CS 331: Algorithms and Complexity (Spring 2024)

Unique numbers: 50930/50935/50940/50945

Handout for Test 2

Recurrences and Master's Theorem

Solve the recurrences below by Master's Theorem where possible. If Master's theorem does not apply, mention why.

1. $T(n) = \sqrt{2}T(n/2) + \log n$
2. $T(n) = 3T(n/3) + n/2$
3. $T(n) = 2T(n/2) + n/\log n$
4. $T(n) = 64T(n/8) - n^2 \log n$
5. $T(n) = 2^n T(n/2) + n^n$
6. $T(n) = 2T(n/4) + n^{0.51}$

Solve the recurrence below by manually unrolling it. Remember to provide a proof of your solution.

1. $T(n) = 2T(n-1) + 1$, $T(1) = 1$

Matrix Multiplication

Recall how matrix multiplication works: if we have a matrix A with n rows and k columns ($n \times k$), and a matrix B with k rows and m columns ($k \times m$), their product is defined by multiplying each row of A by each column of B, and their product is of size $n \times m$. This costs roughly $2mnk$ operations.
In the following equations, $\times$ is a placeholder to indicate some element of a matrix (we don't care about the contents, just the shape).

$$\begin{pmatrix} \times & \times & \times \\ \times & \times & \times \end{pmatrix} \begin{pmatrix} \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \end{pmatrix} = \begin{pmatrix} \times & \times \\ \times & \times \end{pmatrix}$$

Matrix multiplication is associative: if we have matrices A,B, and C, $(AB)C = A(BC)$. However, even if the answers are the same, the *cost* is not. Consider the following example: A is 3×8, B is 8×2, and C is 2×5. ABC is 3×5.
If we do $(AB)C$, we need to calculate $A \cdot B$ and then multiply that result with C. The first multiplication takes $3 \cdot 8 \cdot 2 = 48$ operations and creates a 3×2 matrix. Multiplying this with C takes $3 \cdot 2 \cdot 5 = 30$ operations. This is a total of $48 + 30 = 78$ operations.
However, if we do $A(BC)$, we first multiply BC, at a cost of $8 \cdot 2 \cdot 5 = 80$ operations, creating an 8×5 matrix. We then multiply A with this, at a cost of $3 \cdot 8 \cdot 5 = 120$ operations. The total cost of this is 200 operations—more than twice as much work to get to the same answer!
This effect gets even worse when given a long sequence of matrices to multiply together. Sometimes, it is worth it to take some time to figure out what the optimal matrix multiplication order is, before carrying out

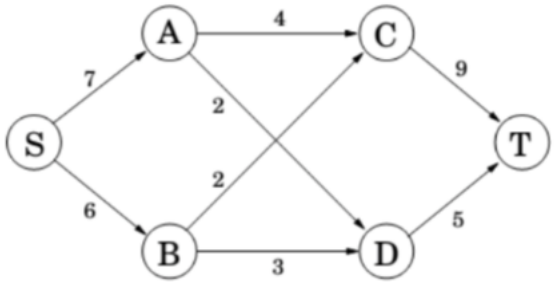
the actual multiplicaton.
You are given a list of tuples. Each element of the list represents one matrix, and the goal is to multiply all of them together at the lowest possible cost. The above example would be given as $[\,(3,8)\,,\,(8,2)\,,\,(2,5)\,]$. Create a dynamic programming algorithm that appropriately orders the matrix multiplication so that the cost of multiplying them all together is as low as possible.

The Edit Distance between two words

You are given two strings, $A[1..n]$ and $B[1..m]$. Given that the cost of inserting, deleting and replacing a character are C_i, C_d, and C_r respectively, write a Dynamic Programming Algorithm to find the minimum cost of transforming A to B.

Network Flow

Consider the following flow network. (the numbers associated with the edges are the edge capacities).



1. Find the maximum flow f and a minimum cut.
2. Draw the final residual graph G_f (along with its edge capacities). In this residual network, mark the vertices reachable from S and the vertices from which T is reachable.
3. An edge of a network is called a *bottleneck* edge if increasing its capacity results in an increase in the maximum flow. List all bottleneck edges in the above network.
4. Give a very simple example (containing at most four vertices) of a network which has no bottleneck edges.
5. Give an efficient algorithm to identify all bottleneck edges in a network. (Hint: Execute a known network flow algorithm (e.g., Ford-Fulkerson), and then examine the residual graph.)

The Escape Problem (Network Flow)

We define the *Escape Problem* as follows. We are given a directed graph $G = (V, E)$ (picture a network of roads). A certain collection of nodes $X \subset V$ are designated as populated nodes, and a certain other collection $S \subset V$ are designated as safe nodes. (Assume that X and S are disjoint.) In case of an emergency, we want evacuation routes from the populated nodes to the safe nodes. A set of evacuation routes is defined as a set of paths in G so that (i) each node in X is the tail of one path, (ii) the last node on each path lies in S, and (iii) the paths do not share any edges. Such a set of paths gives a way for the occupants of the populated nodes to "escape" to S, without overly congesting any edge in G.

1. Given G, X, and S, show how to decide in polynomial time whether such a set of evacuation routes exists.

2. Suppose we have exactly the same problem as in (1), but we want to enforce an even stronger version of the "no congestion" condition (iii). Thus we change (iii) to say "the paths do not share any nodes.". With this new condition, show how to decide in polynomial time whether such a set of evacuation routes exists.

3. Provide an example digraph G, and define X, and S, such that the answer is yes to the question in (1) but no to the question in (2).