COL 351 Sept 8,13,14

TUTORIAL SHEET 5

1. **(Dasgupta, Papadimitriou, Vazirani)** This problem illustrates how to do the Fourier Transform (FT) in modular arithmetic, for example, modulo 7.

   (a) There is a number ω such that all the powers $\omega, \omega^2, \cdots, \omega^6$ are distinct (modulo 7). Find this ω and show that $\omega + \omega^2 + \ldots + \omega^6 = 0$.

   (b) Using the matrix form of the FT, produce the transform of the sequence (0, 1, 1, 1, 5, 2) modulo 7; that is, multiply this vector by the matrix for FT, for the value of ω you found earlier. In the matrix multiplication, all calculations should be performed modulo 7.

   (c) Write down the matrix necessary to perform the inverse FT. Show that multiplying by this matrix returns the original sequence. (Again all arithmetic should be performed modulo 7.)

   (d) Now show how to multiply the polynomials $x^2 + x + 1$ and $x^3 + 2x - 1$ using FT modulo 7.

2. **[KT-Chapter6]** Suppose you own two stores, A and B. On each day you can be either at A or B. If you are currently at store A (or B) then moving to store B the next day (or A) will cost C amount of money. For each day i, $i = 1, \ldots, n$, we are also given the profits $P^A(i)$ and $P^B(i)$ that you will make if you are store A or B on day i respectively. Give a schedule which tells where you should be on each day so that the overall money earned (profit minus the cost of moving between the stores) is maximized.

   Solution: Define two arrays $T_A[]$ and $T_B[]$. $T_A[i]$ gives the most profitable schedule for days $i, \ldots, n$ given that we start at store A on day i. Define $T_B[i]$ similarly. Now, observe that

   $$T_A[i] = P^A(i) + \max(T_A[i+1], T_B[i+1] - C), i < n.$$

   We can write a recurrence for $T_B[i]$ similarly. Further, $T_A[n] = P^A(n), T_B[n] = P^B[n]$.

3. Given a tree T where vertices have weights, an independent set is a subset of vertices such that there is no edge joining any two vertices in this set. Give an efficient algorithm to find an independent set of maximum total weight.

   Solution: Root the tree at any vertex r. For every vertex v, let T_v be the subtree below v. Define two tables: $A(v)$ and $B(v)$. $A(v)$ denotes the maximum independent set of T_v provided v is in the independent set, whereas $B(v)$ denotes the same quantity provided v is not in the independent set. Now, note that for a leaf node v, $A(v)$ is the

weight of v, whereas $B(v)$ is 0. For a non-leaf node v with children $w_1, \ldots, w_k$, observe that

$$A(v) = \text{weight}(v) + \sum_{i=1}^{k} B(w_i), B(v) = \sum_{i=1}^{k} A(w_i).$$

Now show that you can compute these quantities using post-order traversal of the tree.

4. **[KT-Chapter6]** Suppose you are managing the construction of billboards on a heavily-traveled stretch of road that runs west-east for M miles. The possible sites for billboards are given by numbers $x_1, x_2, \ldots, x_n$, each in the interval $[0, M]$ (specifying their position along the highway, measured in miles from its western end). If you place a billboard at location x_i, you receive a revenue of $r_i > 0$. You want to place billboards at a subset of the sites in $\{x_1, \ldots, x_n\}$ so as to maximize your total revenue, subject to the following restrictions – (i) (The environmental constraint) You cannot build two billboards within less than or equal 5 miles of one another on the highway, and (ii) (The boundary constraint) You cannot build a billboard within less than 5 miles of the western or eastern ends of the highway. A subset of sites satisfying these two restrictions will be called valid. Give an algorithm that takes an instance of this problem as input, and returns the maximum total revenue that can be obtained from any valid subset of sites. The running time of the algorithm should be polynomial in n.

   Solution: The second condition is easy to satisfy, just remove all sites which are within less than 5 miles of either end-point. We just have to take care of first constraint – let the sites (after taking care of first constraint) be $\{x_1, \ldots, x_n\}$. For each site i, draw an interval of length 5 starting from x_i. Now notice that this becomes an instance of the interval selection problem.

5. You are given N boxes, where box i has height h_i, width w_i and length l_i. Give an algorithm for finding a stacking of a subset of boxes of maximum total height : box i can be stacked on top of box j if $w_i < w_j$ and $l_i < l_j$.

   Solution: Let the boxes in decreasing order of width be $B_1, \ldots, B_N$. Define table $T[]$ as follows: $T[i]$ gives the maximum total height from boxes $B_i, \ldots, B_n$ *provided* the box B_i is at the bottom. Clearly, $T[n]$ is just h_n. Now, to compute $T[i]$, consider the optimal solution for the instance $B_i, \ldots, B_n$ where B_i is at the bottom. The box just above B_i has to be one of $B_{i+1}, \ldots, B_n$ – let this box be B_k (note that B_k must satisfy the above criteria). Also note that none of the boxes from $B_{i+1}, \ldots, B_{k-1}$ can come above B_k. So,

$$T[i] = h_i + \max_{k=i+1,\ldots,n} T[k],$$

   where the maximum is taken over only those values of k for which $l_k < l_i$.

6. **[KT-Chapter6]** Suppose it is nearing the end of the semester and you are taking n courses, each with a final project that still has to be done. Each project will be graded on the following scale: it will be assigned an integer number on a scale of 1 to $g \geq 1$, higher numbers being better grades. Your goal, of course, is to maximize your average grade on the n projects. Now, you have a total of $H > n$ hours in which to work on the n projects cumulatively, and you want to decide how to divide up this time. For

simplicity, assume H is a positive integer, and you will ll spend an integer number of hours on each project. So as to figure out how best to divide up your time, you have come up with a set of functions $\{f_i : i = 1, 2, \ldots, n\}$ (rough estimates, of course) for each of your n courses; if you spend $h \leq H$ hours on the project for course i, you?ll get a grade of $f_i(h)$. (You may assume that the functions f_i are non-decreasing: if $h < h'$ then $f_i(h) \leq f_i(h')$.) So the problem is: given these functions $\{f_i\}$, decide how many hours to spend on each project (in integer values only) so that your average grade, as computed according to the f_i, is as large as possible. In order to be efficient, the running time of your algorithm should be polynomial in n, g, and H; none of these quantities should appear as an exponent in your running time.

Solution: Define a table $T[h, i]$ which gives the max expected grade for projects $i, \ldots, n$ provided you have only h hours. Now, it is easy to check that

$$T[h, i] = \max_{h'=0,\ldots,h} f_i(h') + T[h - h', i + 1].$$