CSC 373H (2007): Assignment 1

Worth 5%. Due June 7 at 6pm in lecture.

The work you submit must be your own. You may discuss problems with each others; however, you should prepare written solutions alone. Copying assignments is a serious academic offence, and will be dealt with accordingly.

Question 1 [Divide-and-Conquer]

A group of n students $S_1, \ldots, S_n$ work on a common report that consists of n parts $P_1, \ldots, P_n$. They have divided the task so that S_i writes part P_i (for $1 \leq i \leq n$). Now is $(n-1)$ days before the deadline, and they want to cross examine each other's part. Each student S_i will have to review all $(n-1)$ parts P_j for $j \neq i$, and can review only one part a day. The students also want all parts are reviewed every day. ALSO ON EACH DAY, IF A STUDENT S_i REVIEWS PART P_j THEN STUDENT S_j REVIEWS PART P_i. Before start reviewing, their main challenge is to assign themselves each a part to review, for each of the remaining $(n-1)$ days.

Below are examples of possible assignments for the case $n = 2$ and $n = 4$ (day $(n-1)$ is the day before the deadline):

	day 1
S_1	P_2
S_2	P_1

	day 1	day 2	day 3
S_1	P_2	P_3	P_4
S_2	P_1	P_4	P_3
S_3	P_4	P_1	P_2
S_4	P_3	P_2	P_1

Help the students to assign the part to be reviewed by each of them. Use divide-and-conquer technique, you can assume that n is a power of 2. Your output should be in the form of an $n \times (n-1)$ array, where $A[i,j]$ is the part to be reviewed by S_i on day j, for $1 \leq i \leq n$, $1 \leq j \leq n-1$. THE RUNNING TIME OF YOUR ALGORITHM SHOULD BE $\mathcal{O}(n^2)$.

Question 2 [Divide-and-Conquer]

An $n \times n$ grid graph G is the graph whose nodes are order pairs of natural numbers (i,j) where $1 \leq i \leq n$ and $1 \leq j \leq n$; two nodes (i_1, j_1) and (i_2, j_2) are neighbors if and only if $|i_1 - i_2| + |j_1 - j_2| = 1$.

Suppose that each node v of G is labeled by a distinct natural number x_v. We say that v is a local minimum if $x_v < x_u$ for all neighbors u of v. In Example given in Figure 1, there are FOUR local minima: $(2,6), (3,2)$, $(5,1)$ and $(6,6)$.

Given G and its labeling, the problem is to find a local minimum of G.

a) Consider the following algorithm that starts from $(1,1)$ and keeps looking for the smallest neighbor until a local minimum is found:

1. $v \leftarrow (1,1)$
2. While there is a neighbor u of v such that $x_u < x_v$
3.   Let u be the neighbor of v with smallest x_u
4. $v \leftarrow u$
5. End While
6. Return v

6	30	29	41	53	80	10
5	31	36	28	50	71	70
4	32	37	27	57	62	47
3	33	38	26	56	63	45
2	34	39	25	51	24	44
1	35	40	42	52	15	17
	1	2	3	4	5	6

Figure 1: A 6×6 grid graph. The local minima are $(2, 6), (3, 2), (5, 1), (6, 6)$.

Argue that the above algoritm always returns a local minimum of G. What is the running time of this algorithm in the worst case? (State your answer using Θ notation.)

b) Give a divide-and-conquer algorithm that solve this problem in time $\mathcal{O}(n)$. Argue that your algorithm always returns a local minimum. Here you may assume that n is a power of 2.

Question 3 [Dynamic Programming]

Suppose that it's nearing the end of the semester and you're 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 $g > 1$, higher numbers being better grades. Your goal, of course, is to maximize your average grade on the n projects.

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'll spend an integer number of hours on each project. To figure out how best to divide up your time, you've 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 *nondecreasing*: if $h < h'$, then $f_i(h) \leq f_i(h')$.)

So the problem is: Given these functions, 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.