Solution 3: Greedy Algorithms

Problem 1. Recall that a set of vectors $\{v_1, v_2, \ldots, v_n\}$ is called **linearly independent** if the only solution to the equation

$$c_1v_1 + c_2v_2 + \cdots + c_nv_n = 0$$

is $c_1 = c_2 = \cdots = c_n = 0$. In other words, the vectors are linearly independent if no vector in the set can be written as a linear combination of the others.

A **spanning set** is a collection of vectors that can be combined through linear combinations to generate every element of the vector space.

Finally, a **basis** is a linearly independent set that spans the vector space. This implies that every vector in the space can be expressed uniquely as a linear combination of the basis vectors.

Let M be an $n \times m$ matrix. Let R be the set of rows of M. Let ℓ be a subset of $\mathcal{P}(R)$ such that

$$\ell = \{A : A \subset R, A \text{ linearly independent}\}.$$

(a) Show that (R, ℓ) is a matroid.

Hint: It might be helpful to consider components.

(b) Let $w : \ell \to \mathbb{R}^+$ be a function assigns to each subset $A \subset R$ its cardinality, i.e., $w(A) = |A|$. To be consistent with the lecture notes, we also define w as a function of R: $w(v) = 1$ for every $v \in R$. Observe that $w(A) = |A| = \sum_{v \in A} w(v)$.

Let IsLinearlyIndependent be a predicate that checks whether a subset of R is linearly independent or not. Assume this algorithm runs in $O(T)$ time.

Design an efficient algorithm to find a basis of the vector space spanned by R. Prove the correctness/optimality of your algorithm and analyze an asymptotically tight runtime.

Solution.

(a) To show that (R, ℓ) is a matroid we have to verify that

(1) R is a finite set.

(2) For every $B \in \ell$, if $A \subseteq B$ then $A \in \ell$.

(3) For every $A, B \in \ell$, if $|A| < |B|$ then there exists an $x \in B \setminus A$ such that $A \cup \{x\} \in \ell$.

The first condition clearly holds since R is the set of rows of a finite matrix ($|R| = n < \infty$).

The second condition also holds since any subset of a linearly independent set is linearly independent. Suppose, for the sake of contradiction, that a vector can be written as a linear combination of the others in A. Then the same vector can clearly be written as a linear combination in B.

We now verify the final condition. Let $A, B \in \ell$ be two linearly independent subsets of R such that $|A| < |B|$. Suppose, for the sake of contradiction, that for every $v \in B \setminus A$, the set $A \cup \{v\}$ is linearly dependent. Then $B \setminus A \subset \text{span}(A)$. Moreover, $A \subset \text{span}(A)$.

Hence, $B \subset \text{span}(A)$. We know that $\text{span}(A)$ is at most $|A|$-dimensional, which means that any linearly independent set of $\text{span}(A)$ will have at most $|A|$ elements. However, B is a linearly independent set of $\text{span}(A)$ with $|B| > |A|$ elements, which is a contradiction. Thus, there exists $v \in B \setminus A$ such that $A \cup \{v\}$ is linearly independent.

(b) We'll modify the algorithm presented in lectures 5 and 6 using the knowledge that (R, ℓ, w) is a weighted matroid.

Algorithm 1 Basis of the vector space spanned by R

```
Input: A set of vectors, $R$
A weight function, $w : R \to \mathbb{R}^+$
Output: Basis of the vector space spanned by $R$
function BASIS($R$, $w$)
    Initialize $A$ to be an empty set
    for $v \in R$ do
        if IsLinearlyIndependent($A \cup \{v\}$) then
            Add $v$ to $A$
        end if
    end for
    return $A$
end function
```

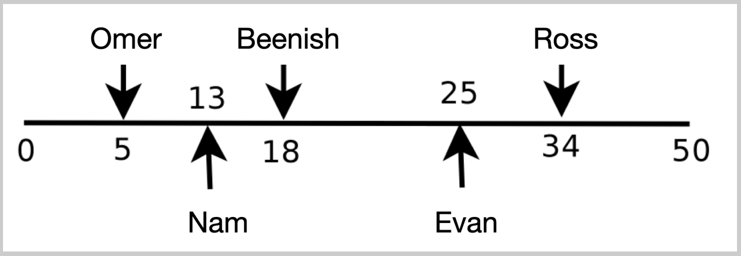
(c) The correctness of this algorithm in the case that (R, ℓ, w) is a weighted matroid was shown in class!

We loop through each of the n vectors in the set and perform an $O(T)$ check to see if the resulting set is linearly independent. Therefore, the overall runtime of this algorithm is $O(Tn)$.

□

Problem 2. The power lines along a country road have been modified to carry broadband Internet. Wi-Fi towers are being built along the road to provide the community with internet. To find the minimum number of towers required so that each house is sufficiently close to at least one tower, we model the problem as follows:

a) The entire course staff has taken up residence on Algorithm Street. The diagram below shows where they all live on the street.



Omer lives 5 miles down the road, Nam lives 13 miles down the road, and so on. Wi-Fi towers have an effective radius of 5 miles. Determine the minimum number of Wi-Fi towers needed such that each staff member has internet, and give the locations for these towers as well.

b) We're given a line segment ℓ, a set of non-negative numbers N that represents the locations of customers on ℓ, and a distance d. We wish to find a set of Wi-Fi towers of minimal size on ℓ such that each location in N is at most d away from some tower. Give an efficient greedy algorithm that returns a minimum size set of points. Prove its correctness and justify its runtime.

Now we generalize our model to account for houses that are not by the side of the road.

c) We're given a line segment ℓ, a set of pairs N representing the locations of customers, and a distance d. For each pair $(x, y) \in N$, let $x \in [0, \infty)$ be the distance along ℓ and $y \in [-d, d]$ be the distance above or below ℓ. We wish to find a set of Wi-Fi towers of minimal size on ℓ such that each location in N is at most distance d from some tower (here, we are using Euclidean distance).

Modify your algorithm from part (b) to solve this variation of the problem. You do not need to prove its correctness, but please explain how your proof from part (b) would (or would not) need to change based on your modifications.

Can we generalize further?

d) Does the correctness of your algorithm depend on the fact that ℓ is a line segment and not some curve? If so, give an example that illustrates the problem with your algorithm when ℓ is a curve. If not, explain how your algorithm could handle a curve. You shouldn't be writing another algorithm, or modifying your existing algorithm, just explain your reasoning.

Solution. a) We position 3 Wi-Fi towers within the ranges [8, 10], [20, 23], and [29, 39] such that no two are placed within the same interval.

b) We will use a greedy algorithm to place the towers. The idea is to place each tower at the farthest possible location that covers the most uncovered customers.

Sort the customer locations in N in increasing order. Let $N = \{n_1, n_2, \ldots, n_k\}$ where $n_1 \leq n_2 \leq \cdots \leq n_k$. Begin with the first customer in the sorted list. Since this customer is uncovered, place a tower at the farthest point that can still cover this customer. Specifically, place the tower at $n_1 + d$ (since the tower covers up to a distance d). After placing a tower, skip all customers that are within d units of the current tower. Then repeat the process for the next uncovered customer. Terminate when all customers are covered (end of list reached).

Below is the pseudocode:

Algorithm 2 Minimum Number of Wifi Towers

```
Input: A set of nonnegative numbers $N$ representing locations of customers and
   nonnegative distance $d$
Output: Minimum number of wifi towers guaranteeing full coverage
function findLocations($N$, $d$)
    Sort($N$)
    towers $\leftarrow 0$
    $i \leftarrow 0$
    $n \leftarrow$ length($N$)
    while $i < n$ do
        towers $+= 1$
        tower_location $\leftarrow N[i] + d$
        while $i < n$ and $N[i] \leq$ tower_location $+d$ do
            $i += 1$
        end while
    end while
    return towers
end function
```

Proof of correctness: We argue correctness using the greedy exchange argument.

Let G be the greedy solution in sorted order, and let g_i be the position of the i-th tower placed by the greedy algorithm. Similarly, let O be any optimal solution in sorted order, and let o_i be the position of the i-th tower placed by the optimal solution.

Suppose $G \neq O$; that is, G and O differ in their tower placements. Let t be the smallest index such that $g_t \neq o_t$. Up to index $t-1$, the greedy solution and the optimal solution place towers at the same positions: $g_i = o_i$ for all $i < t$. Let $a \in N$ be the leftmost customer not covered by the first $t-1$ towers (i.e., by $g_1, \ldots, g_{t-1}$). Since G and O agree on the first $t-1$ towers, a is the same for both solutions.

The greedy algorithm places the t-th tower at $g_t = a + d$. The coverage interval of this tower is $[a, a+2d]$.

The optimal solution places the t-th tower at $o_t \neq g_t$. Since o_t must cover customer a, it must satisfy $o_t \in [a-d, a+d]$.

Replace o_t with g_t in O to form a new solution O' without increasing the number of towers or decreasing coverage. All customers to the left of a are already covered by the first $t-1$ towers in both G and O. Since $g_t \geq o_t$, g_t covers all the customers to the right of a that o_t covers.

We have now decreased the number of differences between G and O by performing the exchange. By iterating this exchange we can turn O into G without impacting the quality of the solution. Therefore, G must be optimal.

Since any optimal solution O can be transformed into the greedy solution G through a series of exchanges that do not increase the number of towers or reduce coverage, the greedy solution G must be optimal.

Runtime Justification: Sorting the list of customer locations takes $O(n \log n)$, where $n = |N|$. Placing the towers and moving through the list takes $O(n)$, since we go through the list of customers exactly once.

Thus, the overall time complexity of the algorithm is $O(n \log n)$.

c) **Algorithm description**: For each customer (x_i, y_i), calculate $h_i = \sqrt{d^2 - y_i^2}$. Then determine the interval $[l_i, r_i] = [x_i - h_i, x_i + h_i]$. This interval represents all possible positions along ℓ where a tower can be placed to cover customer i. Then sort the intervals $[l_i, r_i]$ in increasing order of their right endpoints r_i. Create an empty list S to store tower positions. While there are intervals not yet covered, select the interval with the earliest right endpoint. Let $[l_i, r_i]$ be the interval with the smallest right endpoint r_i. Place a tower at position $s = r_i$ and add s to S. Remove all intervals $[l_j, r_j]$ where $l_j \leq s$. These are the intervals that are covered by the tower at s. Finally, return the size of S.

Below is the pseudocode:

Algorithm 3 Generalized Minimum Number of Wifi Towers

```
Input: A set of pairs $N$ representing locations of customers where $(x, y) \in N$ has
    $x \geq 0$ and $y \in [-d, d]$ and nonnegative distance $d$
Output: Minimum number of wifi towers guaranteeing full coverage
function FINDGENERALIZEDLOCATIONS($N$, $d$)
    Initialize list Intervals
    for $(x_i, y_i) \in N$ do
        $h_i \leftarrow sqrt(d^2 - y_i^2)$
        $l_i \leftarrow x_i - h_i$
        $r_i \leftarrow x_i + h_i$
        Add interval $[l_i, r_i]$ to Intervals
    end for
    Sort Intervals in increasing order of right endpoints $r_i$
    Initialize empty list $S$
    while Intervals is not empty do
        Let $[l_i, r_i]$ be the first interval in Intervals
        Place a tower at position $s = r_i$
        Add $s$ to $S$
        Remove all intervals $[l_j, r_j]$ from Intervals where $l_j \leq s$
    end while
    return $|S|$
end function
```

Modification of the Proof from Part (b): The proof from part (b) needs to be adjusted for part (c) because customers are no longer points on the line but have coverage intervals along the line segment due to their positions above or below it. In part (c), each customer defines an interval of feasible tower positions where they can be covered, turning the problem into an interval covering problem. Therefore, the exchange argument must account for these intervals rather than fixed points. The modified proof demonstrates that placing towers at the right endpoints of the earliest finishing intervals (as per the greedy algorithm) remains optimal. The exchange moves involve replacing towers in any optimal solution with those chosen by the greedy algorithm, ensuring coverage of the same or more intervals without increasing the number of towers. Thus, while the core structure of the proof—the exchange argument—remains the same, it adapts to consider intervals instead of individual points to establish the optimality of the modified algorithm.

d) The correctness of the algorithm depends on the fact that ℓ is a line segment. Consider the unit circle. If a customer is located at $(-1, 0)$, the algorithm from part c) would attempt to place a Wi-Fi tower as far to the right of the customer as possible while staying within the allowed distance d. However, since the circle has two locations (one on the upper semicircle and one on the lower semicircle) that are equidistant from the customer, the algorithm could place the tower in either direction, potentially missing other customers or having to place additional towers. This ambiguity does not arise with a line segment because each location has only one direction that extends to the rightmost distance without looping back. Thus, the correctness of the algorithm in c) relies on ℓ being a line, and the approach may not generalize correctly for non-linear curves.

$\square$