Homework 3

Due Friday, April 2 2010, 12:00pm.

Notes

• You may hand the homework to TAs during office hours. You may also hand the homework to Prof. Ranka in class. No late submissions.

• To force you to write succinctly, we have enforced page limits. These are noted in front of each question.

• Answer each question on a fresh page.

• If the problem necessitates writing an algorithm, you must first informally describe the algorithm, in brief, in a paragraph. You can choose to follow this up with pseudocode that formally describes the algorithm. We will peruse your pseudocode only if your English description is not clear.

• Write your name on the top right hand corner of your homework. Be sure to write your last name as the last word in your name.

• If you are designing an algorithm, you must write a formal proof of correctness.

• Please write legibly.
1. [1 page][15pts] You are given $n$ matrices $M_1, M_2, ..., M_n$ of sizes $s_0 \times s_1, s_1 \times s_2, ..., s_{n-1} \times s_n$ respectively. Suppose you want to compute their product $M_1 \times M_2 \times ... \times M_n$. Matrix multiplication is not commutative (i.e. $M_1 \times M_2 \neq M_2 \times M_1$), but we do know it is associative (i.e. $M_1 \times (M_2 \times M_3) = (M_1 \times M_2) \times M_3$). It turns out that the order in which you compute the product is important because for some such orders, the number of operations (i.e. number of multiplications) required is less than that for other orders. For example, if you have four matrices $M_1, M_2, M_3, M_4$ of sizes $40 \times 25, 25 \times 1, 1 \times 11, 11 \times 101$ respectively, then $(M_1 \times M_2) \times (M_3 \times M_4)$ is the best ordering as it takes only $(40 \times 25 \times 1) + (1 \times 11 \times 101) + (40 \times 1 \times 101) = 6151$ multiplications. The ordering $(M_1 \times ((M_2 \times M_3) \times M_4))$ is the worst. Remember that multiplying a matrix of size $a \times b$ with another of size $b \times c$ takes $O(abc)$ multiplications using the naive method (we aren’t going to consider any faster algorithms like Strassen’s method).

It turns out that for this problem, the greedy approach of choosing the available matrix pair whose product is the cheapest to compute, is suboptimal. Give an example to illustrate this.

Next, prepare an efficient dynamic programming strategy to find the best possible ordering to compute the product of the $n$ given matrices. Write a recurrence relation, explain the algorithm and then write a non-recursive pseudocode. Analyze the complexity of your algorithm. (It turns out that there is an $O(n^3)$ algorithm for this problem).
2. [1 page][15pts] Given a set $S$ of $n$ integers, write a dynamic programming algorithm to check whether there exists a subset of $S$ whose sum equals $W$ where $W$ is an integer. Your algorithm should run in time $O(nW)$. Write the recurrence, explain the algorithm and write a non-recursive pseudocode. Explain whether this can be called a polynomial time algorithm. Extend the algorithm to determine whether there exist two subsets $S_1$ and $S_2$ such that $S_1 \cup S_2 = S$, $S_1 \cap S_2 = \emptyset$ and $\sum_{x \in S_1} x = \sum_{x \in S_2} x$. 
3. [1 page][15pts] A new railway route is being set up from city A to city B. There are totally \( n \) proposed sites along that route, on some of which a railway station is to be built. For each site \( i \) (\( 1 \leq i \leq n \)), you know its distance \( d_i \) from city A. Assume that \( d_1 \leq d_2 \leq \ldots \leq d_{n-1} \leq d_n \). The government is willing to provide fund of \( p_1, p_2, \ldots, p_n \) for the constructing a station at these \( n \) sites respectively (the \( p \) values are not necessarily in increasing order, unlike the \( d \) values). Owing to some regulations laid down by the government, no two railway stations may be located at a distance of less than or equal to 15 miles from one another. Your job is to come up with an efficient dynamic programming algorithm to compute the number of railway stations and their locations that maximize the total amount of money that the government will be willing to pay, subject to the aforementioned constraint. Write the relevant recurrence relation, a non-recursive pseudocode and analyze the time complexity of the algorithm.
4. Pearls have been prized for their beauty and rarity for more than four thousand years. Natural pearls are collected in the coastal regions of Asia, Africa and America. Since pearl collection is dangerous, researchers are asked to build a robot that can do this task. A task for the robot consists of n potential pearl sites all located along a single coastal line. Robot starts at site 1, but can finish at any site. To make things easier we assume that all the sites are on a straight line, one after the other. The time taken to travel from site $i$ to site $i+1$ is $t_{i,i+1}$ hours, $1 \leq i \leq n - 1$. Assume that robots battery lasts for $m$ hours. Also, the expected number of pearls per hour is $p_i$ at site $i$, $1 \leq i \leq n$. It was noticed that each hour of pearl collection decreases the expected number of pearls to be found in the next one hour, by a constant amount. This amount varies from site to site and is given as $d_i$, for site $i$, $1 \leq i \leq n$. Design a dynamic programming algorithm to maximize the total pearls found. Give the pseudo-code and run time complexity.
5. Your favorite sawmill charges by length to cut each board of lumber. For example, to make one cut anywhere on an 8 ft. board of lumber costs $8. The cost of cutting a single board of wood into smaller boards will depend on the order of the cuts. As input, you are given a board of length \( n \) marked with \( k \) locations to cut. Design a dynamic programming algorithm that, given an input length \( n \) of wood and a set of \( k \) desired cut points along the wood, will produce a cutting order with minimal cost in \( O(k^c) \) time, for some constant \( c \).

![Figure 1: Example](image-url)
6. Assume that \( n \) programs are to be stored on two tapes. Let \( l_i \) be the length of tape needed to store the \( i \)th program. Assume that \( \sum l_i \leq L \), where \( L \) is the length of each tape. A program can be stored on either of the two tapes. If \( S_1 \) is the set of programs on tape 1, the the worst-case access time for a program is proportional to \( \max \{ \sum_{i \in S_1} l_i, \sum_{i \not\in S_1} l_i \} \). An optimal assignment of programs to tapes minimizes the worst-case access times. Formulate a dynamic programming approach to determine the worst-case access time of an optimal assignment. Write an algorithm to determine this time. What is the complexity of your algorithm?
7. [1 page][10pts] Let $L$ be an array of $n$ distinct integers. Give an efficient algorithm to find the length of a longest increasing subsequence of entries in $L$. For example, if the entries are 11, 17, 5, 8, 6, 4, 7, 12, 3, a longest increasing subsequence is 5, 6, 7, 12. What is the run time of your algorithm?