Solutions for Assignment 1

CSCI2100B

February 20, 2013

1 Written Assignment

Exercise 1.1

(5) $\sum_{i=1}^{n} a^i$
Solution: According to the summation formula of geometric progression, the answer is $\frac{a^{n+1}-a}{a-1}$.

(8) $\sum_{i=0}^{n} i^2$
Solution:

$$\begin{array}{rc} & i^3-(i-1)^3 = 3i^2-3i+1 \\ \Rightarrow & n^3 = \sum_{i=1}^{n}(i^3-(i-1)^3) = \sum_{i=1}^{n}(3i^2-3i+1) \\ = & 3\sum_{i=1}^{n} i^2 - 3\frac{n(n+1)}{2} + n \\ \Rightarrow & \sum_{i=0}^{n} i^2 = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \end{array}$$

(13) Is $2^{2n} = O(2^n)$?
Solution: No. $2^{2n} = O(4^n)$

Exercise 1.2

(1) Is the GCD function distributive? Associative? Commutative?
Solution: Commutativity: Yes. The definition does not specify the order of a and b.
Associativity: Yes. Suppose $GCD((a, b), c) = m$ and $GCD(a, (b, c)) = n$.

$$\begin{array}{rcl} & GCD((a,b),c) = m & \Rightarrow m|GCD(a,b), m|c \\ \Rightarrow & m|a, m|b, m|c & \Rightarrow m|a, m|GCD(b,c) \\ \Rightarrow & m|n & (n \text{ is the largest } k \text{ satisfying } k|a \text{ and } k|GCD(b,c)) \end{array}$$

We can prove $n|m$ in the same way. Thus, $m = n$.
Distributivity: No. $GCD(3, 2) = 1, GCD(3, 4) = 1$, but $GCD(3, (2 + 4)) = 3$.

(3) If $GCD(a, b) = p$ and $GCD(c, d) = q$, is $GCD(ac, bd) = pq$ true for all the a, b, c, d? Either prove it or give a counterexample.
Solution: Counterexample: a = 5, b = 4, c =4, d = 5.

Exercise 1.3

(5) Solve $x_n = x_{n-1} - \frac{1}{4}x_{n-2}$, with $x_0 = 1, x_1 = 1/2$.
Solution: First solve the quadratic formula $t^2 - t + \frac{1}{4} = 0$.

The solutions are $t_1 = t_2 = \frac{1}{2}$. Thus the solution is of the form $x_n = a(\frac{1}{2})^n$.
To satisfy the initial conditions, we can obtain $a = 1$. Thus, $x_n = (\frac{1}{2})^n$.

(8) Solve $T(1) = 1$, and for all $n \geq 2$, $T(n) = 3T(n-1) + 2$.
Solution:

$$\begin{aligned} & T(n) = 3T(n-1) + 2 \\ \Rightarrow \quad & T(n) + 1 = 3T(n-1) + 3 = 3(T(n-1) + 1) \end{aligned}$$

Let $S(n) = T(n) + 1$. Then

$$\begin{aligned} & S(n) = 3S(n-1), S(1) = 2 \\ \Rightarrow \quad & S(n) = 2 \times 3^{n-1} \\ \Rightarrow \quad & T(n) = 2 \times 3^{n-1} - 1 \end{aligned}$$

Exercise 1.4

(3) Prove $\sum_{i=1}^{n}(2i-1) = n^2$.
Solution:

$$\begin{aligned} & i^2 - (i-1)^2 = (i - i + 1)(i + i - 1) = 2i - 1 \\ \Rightarrow & \sum_{i=1}^{n}(2i-1) = \sum_{i=1}^{n}(i^2 - (i-1)^2) = n^2 - 0^2 = n^2 \end{aligned}$$

(5) Prove $2lg(n!) > nlgn$ by using Induction, where n is a positive integer greater than 2.
Solution: Let $P(n)$ be $lg(n!) > nlgn$, where n is a positive integer.
For $n = 1$, $L.H.S = 2lg(1!) > lg(1) = R.H.S.$ P(1) is true.
Assume P(k) is true, i.e. $2lg(k!) > klgk$, where k is a positive integer
For $n = k + 1$,

$$\begin{aligned} \text{L.H.S} &= 2lg((k+1)!) \\ &= 2(lg(k!) + lg(k+1)) \\ &> klgk + 2lg(k+1) && \text{(by assumption)} \\ &> (k-1)lg(k+1) + 2lg(k+1) && (k+1 > e > (1+\tfrac{1}{k})^k \Rightarrow k^k > (k+1)^{k-1} \text{ for } k \geq 2) \\ &= (k+1)lg(k+1) \\ &= \text{R.H.S} \end{aligned}$$

$P(k+1)$ is also true.
Therefore, by M.I., $P(n)$ is true for all positive integer n.

Exercise 1.6

```
(2) for i = 1 to n;
        for j = i to n;
            x := x + 1;
```

Solution: $f(n) = n + (n-1) + ... + 1 = \frac{n(n+1)}{2}$, $g(n) = n^2$.

```
(4) for i = 1 to n;
        for j = i to n;
            for k = 1 to i*n;
                x := x + 1;
```

Solution: $f(n) = \sum_{i=1}^{n}(n-i+1)in = n^2\sum_{i=1}^{n} i + n\sum_{i=1}^{n}(i-i^2) = \frac{n^2(n+1)(n+2)}{6},\ g(n) = n^4$.

```
(6) for i = 1 to n-1;
        for j = i+1 to n;
            for k = 1 to j;
                x := x + 1;
```

Solution: $f(n) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} j = \sum_{i=1}^{n-1}\frac{(n+i+1)(n-i)}{2} = \frac{1}{2}\sum_{i=1}^{n-1}(n^2-i^2+n-i)$
$= \frac{1}{2}(n(n+1)(n-1) - \frac{(n-1)n(2n-1)}{6} - \frac{n(n-1)}{2}) = \frac{(n-1)n(n+1)}{3},\ g(n) = n^3$.

Exercise 1.9

(3) If we define the total cost, $C(n)$, of the algorithm as:

$$C(n) = t(n) + 5 \times s(n).$$

Now calculate the average cost for each of the two algorithms. Which one is the better algorithm? By how much?
Solution: You can use the following program to calculate the answer.

```
#include<stdio.h>
#define MAXN 100
int main()
{
        int i;
        double A=0, B=0;
        for (i=1; i<=MAXN; i++)
        {
                if (i<10) A+=i*i;
                else if (i<50) A+=i;
                else A+=i*i*i;

                if (i<20) A+=5*i;
                else A+=5*1.5*i;

                if (i<30) B+=i;
                else if (i<70) B+=i*i;
                else B+=i*i*i;

                if (i<50) B+=5*5*i;
                else B+=5*0.5*i;
        }
        A = A/MAXN;
        B = B/MAXN;
        printf("The average cost of A is %f more than that of B.\n", A-B);
        return 0;
}
```

B is better than A for 42265.025.

(4) Come up with a strategy that you would use to minimize the time and space complexity individually?
Solution: $t(n) = min\{t_A(n),\ t_B(n)\}$, $s(n) = min\{s_A(n),\ s_B(n)\}$. Thus,

$$t(n) = \begin{cases} n & \text{if } 1 \le n < 50 \\ n^2 & \text{if } 50 \le n < 70 \\ n^3 & \text{if } 70 \le n \le 100 \end{cases}$$

$$s(n) = \begin{cases} n & \text{if } 1 \le n < 20 \\ 1.5n & \text{if } 20 \le n < 50 \\ 0.5n & \text{if } 50 \le n \le 100 \end{cases}$$

2 Programming Assignment

Exercise 1.15

Analysis: The most intuitive implementation is to convert both the decimal numbers into binary ones. Then compare them bit by bit. A easier way without converting is to directly divide 2 until both of them are 0. In each step, if the results of mod 2 are different, add the answer by 1. One place you should pay attention to is that if you stop the while loop when any one of them is 0, you will be wrong on the case like 65535 and 0. The sample code is shown below.

```
#include <stdio.h>

int main()
{
        int T, a, b, ans, i;

        scanf("%d", &T);
        for (i = 0; i<T; i++) {
                scanf("%d%d", &a, &b);
                ans = 0;
                while (a || b) { /*stop when a and b are both 0*/
                        if (a%2!=b%2) ans++; /*start to compare from the low order position*/
                        a = a/2; /*then drop the position which has been compared*/
                        b = b/2;
                }
                printf("%d\n", ans);
        }
        return 0;
}
```

Exercise 1.19

Analysis: About input, if you manipulate char instead of string, please do not remember to read the new line character at the end of the first line. This problem mainly aims to examine your skill on dealing with strings. In C language, you can directly convert char to int by using a converter (int) before the variable. Some tricky points you may pay attention to: i). 0 and 1 are not prime; ii). n_{min} can never be 0; iii). do not update n_{min} until you have read the whole string. The sample code is shown below.

```
#include <stdio.h>
#include <math.h>

int isprime(int num)
{
        if (num == 0 || num == 1) return 0; /*special cases*/
```

```
	int start = (int)sqrt(num);
	int i, ans = 1;
	for (i=2; i<=start; i++) {
		if (!(num%i)) {
			ans = 0;
			break;
		}
	}
	return ans;
}

int main()
{
	int T, min, max, i, j;
	char ch;
	int times[26]; /*save the times one character appears in the string*/

	scanf("%d", &T);
	getchar(); /*read the new line character*/
	for (i = 0; i<T; i++) {
		min = 256;
		max = 0;
		for (j=0; j<26;j++)  times[j] = 0;
		while ((ch=getchar())!='\n')  times[(int)(ch)-(int)('a')]++;
		for (j=0; j<26;j++) {
			if (times[j]) {
				if (times[j]>max) max=times[j];
				if (times[j]<min) min=times[j];
			}
		}
		if (isprime(max-min)) printf("YES\n");
		else printf("NO\n");
	}
	return 0;
}
```