Practice Problems 8

Problem 1. In your new job for a major chip manufacturer, you are tasked to help processing defects in the fabrication process. A fabricated chip is a square of some given side length L. After fabrication, it is tested for defects. Let us assume that the defects take the form of a set of points $P = \{p_1, \ldots, p_n\}$ (see Fig. 1(a)).

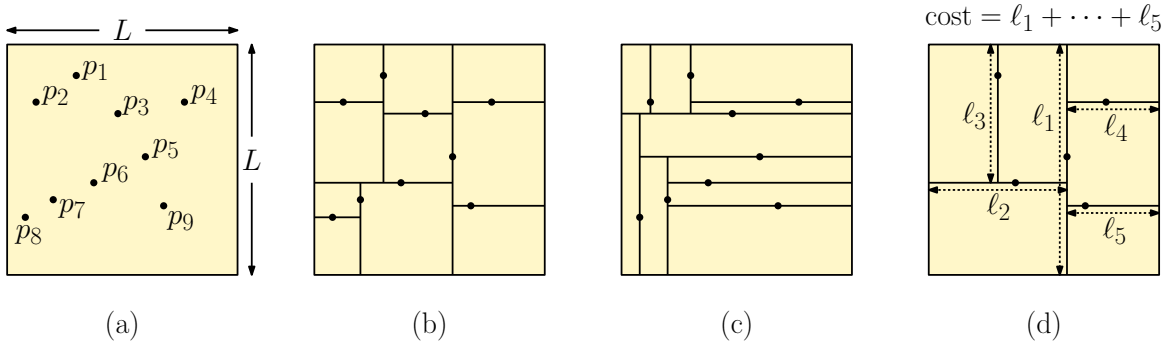

Figure 1: Cutting away defects in a fabricated chip.

Since we cannot sell chips with defects, we need to cut them out. The laser cutting tool makes a horizontal or vertical cut through a defect point that runs the entire length of chip. This splits the chip into two smaller chips. The cutting process is then repeated on each of these smaller chips. This is repeated until all the defects are cut out. The resulting set of defect-free rectangles are called *subchips* (see Fig. 1(b)). This is called a *hierarchical cutting*.

There are many different hierarchical cuttings for a given set of defects. To maximize profits, we want the sum of lengths of laser cuts to be as small as possible. (For example, Fig. 1(c) shows another valid cutting, but uses longer cuts than (b).) Define the *total cost* of a hierarchical cutting to be the sum of lengths of all cuts (see Fig. 1(d)).

The objective of this problem is to devise an efficient DP algorithm, which given an $L \times L$ chip and set P of defect points, computes a hierarchical cutting that minimizes the total cost.

(a) Throughout, let us assume that, among the defect points, there are no duplicate x-coordinates nor duplicate y-coordinates. Prove that if the number of defects is n, then the number of subchips in any hierarchical cutting is $n + 1$.

(b) Derive a DP formulation, which given a set $P = \{p_1, \ldots, p_n\}$ of defects, determines the best (that is, minimum) total cost of any hierarchical cutting. Your formulation should be expressed as a recursive function. Be sure to include the basis case(s) for the recursive function, and indicate what initial function call provides the global answer. Justify the correctness of your formulation.

(Hint: The subproblems are associated with rectangles of the original image, which can arise through any valid hierarchical cutting process. It may be helpful to presort the x- and y-coordinates of the defect points.)

Problem 2. A *probabilistic digraph* is a digraph $G = (V, E)$ where, $V = \{1, \ldots, n\}$. Each edge $(i, j) \in E$ is associated with a probability $p(i, j)$, that this edge exists, where $0 \leq p(i, j) \leq 1$. (Nonexistent edges G can be handled by setting $p(i, j) = 0$.) Given any sequence of vertices $\pi = \langle v_0, v_1, \ldots, v_k \rangle$, the *probability* that this path exists in G is the product of edge probabilities $\prod_{i=1}^{k} p(v_{i-1}, v_i)$. In this problem, we will consider a problem called the *all-pairs max probability path problem*.

(a) Present an algorithm that, given G, computes for each pair of vertices i and j, the maximum probability path from i to j. (Rather than computing the path itself, it suffices to compute the probability of the maximum probability path.) Briefly justify your algorithm's correctness and derive its running time.

Hint: Modify the Floyd-Warshall algorithm. You may assume that G is presented as an adjacency matrix. It suffices to give the DP formulation, rather than full pseudocode. The running time, if implemented, should be $O(n^3)$.

(b) Show that it is possible to modify the weights in the digraph G, forming a new digraph G' such that, by running the original Floyd-Warshall algorithm (from class), it is possible to easily convert the output into the solution to the all-pairs maximum probability problem.

Problem 3. Let $G = (V, E)$ be a DAG. Let us assume that G is presented as an adjacency matrix where $\text{adj}[i, j] = 1$ if $(i, j) \in E$ and 0 otherwise. Let us assume that $V = \{1, \ldots, n\}$. For every $i, j \in V$, define an $n \times n$ matrix $C[1..n, 1..n]$, where $C[i, j]$ is the number of distinct paths from i to j. Present an algorithm that computes this matrix.

Hint: (Same hint as Problem 2(a).)

Problem 4. In this problem we will consider network flows involving networks with vertex capacities, rather than edge capacities. You are given a directed s-t network $G = (V, E)$, in which each vertex $u \in V \setminus \{s, t\}$ (that is, every vertex excluding the source and sink) is associated with a nonnegative *capacity*, denoted $c(u)$ (see Fig. 2(a)). We call this a *vertex-capacitated network*. A *flow* in G is defined the same as for a standard network except the capacity constraint applies to the flow into of each vertex (excluding s and t), that is,

$$\forall u \in V \setminus \{s, t\}, \quad \sum_{(w,u) \in E} f(w, u) \leq c(u)$$

As with standard flows, excluding s and t, the flow into the vertex must equal the flow out of the vertex (see Fig. 2(b)).

We assert that having vertex capacities is essentially the same as having edge capacities. To show this, answer the following two questions.

(a) Explain how to modify any vertex-capacitated network G into an equivalent edge-capacitated network G' so that, for any flow in G there exists a flow of equivalent value in G', and vice versa.

(b) Explain how to modify any edge-capacitated network G into an equivalent vertex-capacitated network G' so that, for any flow in G there exists a flow of equivalent value in G', and vice versa.

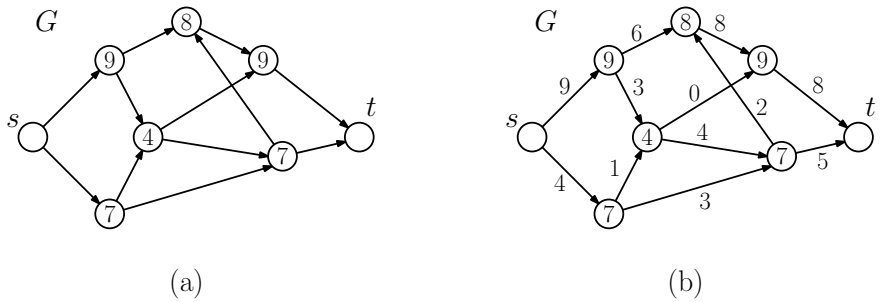

Figure 2: (a) A network with capacities on the vertices and (b) a valid flow.

Your answers should first explain how to convert G into G'. (**Hint:** The conversions are both very simple and are implementable in $O(m+n)$ time. They do *not* involve computing flows, residual networks, or cuts.) Next, show for any flow f in G, there exists a flow f' in G' of equal value, and vice versa. If done carefully, these proofs can be quite long. Instead, it is fine to give a brief explanation of your construction, and then provide a few figures illustrating your conversion and how flows in G correspond to flows in G'.