EE2410 Data Structure Hw #1 (Chapter 1~2 of textbook)

due date 3/29/2024 23:59

Student No.: ____111060005_____

Name: ______胡昱煊________

Note: Use MS Word to **edit this file** by directly typing your student number and name in above blanks and your answer to each homework problem right in the **Sol:** blanks as shown below. Then save your file as **Hw1-SNo.pdf**, where SNo is your student number. Submit the **Hw1-SNo.pdf** file via eLearn. The grading will be based on the correctness of your answers to the problems, and the **format requirement**. Fail to comply with the aforementioned format (file name, header, problem, answer, problem, answer,…), will certainly degrade your score. If you have any questions, please feel free to ask. Submit your homework before the deadline (midnight of 3/29). Fail to comply (**late** homework) will have ZERO score. **Copy** homework will have SERIOUS consequences.

1. (10%) Determine the asymptotic complexity ($\Theta()$) by step table of the each of the following two program segments respectively:

   Code (a):

```
for(i=1; i<=n; i++)
    for(j=1; j<=i; j++)
        for(k=1; k<=j; k++)
            x++;
```

   Code (b)

```
i=1;
while(i<=n)
{
      x++;
      i++;
}
```

Sol:

(a)

	s/e	Frequency	Subtotal
for(i=1; i<=n; i++)	1	$n+1$	$n+1$
for(j=1; j<=i; j++)	1	$\sum_{j=1}^{n}(j+1)$	$\frac{n^2+3n}{2}$

for(k=1; k<=j; k++)	1	$\sum_{i=1}^{n}\sum_{k=1}^{i}(k+1)$	$\frac{n^3+6n^2+5n}{6}$
x++;	1	$\sum_{i=1}^{n}\sum_{k=1}^{i}k$	$\frac{n^3+3n^2+2n}{6}$
Total	$\frac{n^3+6n^2+11n+3}{3}$		

(b)

	s/e	Frequency	Subtotal
i=1	1	1	1
while(i<=n)	1	$n+1$	$n+1$
x++;	1	n	n
i++;	1	n	n
Total	$3n+2$		

2. (10%) For the function Multiply() shown below,
   (a) Obtain the step count for the function using the step table method.
   (b) Repeat (a) but use Θ() notation instead of exact step counts and frequency counts..

```
void Multiply(int **a, int **b, int **c, int m, int n, int p)
{
    for(int i=0;i<m;i++)
        for(int j=0; j<p; j++)
        {
            c[i][j] = 0;
            for(int k=0;k<n;k++)
              c[i][j] += a[i][k] * b[k][j];
        }
}
```

Sol:

(a)

	s/e	Frequency	Subtotal
{	0	1	0
for(int i=0;i<m;i++)	1	$m+1$	$m+1$

for(int j=0; j<p; j++)	1	$m(p+1)$	$m(p+1)$
{	0	1	0
c[i][j] = 0;	1	mp	mp
for(int k=0;k<n;k++)	1	$mp(n+1)$	$mp(n+1)$
c[i][j] += a[i][k] * b[k][j];	1	mpn	mpn
}	0	1	0
}	0	1	0
Total	$2mpn + 3mp + 2m + 1$		

(b)

	s/e	Frequency	Subtotal
{	0	1	0
for(int i=0;i<m;i++)	1	$\Theta(m)$	$\Theta(m)$
for(int j=0; j<p; j++)	1	$\Theta(mp)$	$\Theta(mp)$
{	0	1	0
c[i][j] = 0;	1	$\Theta(mp)$	$\Theta(mp)$
for(int k=0;k<n;k++)	1	$\Theta(mpn)$	$\Theta(mpn)$
c[i][j] += a[i][k] * b[k][j];	1	$\Theta(mpn)$	$\Theta(mpn)$
}	0	1	0
}	0	1	0
Total	$\Theta(mpn)$		

3. (10%) For each of the complexity expression below, determine its overall complexity. For example, given expression $2n + 3n$, the overall complexity should be $O(n)$, not $O(n^2)$, i.e., use the smallest possible class of functions.
   (a) $2n^2 - 3n = O(?)$
   (b) $n! + 2^n = O(?)$
   (c) $5n^2 + n \log n = O(?)$
   (d) $n^{1.001} + n \log n = O(?)$
   (e) $5n^3 - 3n^2 \log n + 2n = O(?)$

Sol:

(a) $2n^2 - 3n = O(n^2)$

(b) $n! + 2^n = O(n!)$

(c) $5n^2 + n \log n = O(n^2)$

(d) $n^{1.001} + n \log n = O(n \log n)$

(e) $5n^3 - 3n^2 \log(n) + 2n = O(n^3)$

4. (10%) **(Recursion and Performance Analysis)** According to the below recursive method, answer the following questions.

```
int recurse(int n)
{
    if (n <= 0)
    return 1;
    else
    return 1 + recurse(n/2);
}
```

a) What does the method return for n=8?

b) How many times will your method be called recursively for n=256?

c) What is the algorithm complexity of the recursive method in big Oh notation?

d) What is the total variable space needed for the execution of the method for an input size of 256, if the integer size is 4 bytes?

Sol:

(a) $rec(8) = 1 + rec(4) = 1 + (1 + rec(2)) = 1 + 1 + (1 + rec(1))$
$= 1 + 1 + 1 + (1 + rec(0)) = 1 + 1 + 1 + 1 + 1 = 5.$

(b) By (a), when $n = 8 = 2^3$, the method is called $3 + 2 = 5$ times. Therefore, when $n = 256 = 2^8$, the method will be called $8 + 2 = 10$ times.

(c)

Condition: n > 0	s/e	Frequency	Subtotal
if (n <= 0)	1	1	1
return 1;	1	0	0
else	0	0	0
return 1 + recurse(n/2);	$1 + T(\frac{n}{2})$	1	$1 + T(\frac{n}{2})$
Total	$2 + T(\frac{n}{2})$, assume $n = 2^k$		

$$T(n) = 2 + T\left(\frac{n}{2}\right) = 2 + 2 + T\left(\frac{n}{4}\right) = \cdots = 2k + T(1)$$
$$= 2\log_2 n + T(1) = O(\log n).$$

(d) When input $n = 256$, the method will be called 10 times. Therefore, $4 * (\log_2 256 + 2) = 40$ bytes of space are needed to store all variables.

5. (10%) Given an algorithm that solves a problem in three phases. The first phase takes $O(100n)$ to input the data of size n. The second phase takes $O(n \log n)$ to process the data. The third phase takes $O(\log n)$ to output the data.

   (a) What is the complexity of the algorithm?

   (b) If the data size is 10, which phase is most likely to take the longest time to execute?

Sol:

(a) Since $O(n \log n)$ has the highest level of Big Oh Hierarchy among the three, the complexity of the algorithm is $O(n \log n)$.

(b) The first phase: $100 * 10 = 1000$

The second phase: $10 * \log(10) = 10$

The third phase: $\log(10) = 1$

Since $1000 > 10 > 1$, the first phase is most likely to take the longest time to execute.

6. (10%)

   (a) Given a list of objects stored in a sorted array, describe an algorithm to search as efficiently as possible for an object based on a key. (Note that the key type matches the field type by which the array is sorted.) You may describe your algorithm in English, pseudocode, and/or C++ code, as long as your description is clear.

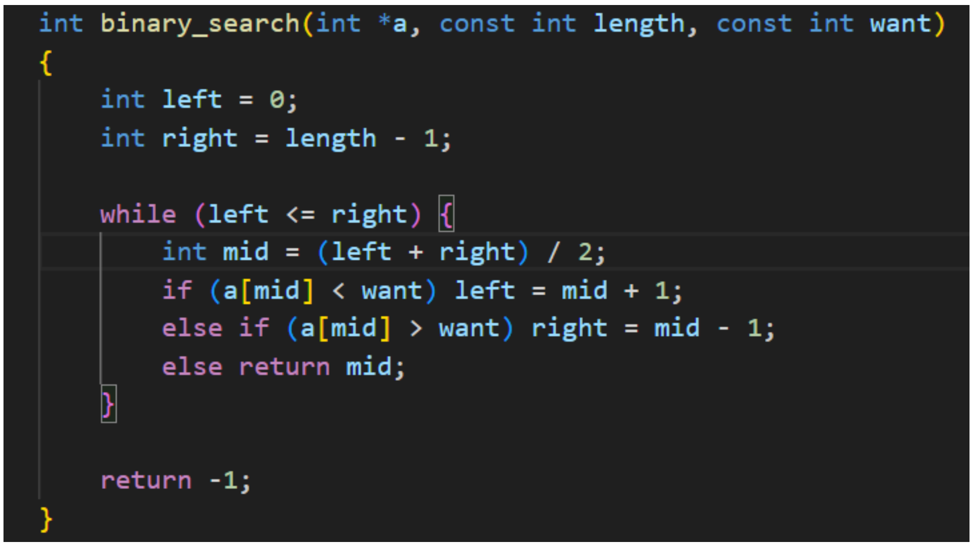
   (b) What will be the Big O running time of the algorithm you described in (a)? Why?

   (c) Given a list of objects stored in an unsorted array, describe an algorithm to search as efficiently as possible for an object based on a key. You may describe your algorithm in English, pseudocode, and/or C++ code, as long as your description is clear.

   (d) What will be the Big O running time of the algorithm you described in (c)? Why?

Sol:

(a) binary search



```c
int binary_search(int *a, const int length, const int want)
{
    int left = 0;
    int right = length - 1;

    while (left <= right) {
        int mid = (left + right) / 2;
        if (a[mid] < want) left = mid + 1;
        else if (a[mid] > want) right = mid - 1;
        else return mid;
    }

    return -1;
}
```

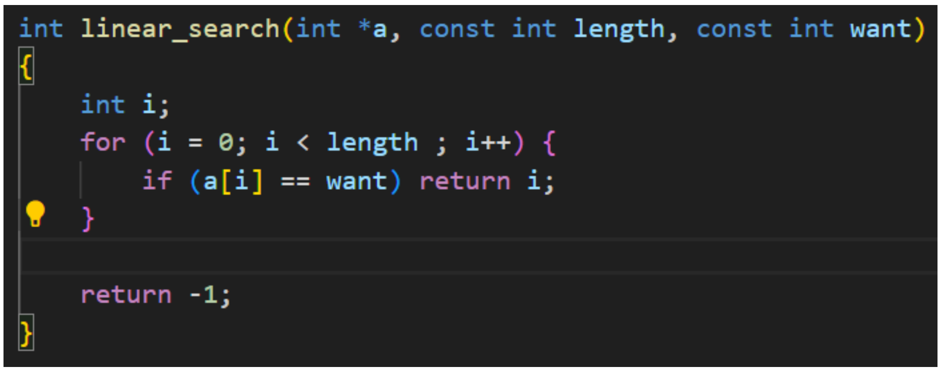
First, compare the middle element "a[mid]" with "want". If "want" is larger than "a[mid]", the program will just focus on the right-half-side of the array, which is a[mid + 1] to a[length – 1], and do the searching since the array is sorted so that elements inside are arranged from small (left) to large (right), and vice versa. Eventually, the program will either find the target "want", which will return the position of the target "mid", or find nothing, which will return -1.

(b)

$O(\log n)$ in the average case and worst case since the problem size is divided by 2 in each loop.

$O(1)$ in the best case.

(c) linear search



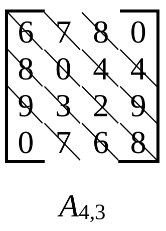
```c
int linear_search(int *a, const int length, const int want)
{
    int i;
    for (i = 0; i < length ; i++) {
        if (a[i] == want) return i;
    }

    return -1;
}
```

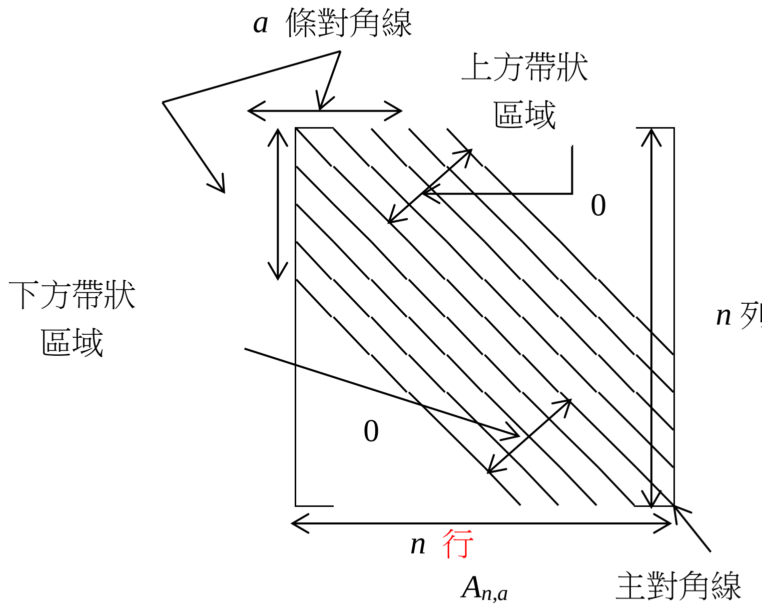
The linear search algorithm will go through the entire array and try to find the target "want". If "want" is found, the program will return the position of "want" in the array, otherwise it will return -1.

(d)

$O(n)$ in the average case and worst case since the program needs to go through every element in the array.
$O(1)$ in the best case.

7. (10%) A square band matrix $A_{n,a}$ is an n x n matrix A in which all the nonzero terms lie in a band centered around the main diagonal. The band includes a-1 diagonals below and above the main diagonal as shown below.





(a) How many elements are there in the band of $A_{n,a}$?
(b) What is the relationship between i and j for element A_{ij} in the band of $A_{n,a}$?
(c) Assume that the band of $A_{n,a}$ is stored sequentially in an array b by diagonals starting with the lowermost diagonal. Thus $A_{4,3}$ above would have the following representation:

b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	b[9]	b[10]	b[11]	b[12]	b[13]
9	7	8	3	6	6	0	2	8	7	4	9	8	4
A_{20}	A_{31}	A_{10}	A_{21}	A_{32}	A_{00}	A_{11}	A_{22}	A_{33}	A_{01}	A_{12}	A_{23}	A_{02}	A_{13}

Obtain an addressing formula for the location of an element A_{ij} in the lower band of $A_{n,a}$, e.g., LOC(A_{20}) = 0, LOC(A_{31}) = 1 in the example above.

Sol:

(a) $n^2 - 2\sum_{j=1}^{n-a} j = n^2 - 2 * \frac{(n-a)(n-a+1)}{2} = -a^2 + a + 2na - n$

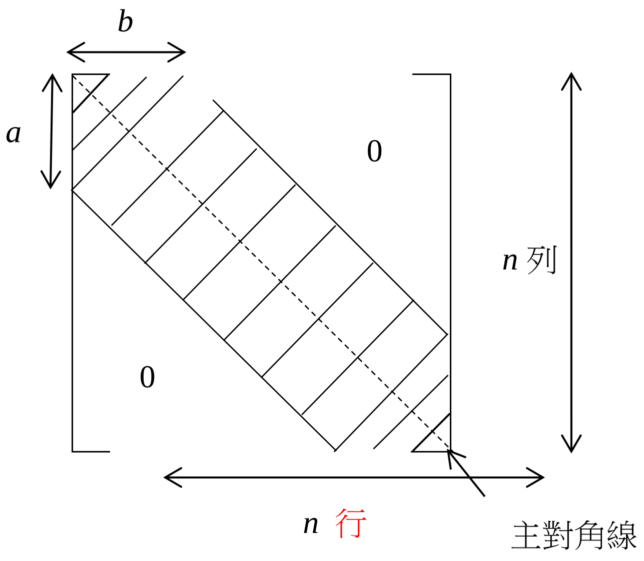
(b) The i, j of elements lying in the same diagonal have the same difference $i-j$, and the absolute value of this difference is the distance between the main diagonal and the diagonal the element lies in.

$i-j$ relation	location of the element
$i-j>0$	below the main diagonal
$i-j=0$	on the main diagonal
$i-j<0$	above the main diagonal

(c) $(i-j)$ means the distance between the line the element is located to. Since there are $a-1-(i-j)$ lines below the element, elements in those lines are $\sum_{k=i-j+1}^{a-1} n-k$. Then calculate the element's position on its line, which is $(j+1)^{th}$. That way, we can get the element's position, which is:

$$A_{ij} = \left(\sum_{k=i-j+1}^{a-1} n-k\right) + j$$

8. (10%) A generalized band matrix $A_{n,a,b}$ is an n x n matrix A in which all the nonzero terms lie in a band made up of a-1 diagonals below the main diagonal, the main diagonal, and b-1 above the main diagonal as shown below.



$A_{n,a,b}$

(a) How many elements are there in the band of $A_{n,a,b}$?

(b) What is the relationship between i and j for element A_{ij} in the band of $A_{n,a,b}$?

(c) Obtain a sequential representation of the band of $A_{n,a,b}$ in one-dimensional array c.

Sol:

(a) $n^2 - \sum_{j=1}^{n-a} j - \sum_{j=1}^{n-b} j$

$$= n^2 - \frac{(n-a)(n-a+1)}{2} - \frac{(n-b)(n-b+1)}{2}$$

$$= n(a+b-1) - \frac{a(a-1)+b(b-1)}{2}$$

(b) The i, j of elements lying in the same diagonal have the same difference $i-j$, and the absolute value of this difference is the distance between the main diagonal and the diagonal the element lies in.

$i-j$ relation	location of the element
$i-j>0$	below the main diagonal
$i-j=0$	on the main diagonal
$i-j<0$	above the main diagonal

(c) In the lower band and diagonals:

$$A_{ij} = \left(\sum_{k=i-j+1}^{a-1} n-k\right) + j$$

In the upper band:

$$A_{ij} = \left(\sum_{k=0}^{j-i+1} n-k\right) + i + \left(\sum_{k=1}^{a-1} n-k\right)$$

9. (10%) String matching by KMP algorithm
   (a) Please describe the Knuth, Morris, and Pratt's pattern matching algorithm.
   (b) Compute the failure function for the following string:
   b a b c b c b a b c b a b c
   (c) Assume string s = "a pattern matching algorithm" with 28 characters, and p = "rithm" with 5 characters. Illustrate the matching process of the KMP algorithm graphically (as in the handout ppt file) and how many character comparisons are performed in the process?

Sol:

a		p	a	t	t	e	r	n		m	a	t	c	h	i	n	g		a	l	g	o	r	i	t	h	m

r	i	t	h	m

(a)

Usage scenario:

- string $s = s_0 \dots s_{n-1}$, pattern $p = p_0 \dots p_{m-1}$, failure function f(j) for pattern p
- We are matching s_i with a partial match of first j character of pattern $p_0 \dots p_{j-1}$

Rule of pattern matching:

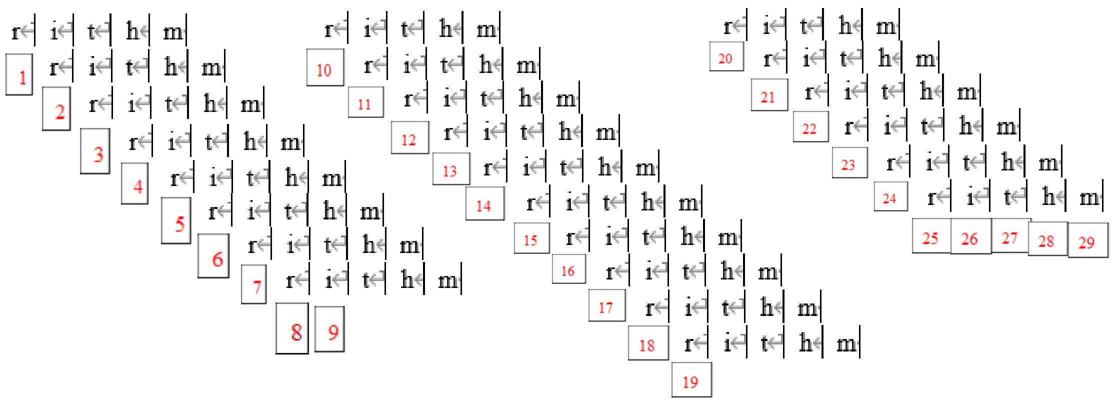
- If a partial match is found such that $s_{i-j} \dots s_{i-1} = p_0 \dots p_{j-1}$, and $s_i \neq p_j$, $j \neq 0$, then resume matching by comparing s_i and $p_{f(j-1)+1}$.
- If $j = 0$, it means that no partial match so far, then continue matching by comparing s_{i+1} and p_0.

(b)

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
P[j]	b	a	b	c	b	c	b	a	b	c	b	a	b	c
f(j)	-1	-1	0	-1	0	-1	0	1	2	3	4	1	2	3

(c)

a		p	a	t	t	e	r	n		m	a	t	c	h	i	n	g		a	l	g	o	r	i	t	h	m



Ans: 29 matches

10. (10%) String matching by Boyer-Moore algorithm

(a) Please describe the Boyer-Moore Bad Character Heuristics pattern matching algorithm.

(b) Compute the last occurrence function for the following string if the alphabet is Σ = {a, b, c, d}:

b a b c b c b a b c b a b c

(c) Assume string s = “a pattern matching algorithm” with 28 characters, and p = “rithm” with 5 characters. Illustrate the matching process of BM algorithm graphically (as in the handout ppt file) and how many character comparisons are performed in the process?

Sol:

a		p	a	t	t	e	r	n		m	a	t	c	h	i	n	g		a	l	g	o	r	i	t	h	m

r	i	t	h	m

(a)

Usage scenario:

- string $s = s_0 \dots s_{n-1}$, pattern $p = p_0 \dots p_{m-1}$

Rule of pattern matching:

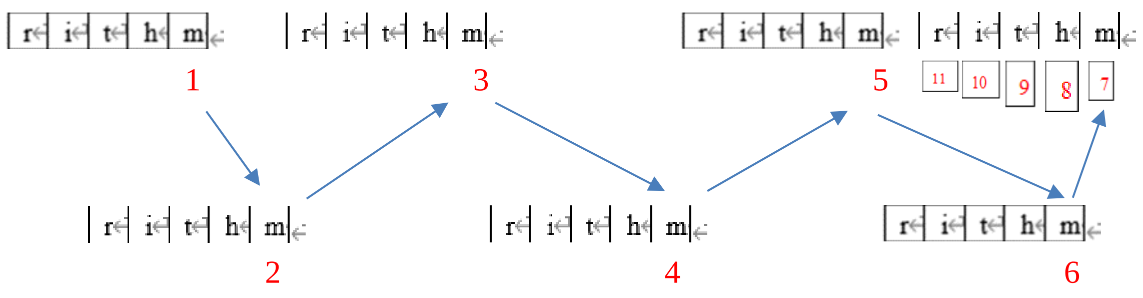
- Looking-glass heuristic: Compare p with s. (from the rightmost to the leftmost of s)
- Character-jump heuristic: When a mismatch occurs at $s_i = c$ (c is the character mismatched),
    - If pattern p contains character c, then shift p to align the last occurrence of c in p with s_i.
    - If not, shift p to align p_0 with s_{i+1}.

(b)

c	a	b	c	d
L(c)	11	12	13	-1

(c)

a		p	a	t	t	e	r	n		m	a	t	c	h	i	n	g		a	l	g	o	r	i	t	h	m



Ans: 11 matches