Practice Problems 5

Problem 1. Suppose you are given a set S of nine points equally spaced along the real line at x-coordinates $S = \{0, 1, 2, \ldots, 8\}$. Suppose you run Gonzalez's algorithm on this set with $k = 9$. The algorithm starts with the leftmost point ($x = 0$), and generally, whenever there are multiple possible points that can be selected next, take the one that is farthest to the left (having the smallest value).

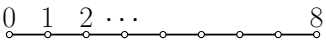

Figure 1: Gonzalez's algorithm on a 1-dimensional point set.

List the nine points in the order in which Gonzalez's algorithm selects. (Fun fact: This can be applied to any point set in any dimension, and it yields a permutation of the set, which is known as the *greedy permutation*.)

Problem 2. Recall that in the weighted-interval scheduling problem, we are given a set of n requests, each of which is an interval, $R = \{[s_1, f_1], \ldots, [s_n, f_n]\}$ of the requests' start and finish times. Let us make the simplifying assumption that there are no duplicate start or finish times. When the algorithm starts, it is assumed that the requests are sorted by increasing order of finish times, $f_1 < f_2 < \cdots < f_n$. The algorithm made use of an array prior$[1..n]$, which is defined to be the largest integer such that $f_{\text{prior}[j]} < s_j$, that is, prior$[j]$ is latest request by finish times that does *not* overlap request j. If there is no such request, prior$[j]$ is defined to be 0.

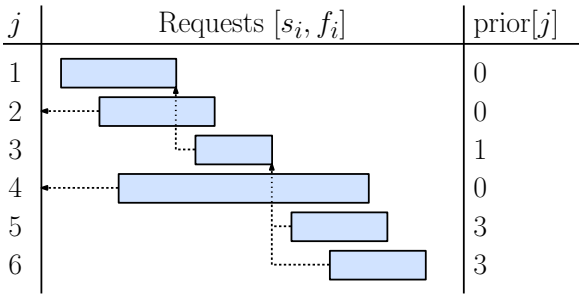

Figure 2: Example of the prior$[1..n]$ array.

Present an efficient algorithm, which given the start and finish times $s[1..n]$ and $f[1..n]$, assumed to be sorted in increasing order by finish times, computes the values of prior$[j]$, for $1 \le j \le n$. For full credit, your algorithm should run in time $O(n \log n)$. (Note that $O(n^2)$ is trivial.) Explain your algorithm and derive its running time. (Hint: This can be done using the data structures taught in CMSC 420, it can also be done assuming nothing more than a sorting algorithm.)

Problem 3. Consider the weighted interval scheduling instance shown in Fig. 3. Show the result of executing the algorithm given in class for this problem.

In particular, show the final contents of the following:

- The contents of the prior$[1..n]$ array.
- The contents of the $W[0..n]$ array.
- The contents of the accept$[1..n]$ array.

Finally, indicate what the final optimal schedule is, as determined by the algorithm. (Throughout, if there are ties, favor the one as given by the algorithm given in class.)

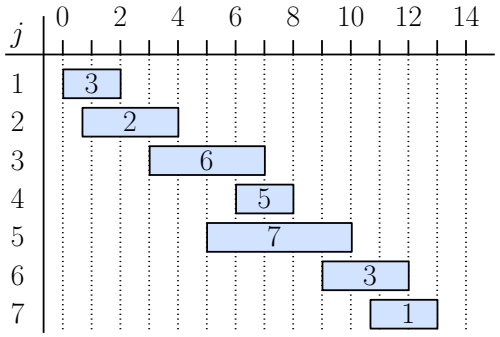

Figure 3: Weighted interval scheduling.

Problem 4. Gonzalez's algorithm can be applied to any *metric*, that is, any distance function that satisfies the properties of positivity, symmetry, and triangle inequality. (See the lecture on the k-center problem for formal definitions.)

Suppose that you are running Gonzalez's algorithm on a distance function that fails to be a metric because it does not satisfy symmetry nor triangle inequality. But your (non-metric) distance function does satisfy the following weaker properties:

Relaxed symmetry: Given any two points x, y, $\text{dist}(x, y) \leq 2 \cdot \text{dist}(y, x)$.

Relaxed triangle inequality: Given any three points x, y, z, $\text{dist}(x, z) \leq 1.5(\text{dist}(x, y) + \text{dist}(y, z))$.

Show that Gonzalez's algorithm is still a constant-factor approximation algorithm to the k-center problem for such a distance function, but possibly with a different approximation factor (that is, different from 2). What is the best approximation factor that results from your analysis? (Hint: Work through the analysis given in class, and make adjustments wherever symmetry and triangle inequality are used.)

Problem 5. In some applications of the set cover problem, we require that each element be covered more than once. (For example, when guarding the paintings in a art gallery, each painting should be visible from at least two security cameras, in case one fails.)

Recalling the notation from the lecture on the set-cover problem, given a set system (X, S) of elements $X = \{x_1, \ldots, x_n\}$ and sets S is a collection of subsets of X, we say that a subset

$S' \subseteq S$ is a *2-cover* of S if every element of X is contained within at least two sets of S'. We assume that there are sufficient sets in S that a 2-cover exists.

As with the standard set-cover problem, there is a natural greedy heuristic for the 2-cover problem. Initially, the elements of X are not covered at all. Each time we add a set to the cover, we count the number of elements of X that are either newly covered, or that were in one set, and are now in two sets. (Once a set is in at least two sets, it is no longer counted.) We take the set of S that has the highest count of elements, and add it to the 2-cover. When all the elements of X have 2-covered, the algorithm terminates.

Modify the proof of the set-cover algorithm given in class, to prove the following theorem.

Theorem: Given any set system $\Sigma = (X, S)$, let G be the output of the above greedy heuristic, and let O be any 2-cover of X. Then $|G| \leq 1 + (|O| \cdot \ln(2n))$, where $n = |X|$.