ECE 374 B - Fall 2023

Homework 6

- **Submit your solutions electronically on the course Gradescope site as PDF files.** If you plan to typeset your solutions, please use the LaTeX solution template on the course web site. If you must submit scanned handwritten solutions, please use a black pen on blank white paper and a high-quality scanner app (or an actual scanner, not just a phone camera). We will mark difficult to read solutions as incorrect and move on.
- **Every homework problem must be done *individually*.** Each problem needs to be submitted to Gradescope before 6AM of the due data which can be found on the course website: https://ecealgo.com/homeworks.html.
- For nearly every problem, **we have covered all the requisite knowledge required to complete a homework assignment prior to the "assigned" date.** This means that there is no reason not to begin a homework assignment as soon as it is assigned. Starting a problem the night before it is due a recipe for failure.

Policies to keep in mind

- **You may use any source at your disposal**—paper, electronic, or human—but you ***must*** cite ***every*** source that you use, and you ***must*** write everything yourself in your own words. See the academic integrity policies on the course web site for more details.
- **Being able to clearly and concisely explain your solution is a part of the grade you will receive.** Before submitting a solution ask yourself, if you were reading the solution without having seen it before, would you be able to understand it within two minutes? If not, you need to edit. Images and flow-charts are very useful for concisely explain difficult concepts.

See the course web site (https://ecealgo.com) for more information.

If you have any questions about these policies,
please don't hesitate to ask in class, in office hours, or on Piazza.

Extra Instructions Solutions to a dynamic programming problem have (at minimum) three things:

- A recurrence relation
- A *brief* description of what your recurrence function represents and what each case represents.
- A *brief* description of the memory element/storage and how it's filled in.

---

1. **Largest Square of 1's** You are given a $n \times n$ bitonic array A and the goal is to find the set of elements within that array that form a square filled with only 1's.

$j \rightarrow$

$i \downarrow$

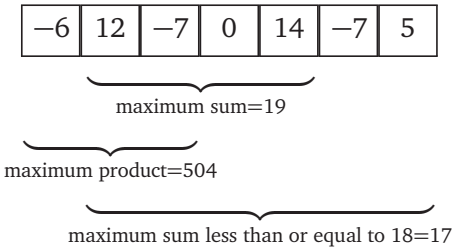
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Figure 1: Example: The output is the side-length of the largest square of 1's (4 in the case of the graph above, yes there can be multiple squares of the greatest size).

2. Suppose you are given an array $A[1..n]$ of arbitrary real numbers. Recall a *subarray of an array A* is by definition a *contiguous* subsequence of A. Define the sum and product of an empty array to be 0 and 1, respectively. For any array $A[i..j]$ where $i \le j$, define its sum and product to be

$$\sum_{k=i}^{j} A[k] \quad \text{and} \quad \prod_{k=i}^{j} A[k],$$

respectively. For the sake of analysis, assume that comparing, adding and multiplying any pair of numbers takes $O(1)$ time.



(a) Describe and analyze an algorithm to compute the *maximum sum* of any subarray of A. For example, given $A = [-6, 12, -7, 0, 14, -7, 5]$, your algorithm should return 19 as illustrated above.

(b) Describe and analyze an algorithm to compute the *maximum product* of any subarray of $A[1..n]$. For example, given $A = [-6, 12, -7, 0, 14, -7, 5]$, your algorithm should return 504 as illustrated above.

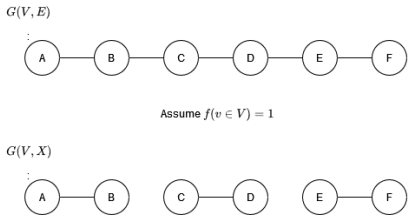
(c) Suppose you are also given an arbitrary integer $X \ge 0$. Describe and analyze an algorithm to compute the *maximum sum* of any subarray of *A less than or equal to X*. For example, given $A = [-6, 12, -7, 0, 14, -7, 5]$ and $X = 18$, your algorithm should return 17 as illustrated above.

(d) Describe a faster algorithm for part (c) when every element in the array A is nonnegative.

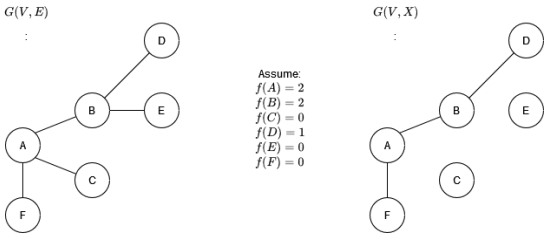
3. We are given a undirected tree $T = (V, E)$ with n vertices. Assume that the degree (number of edges connected to a vertex) of all the vertices in T is at most 3. You are given a function $f : V \to \{0, 1, 2, 3\}$. The task is to compute a subset X of edges, such that for every node $v \in V$, there are at least $f(v)$ distinct edges in X that are adjacent to v.

   Describe an algorithm, as fast as possible, that computes the minimum size set $X \subseteq E$ that meets the needs of all the nodes in the tree. The algorithm should output both $|X|$ and X itself.

   Some Examples:



(a) Example 1



(b) Example 2

4. Plum blossom poles are a Kung Fu training technique, consisting of n large posts partially sunk into the ground, with each pole pi at position (xi, yi). Students practice martial arts techniques by stepping from the top of one pole to the top of another pole. In order to keep balance, each step must be more than d meters but less than 2d meters. Give an efficient algorithm to find a safe path from pole ps to pt if it exists.