Data Structures
Homework #2

Sketched Solution

1. Please refer the code shown as below. Give the number of operations (steps) for each statement. Then, sum up these numbers to derive the total number of operations and present this total number using the asymptotic notations.

   Prod($A[p][q], B[q][r]$)
   for $i = 1$ **to** p
       for $j = 1$ **to** r
           $C[i][j] = 0$
           for $k = 1$ **to** q
               $C[i][j] = C[i][j] + A[i][k] \times B[k][j]$;

   Solution:

Prod($A[p][q], B[q][r]$)	time	asym.
for $i = 1$ **to** p	p	O(p)
for $j = 1$ **to** r	pr	O(pr)
$C[i][j] = 0$	pr	O(pr)
for $k = 1$ **to** q	pqr	O(pqr)
$C[i][j] = C[i][j] + A[i][k] \times B[k][j]$;	pqr	O(pqr)
overall		O(pqr)

2. An array A contains $n-1$ unique integers in the range $[0, n-1]$. So, there is one number in this range that is not in A. Design an $O(n)$-time algorithm for finding that number. You are only allowed to use $O(1)$ additional space besides the array A itself. Please write your idea first. Then, present your algorithm in pseudo-code and analyze it in time and space.

   Solution: First calculate the sum $\sum_{i=1}^{n-1} = \frac{n(n-1)}{2}$. Then calculate the sum of all values in the array A. The missing element is the difference between these two numbers. All the space you need is two additional temporary integer variables. To calculate the sum one needs $O(n)$ time.

   MissingNumber(A)
   $S = \frac{n(n-1)}{2}$;
   $SUM = 0$;
   for $i = 0$ **to** $(n-2)$
       $SUM = SUM + A[i]$;
   return $(S - SUM)$

3. Find a Big-O notation in terms of n for the number of times the statement $x = x + 1$ is executed in the following segment

```
j = n
while (j ≥ 1)
    for i = 1 to j
        x = x + 1;
    j = j/2;
```

Solution:
Let $t(n)$ denote the number of times the statement $x = x + 1$ is executed. Then,

$$t(n) \leq \frac{n}{1 - \frac{1}{2}} = 2n.$$

So, $t(n) = \mathrm{O}(n)$.

4. Please show the followings:

   (a) let x and y be real numbers with $0 < x < y$. Prove that n^x is $\mathrm{O}(n^y)$, but n^y is not $\mathrm{O}(n^x)$.

   (b) $\log_a n$ is $\mathrm{O}(\log_b n)$ for any real numbers $a > 1$ and $b > 1$.

   Solution:

   (a) n^x **is** $\mathbf{O}(n^y)$:
   Since $y - x > 0$, we know that $n^{y-x} > 1$ for all $n > 1$ and hence $n^x < n^y$. Therefore, n^x is $\mathrm{O}(n^y)$.
   n^y **is not** $\mathbf{O}(n^x)$:
   Suppose that n^y is $\mathrm{O}(n^x)$. This implies that $n^y \leq cn^x$ for some constant c and for sufficiently large n. It follows that

   $$n^{y-x} \leq c, \text{ so} (n^{y-x})^{\frac{1}{y-x}} \leq c^{\frac{1}{y-x}}, \text{ which means } n \leq c^{\frac{1}{y-x}}.$$

   The last statement above if obviously false for large enough n and therefore n^y is not $\mathrm{O}(n^x)$.

   (b) By change of base,

   $$\log_a n = (\log_a b)(\log_b n)$$
   $$\log_a n = c(\log_b n),$$

   where $c = \log_a b$. Therefore, by definition, $\log_a n$ is $\mathrm{O}(\log_b n)$.

5. Show that $O(max\{f(n), g(n)\}) = O(f(n) + g(n))$.

   Solution:

   $$\text{let } h(n) = max\{f(n), g(n)\}, \text{ then we get}$$

   $$\begin{array}{rcl} 2h(n) & \geq & f(n) + g(n) \\ h(n) & \geq & \frac{1}{2}(f(n) + g(n)) \end{array}$$

   $$\text{Thus, } O(max\{f(n), g(n)\}) = O(f(n) + g(n)).$$

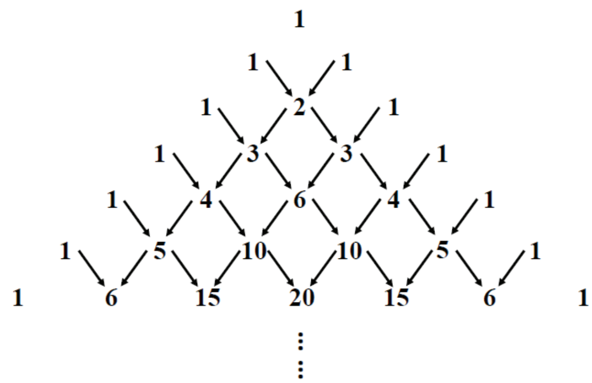

Figure 1: The top of Pascal's triangle of binomial coefficients

6. (**Programming**)

The binomial coefficients may be defined by the following recurrence relation, which is the idea of ***Pascal's triangle***. The top of Pascal's triangle is shown in Figure 1.

$$\begin{array}{ll} C(n,0)=1 \quad \text{and} \quad C(n,n)=1 & \text{for } n \geq 0 \\ C(n,k)=C(n-1,k)+C(n-1,k-1) & \text{for } n > k > 0 \end{array}$$

(a) Draw the recursion tree for calculating $C(6,4)$.

(b) Write a function `PA_r` using *recursive* function to generate Pascal's triangle in the lower left half of the array.

(c) As (b), write a *nonrecursive* function `PA_n` to generate Pascal's triangle in the lower left half of the array.

(d) Again, write a nonrecursive program `PA_d` that uses neither an array to calculate each $C(n,k)$ directly.

(e) Please do some experiments by yourself to observe the running time and perform a comparison on these three versions by the execution time. Thus, you need to write a Python program to do the comparison, which will call `PA_r`, `PA_n`, and `PA_d`, respectively. Please see the execution example below. To measure the execution time. Please use Python's `timeit` module to do the measuring.

(f) Determine the approximate space and time requirements for each of the algorithms devised in parts (b), (c), and (d).

Execution Example

```
input the degree n:  6
Recursion execution time:  6.79966206962469e-05
Non-Recursion execution time:  5.8053718613296224e-05
Direct Computation execution time:  9.942902082935023e-06
input the degree n:
```

Solution:

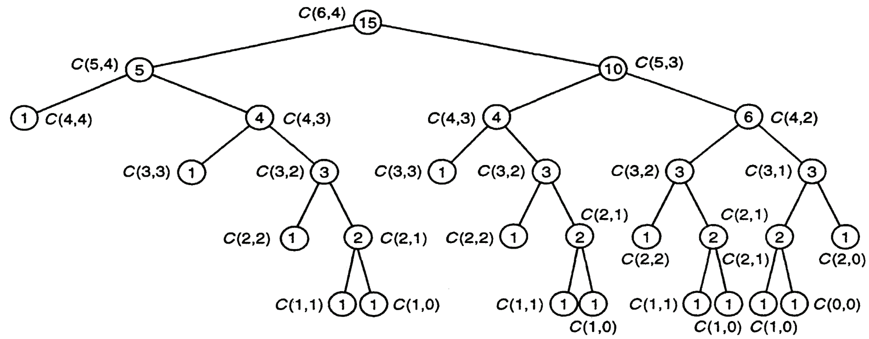

Figure 2: The trace on the recursion tree for $C(6,4)$

(a) The recursion tree for calculating $C(6,4)$ is in Figure 2.

(b) Program Omitted

(c) Program Omitted

(d) The term $C(n,k)$ can be calculated as

$$C(n,k) = \frac{n(n-1)\cdots(n-k+1)}{k!} = \prod_{i=1}^{k} \frac{n-i+1}{i}.$$

So, the program can use this definition to derive the Pascal Triangle.

About submitting this homework

1. For problem 1, 2, 3, 4, and 5, Please
   (1) write all of your solutions on the papers of size A4,
   (2) leave you name and student ID on the first page, and
   (3) hand in your solutions on papers to me in class
2. For problem 6, things to be submitted include:
   (1) README: a plain text file (.txt, not WORD .doc file) describes your student id, your name, what is in your homework, how to execute your Python scripts, what you have observed, and you conclusion.
   (2) the source code;
   (3) anything that will help us to understand your programs.

   Note: Please upload your code to `http://www.ischool.ntut.edu.tw/`.
3. There will be some **penalty** on the things you miss to submit or different zipped file format. **Late work** is not acceptable. Remember, the **deadline** is the **midnight of Oct 16**, 2018.
4. Honest Policy: We encourage students to discuss their work with the peer. However, each student should write the program or the problem solutions on her/his own. Those who copy others work will get 0 on the homework grade.