CS180 Final Exam

Due: 11:29 am PDT, June 9
Please submit on gradescope

For all the algorithms you design, in addition to describe your algorithm clearly, please also (a) briefly justify the correctness of the algorithm; (b) present the time complexity of the algorithm and briefly justify the reason. Partial credits will be given if your algorithm has complexity slightly worse than the solution for all the problems.

1. (20 pt) Given an undirected connected graph where each edge is associated with a positive weight, we want to find a set of edges such that removing those edges will make the graph acyclic. Design an algorithm to find such edge set with the smallest total weight. The algorithm should run in $O((m+n)\log n)$ time.

2. (25 pt) In this problem, our goal is to design sublinear time algorithms for finding a "hill" in a given 1D or 2D array. We say an element in a 1D or 2D array is a "hill" if and only if its value is larger than all its neighbors. In 1D array the neighbors for $A[i]$ are $A[i-1]$ and $A[i+1]$ and in 2D the neighbors for $A[i,j]$ are $A[i-1,j], A[i+1,j], A[i,j-1], A[i,j+1]$. Elements on the boundary of arrays will have less neighbors, for instance $A[0]$ only has one neighbor $A[1]$; $A[0,0]$ only has two neighbors $A[0,1], A[1,0]$. An array could have multiple hills, and we only need to find one of them. Figure 1 illustrates two examples, one in 1D and another in 2D.

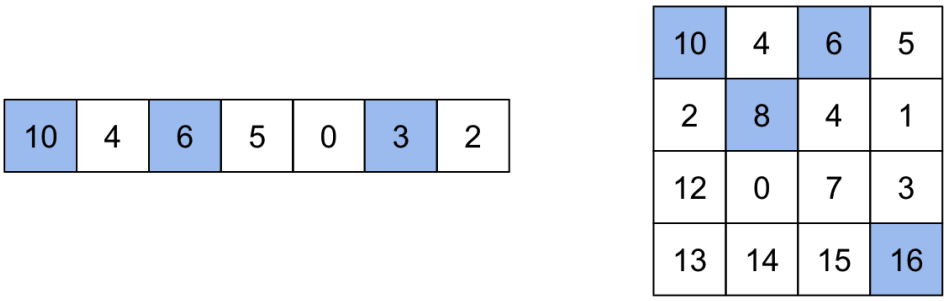

Figure 1: The right panel illustrates a 1D example and the left panel illustrates a 2D example, where the blue cells are hills. There could be multiple hills and our goal is to find one of them.

(a) (10 pt) Given a 1D integer array of size n and assume the values are distinct. Design an algorithm to find a hill in $O(\log n)$ time.

(b) (15 pt) Now we extend the algorithm to find a hill in a 2D array of size $n \times n$. Design an algorithm to return the position of one of the hills in $O(n)$ time. Partial credits will be given to algorithms with slightly higher complexity, for instance, a solution with time complexity $O(n \log n)$ will get 10 points.

3. (25 pt) There are n cities on a highway with coordinates $x_1, \ldots, x_n$ and we aim to build $K < n$ fire stations to cover these cities. Each fire station has to be built in one of the cities, and we hope to minimize the average distance from each city to the closest fire station. Please give an algorithm to compute the optimal way to place these K fire stations. The algorithm should run in $O(n^2K)$ time. Partial credits will be given to algorithms with slightly higher complexity, for instance, a solution with time complexity $O(n^3K)$ will get 15 points.

4. (30 pt) Decision tree is an important model for binary classification. Given an input binary string $\boldsymbol{x} = x_1x_2\ldots x_d$, each x_i denotes a binary attribute of an input instance (e.g., in practice an input instance could be a document, an image, or a job application). A decision tree tries to map this string to a prediction value based on a tree structure—starting from root node, at each node we decide going left or right by the value of an attribute x_i; and at each leaf node will assign either $+1$ or -1 to the input. A decision forest consists of multiple decision trees, and the final prediction value is the sum of all these predictions. If we use $f_t(\boldsymbol{x})$ to denote the prediction value of the t-th tree and assume there are in total T trees, the final prediction of the decision forest is

$$\begin{cases} True & \text{if } \sum_{t=1}^{T} f_t(\boldsymbol{x}) \geq 0 \\ False & \text{otherwise.} \end{cases}$$

For example, Figure 2 illustrates a decision forest and the prediction values for several input strings.

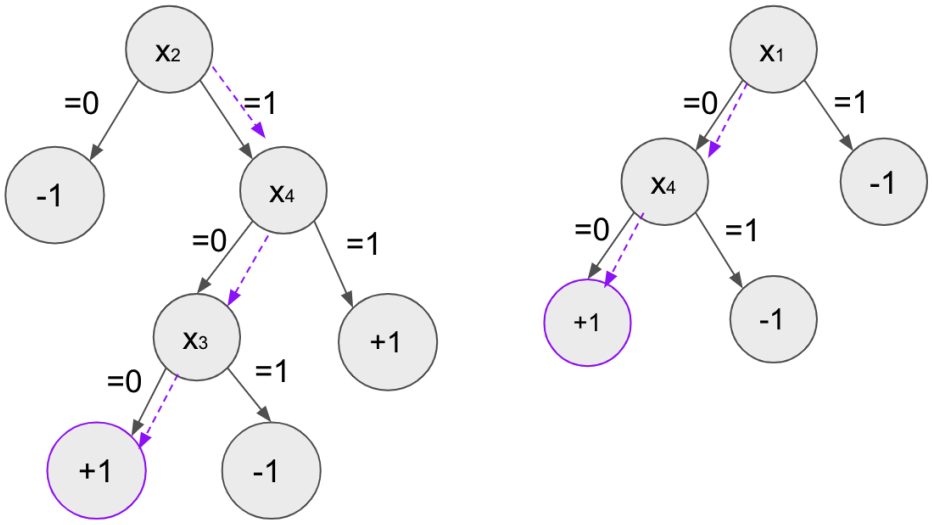

Figure 2: A decision forest. For input $x_1x_2x_3x_4 = 0100$ it wil traverse the trees based on the dashed arrow, so the first tree outputs $+1$, the second tree output $+1$, and the final output is True. For the same decision forest, the input $x_1x_2x_3x_4 = 0011$ will produce -2, thus False.

An important property for a machine learning model is that the model can't always produce the same output. Therefore, we want to solve the **Forest-Verify** problem such that given a decision forest, determine whether there exists a d-dimensional input binary string $\boldsymbol{x}$ such that the prediction of this decision forest is $True$. (The same procedure can also detect whether there exists an input to produce $False$).

Show the **Forest-Verify** problem is NP-complete.

(a) (7 pt) Show the Forest-Verify problem belongs to NP.

(b) (7 pt) Let's first assume there's only one Clause in 3-SAT, can you turn this into a single decision tree such that the prediction of Decision tree corresponds to the value of this Clause?

(c) (16 pt) Derive a polynomial time reduction from 3-SAT to Forest-Verify.