Assignment 3, Winter, 2017
COSC 3P03: Design and Analysis of Algorithms

Due: March 22, Wed., 5:00 PM.

1. (10) In the linear-time selection algorithm discussed in class, after medians of all groups $\{x_1, x_2, x_3, x_4, x_5\}$, $\{x_6, x_7, x_8, x_9, x_{10}\}$, ..., are found, we recursively find m, the median of these medians. Is m necessarily the median of the input elements x_1, x_2, ...? Why or why not?

No. There are many, many examples. Here is just one of them:

1	2	11
3	4	12
5	6	13
7	8	14
9	10	15

The medians of the three groups are 5, 6, and 13 and their median is 6 while the median of all 15 numbers is 8.

2. (40) Consider the problem of computing binomial coefficients. A recursive algorithm is as follows:

```
bin(n, k)
   if k=0 or k=n
      return 1
   else
      return bin(n-1, k-1) + bin(n-1, k)
end
```

(a) Use induction on n to show that this algorithm computes $2\binom{n}{k} - 1$ terms to determine $\binom{n}{k}$.
(b) Design and implement (programming) an algorithm using dynamic programming to compute $\binom{n}{k}$. Please

- Give the recurrence
- Describe your algorithm
- Give a listing of your code
- Results of testing your algorithm for various n's and k's
- For this and Q4, use submit3p03 to submit a softcopy of your programming portion

What are its time and space complexities?

For $n = 1$ and any k, there is only one term involved, namely 1, which is $2\binom{1}{k} - 1 = 2 - 1$.

Assume that the claim is true for $n-1$, namely, for any x, to compute $\binom{n-1}{x}$, we need to compute $2\binom{n-1}{x} - 1$ terms, then for n, to compute $\binom{n}{k}$, according to the algorithm above, we need to compute $\binom{n-1}{k-1}$ and $\binom{n-1}{k}$ plus one more term. So with the induction hypothesis, the total numbner of terms involved is

$$
\begin{aligned}
(2\binom{n-1}{k-1}-1)+(2\binom{n-1}{k}-1)+1 &= 2(\binom{n-1}{k-1}+\binom{n-1}{k})-1 \\
&= \binom{n}{k}-1.
\end{aligned}
$$

The recurrence (obviously) is as follows:

$$
\binom{n}{k}=\binom{n-1}{k-1}+\binom{n-1}{k},
$$

with the initial condition $\binom{n}{0}=\binom{n}{n}=1$ for any $n \geq 0$.
Algorithm (it actually computes the first $n+1$ row of the Pascal triangle for any n and k) :
Build a table P[0..m][0..m] to store the Pascal triangle

```
P[0][0] = 1;
P[1][0] = 1;
P[1][1] = 1;


for (i=2; i<=n; i++) {
        for (j=0; j<=i; j++) {
                P[i][j] = P[i-1][j-1] + P[i-1][j];
        }
}
```

The table size is $O(n^2)$ and each entry takes constant time. So the space and time complexities are both $O(n^2)$.

3. (25) Two character strings may have many common substrings. Substrings are required to be contiguous in the original string. For example, *photograph* and *tomography* have several common substrings of length one (i.e., single letters), and common substrings *ph*, *to*, and *ograph* as well as all the substrings of *ograph*. The maximum common substring length is 6. Let $X = x_1x_2\cdots x_m$ and $Y = y_1y_2\cdots y_n$ be two character strings. Design an algorithm to find the maximum common substring length for X and Y using dynamic programming. Analyze the worst-case running time and space requirements of your algorithm as functions of m and n.

Let C_{ij} be the length of a longest common substring ending at x_i and y_j.

$$
\begin{aligned}
C_{i0} &= C_{0,j} = 0 \\
C_{ij} &= \begin{cases} C_{i-1,j-1}+1 & \text{if } x_i = y_j \\ 0 & \text{else} \end{cases}
\end{aligned}
$$

The final result, namely, the length of a longest common substring, is $\max_{1\leq i\leq m, 1\leq j\leq n}\{C_{ij}\}$. Both the space and time required are $O(mn)$.

4. (25) (Programming) Implement the algorithm discussed in class to find an optimal way to multiply n matrices. Please note that you need to follow the following steps:

- ask the user for an input, n and $r_0, r_1, \cdots, r_n$ that represent the dimensions of the n matrices.
- implement the dynamic programming algorithm covered in class.

- then print out the optimal way of multiplying these n matrices. For example, if $n = 4$ and the optimal way is to multiply M_2 and M_3 first, followed by multiplying the product just obtained with M_4, followed by multiplying M_1 with the last product, you should print out $(M_1 \times ((M_2 \times M_3) \times M_4))$.

Please follow the instructions for Q2 as to what to submit.

We need to modify the algorithm presented in class by adding an $n \times n$ matrix $K = (k_{ij})$ where k_{ij} is used to record the k that gives us m_{ij} as in the recurrence

$$m_{ij} = \min_{i \leq k < j} \{m_{ik} + m_{k+1,j} + r_{i-1} \times r_k \times r_j\}.$$

Once we have the matrix K, the routing for printing out the best order for $M_i \times M_{i+1} \times \cdots \times M_j$ is as follows:

```
order(i, j)
     if (i=j)
          print Mi
     else
          print (
          order(i, K[i][j])
          print X
          order(K[i][j]+1, j)
          print )
```