

1. a) Consider the following optimization problem:

INPUT: An undirected graph  $G = (E, V)$ , a partition of the set of edges  $E$  into subsets  $E_1, E_2, \dots, E_k$ , and a weight function for the edges  $w : E \rightarrow \mathfrak{R}$ .

OUTPUT: A maximum weight subset of the edges  $S \subseteq E$  such that there is at most one edge in  $S$  contained in each subset  $E_i$  (i.e.,  $\forall i, |S \cap E_i| \leq 1$ ).

Define a subset system that describes the set of valid (but not necessarily maximum weight) solutions to this problem. Prove that this subset system is a matroid.

b) Give an efficient algorithm for this problem. Analyze its running time, and demonstrate that it always provides an optimal solution.

c) Given an undirected graph  $G = (E, V)$ , and a subset of the vertices  $R$ , consider the problem of determining whether or not there exists a spanning tree  $T$  for  $G$  such that every vertex in  $R$  is a leaf of the tree  $T$ . Use your matroid from part (a) to demonstrate that there exists a polynomial time algorithm that decides whether such a tree  $T$  exists or not. You do not have to describe the algorithm or its running time.

2. Problem 22-2 from [CLR] page 459.

3. Consider the following paragraphing problem:

INPUT: a text description in the form of a sequence  $W = (w_1, w_2, \dots, w_n)$  of word lengths, and a maximum line length  $L$ , where  $\forall i, 1 \leq w_i \leq L$ .

OUTPUT: an optimal decomposition of the text into lines, where each pair of adjacent words on a line is separated by a single space and the total length of a line may not exceed  $L$ .

An optimal decomposition is one of minimum cost, where the cost of a line is the amount of additional blank space that it contains. Thus, for example, if a line contains the words  $w_i, w_{i+1}, \dots, w_{i+j}$ , its cost is  $L - \sum_{k=0}^j w_{i+k} - j$ . The cost of a decomposition of  $W$  into lines is the **maximum** cost of any line in the decomposition.

a) Provide a greedy algorithm for the paragraphing problem, and give a small example demonstrating that it doesn't always provide the optimal solution.

b) Describe a dynamic programming algorithm that computes the cost of an optimal decomposition in time  $O(n \cdot m(W, L))$ , where  $m(W, L)$  is the maximum number of words that could possibly appear on any line. (Hint: let  $C(i)$  be the cost of an optimal solution for the last  $i$  words of  $W$ . What is a recursive expression for  $C(i)$ ? Note that  $C(0) = 0$ , and  $C(n)$  is the optimal cost.)

c) Describe how to modify the algorithm so that it also computes the actual decomposition that achieves the optimal cost.

d) Describe how the algorithm would have to be modified if the cost of a decomposition were the **sum** of the line costs.

4. In Lecture 8, we saw a dynamic programming algorithm the all pairs shortest paths problem that runs in time  $O(|V|^3)$ . In Lecture 9, we saw a faster algorithm that uses matrix multiplication to solve this problem faster for the special case of an unweighted, undirected graph. In this question,

we shall develop a faster algorithm for the more general case of directed graphs with integer weights. This algorithm is also based on matrix multiplication. We shall use the convention that the weight of an edge that is not present in a graph is  $\infty$ .

a) For two  $n \times n$  matrices  $A$  and  $B$  of positive integers and  $\infty$ , define the matrix  $A \oplus B$  by

$$[A \oplus B](i, j) = \min_{1 \leq k \leq n} (A(i, k) + B(k, j)).$$

Show that the all pairs shortest paths problem where all weights are integers can be solved in time  $O(S(|V|) \log |V|)$ , where  $S(n)$  is the time required to perform the operation  $\oplus$  on a pair of  $n \times n$  matrices.

b) For an  $n \times n$  matrix  $A$  of positive integers, and a positive integer  $s$ , define the matrix  $A^{(s)}$  such that  $A^{(s)}(i, j) = 2^{-sA(i, j)}$ . Show how  $A \oplus B$  can be computed in  $O(n^2)$  time, given the ordinary matrix product  $A^{(s)}B^{(s)}$ , provided that  $s > \log_2 n$ .

c) Construct an algorithm for the all pairs shortest paths problem that runs in time  $O(\mu(|V|) \log |V|)$ , where  $\mu(n)$  is the time required to multiply two  $n \times n$  real matrices. You can assume that arithmetic operations on arbitrary reals are performed in constant time.

d) Comment on the practical utility of the above algorithm.