n-1}, A_n = s_n \rangle,$$

  with the objective function

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

  We can construct another sequence A' such that

  $$A' = \langle s_n = A_n, A_{n-1}, A_{n-2}, \cdots, A_2, s_k = A_1 \rangle,$$

  with

  $$f(A') = s_n^2 + \sum_{i=2}^{n} (A_i - A_{i-1})^2 = s_n^2 + f(A) - s_k^2 > f(A).$$

  So A cannot be the optimal solution.

- If $s_n = A_p$ for some $p < n$, suppose $A_{p+1} = s_q$ and the sequence A is

  $$A = \langle A_1 = s_k, A_2, \cdots, A_p = s_n, A_{p+1} = s_q, \cdots, \rangle.$$

  We have that

  $$f(A) = s_k^2 + \sum_{i=2}^{p} (A_i - A_{i-1})^2 + (s_q - s_n)^2 + R$$

where $R = \sum_{i=p+2}^{n} (A_i - A_{i-1})^2$. We can construct another sequence A' such that

$$A' = \langle s_n = A_p, A_{p-1}, A_{p-2}, \cdots, A_2, s_k = A_1, s_q = A_{p+1}, \cdots \rangle$$

with

$$f(A') = s_n^2 + \sum_{i=2}^{p} (A_i - A_{i-1})^2 + (s_q - s_k)^2 + R.$$

Since

$$\begin{aligned} f(A') - f(A) &= s_n^2 + (s_q - s_k)^2 - s_k^2 - (s_q - s_n)^2 \\ &= 2s_q (s_n - s_k) > 0, \end{aligned}$$

A is not the optimal solution.

Hence we conclude that in the optimal solution $A_1 = s_n$. □

The sequence A is easily obtained after sorting the given data $s_1, \cdots, s_n$ in ascending order. The time complexity of our alogrithm is $O(n \log n)$ if we use an $O(n \log n)$ sorting algorithm.

5. (1 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: Pick any $x \in \mathbb{R}^n$ and we have that

$$\begin{aligned} x^T Lx &= x^T Dx - x^T Ax \\ &= \sum_{i=1}^{n} x_i^2 \deg(v_i) - \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_i x_j \\ &= \sum_{i=1}^{n} x_i^2 \sum_{j=1}^{n} a_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_i x_j \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\left(x_i^2 - x_i x_j\right). \end{aligned}$$

Note that by symmetry, this must be equal to

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\left(x_j^2 - x_i x_j\right).$$

Therefore

$$\begin{aligned} x^T Lx &= \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\left(x_i^2 - x_- x_j\right) + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\left(x_j^2 - x_i x_j\right)\right) \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\left(x_i^2 + x_j^2 - 2x_i x_j\right) \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}(x_i - x_j)^2 \geqslant 0. \end{aligned}$$

Remark: Another way to prove this is to use the *incidence matrix*.

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

Solution: The Kirchhoff's theorem.