**EECS2040 Data Structure Hw #1 (Chapter 1, 2 of textbook) due date 3/21/2022 by 109070025 林泓錩**.

**Part 1 (2% of final Grade)**

1. (10%) Using the ADT1.1 *NaturalNumber* in the textbook pp.10, add the following operations to the *NaturalNumber* ADT: Predecessor, *IsGreater*, *Multiply*, *Divide*.

**Answer:**

Predecessor(x): NaturalNumber ::= if(x==0) Predecessor = x

else predecessor = x-1

IsGreater(x, y): Boolean ::= if(x>y) IsGreater = TRUE

else IsGreater = FALSE

Multiply(x, y): NaturalNumber ::= if(x\*y<=MAXINT) Multiply = x\*y

else Multiply = MAXINT

Divide(x, y): NaturalNumber ::= if(y>0) Divide = x/y

else Divide = MAXINT

1. (10%) Determine the frequency counts for all statements (by step table) in the following two program segments:

**Answer:**

Code (a): s/e frequency subtotal

1. for(i=1; i<=n; i++) 1 n+1 n+1
2. for(j=1; j<=i; j++) 1
3. for(k=1; k<=j; k++) 1
4. x++; 1

Total:

**Answer:**

Code (b) s/e frequency subtotal

1. i=1; 1 1 1
2. while(i<=n) 1 n+1 n+1
3. { 0 0 0
4. x++; 1 n n
5. i++; 1 n n
6. } 0 0 0

Total: 3n+2

1. (10%) For the function Multiply() shown below,
2. Introduce statements to increment count at all appropriate points and compute the count
3. Simplify the resulting program by eliminating statement and compute the count
4. Obtain the step count for the function using the step table method.
5. Repeat (c) but use () 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];

}

}

**Answer:**

(a)

void Multiply(int \*\*a, int \*\*b, int \*\*c, int m, int n, int p)

{

for(int i=0;i<m;i++){

count++;

for(int j=0; j<p; j++){

count++;

c[i][j] = 0;

count++;

for(int k=0;k<n;k++){

count++;

c[i][j] += a[i][k] \* b[k][j];

count++;

}

count++;

}

count++;

}

count++;

}

(b)

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];

count+=2;

}

count+=3;

}

count+=2;

}

count++;

}

(c) s/e frequency subtotal

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)

c[i][j] = 0; 1 m\*p m\*p

for(int k=0;k<n;k++) 1 m\*p\*(n+1) m\*p\*(n+1)

c[i][j] += a[i][k] \* b[k][j]; 1 m\*p\*n m\*p\*n

Total: 2mpn+3mp+2m+1

(d) s/e frequency subtotal

for(int i=0;i<m;i++) 1 ʘ(m) ʘ(m)

for(int j=0; j<p; j++) 1 ʘ(mp) ʘ(mp)

c[i][j] = 0; 1 ʘ(mp) ʘ(mp)

for(int k=0;k<n;k++) 1 ʘ(mpn) ʘ(mpn)

c[i][j] += a[i][k] \* b[k][j]; 1 ʘ(mpn) ʘ(mpn)

Total: ʘ(mpn)

1. (10%) A complex-valued matrix X is represented by a pair of matrices (A, B) where A and B contains real values.
2. Write a C++ program that computes the product of two complex-valued matrices (A, B) and (C, D), where (A, B) \* (C, D) = (A+iB)\*(C+iD) = (AC-BD) + i(AD + BC).
3. Determine the number of additions and multiplications if the matrices are all nxn.

**Answer:**

(a)

#include<iostream>

using namespace std;

class Matrix {

public:

​ float real\_part;

float imag\_part;

};

void Multiply(Matrix \*\*a, Matrix \*\*b, Matrix \*\*c, int m, int n, int p)

{

for(int i=0;i<m;i++) {

for(int j=0; j<p; j++)

{

​c[i][j].real\_part = 0;

​c[i][j].imag\_part = 0; ​

​for(int k=0;k<n;k++) {

​c[i][j].real\_part += (a[i][k].real\_part \* b[k][j].real\_part - a[i][k].imag\_part \* b[k][j].imag\_part);

​c[i][j].imag\_part += (a[i][k].real\_part \* b[k][j].imag\_part + a[i][k].imag\_part \* b[k][j].real\_part);

​}

}

}

}

int main() {

​int n;

​cin>>n;

​Matrix \*\*a = new Matrix \*[n], \*\*b = new Matrix \*[n], \*\*c = new Matrix \*[n];

​for (int i = 0; i < n; i++)

​​a[i] = new Matrix [n];

​for (int i = 0; i < n; i++)

​​b[i] = new Matrix [n];

​for (int i = 0; i < n; i++)

​​c[i] = new Matrix [n];

​for (int i = 0; i < n; i++)

​​ for (int j = 0; j < n; j++) {

​​​ cout<<"a["<<i<<"]["<<j<<"] = ";

​​​ cin>>a[i][j].real\_part;

​​​ cin>>a[i][j].imag\_part;

​​ }

}

​for (int i = 0; i < n; i++){

​​ for (int j = 0; j < n; j++) {

​​​ cout<<"b["<<i<<"]["<<j<<"] = ";

​​​ cin>>b[i][j].real\_part;

​​​ cin>>b[i][j].imag\_part;

​​ }

​}

​Multiply(a, b, c, n, n, n);​

​for (int i = 0; i < n; i++) {

​​ for (int j = 0; j < n; j++) {

​​​ cout<<"c["<<i<<"]["<<j<<"] = "<<c[i][j].real\_part<<" + "<<c[i][j].imag\_part<<"j"<<endl;

​​ }

​}

​for (int i = 0; i < n; i++) {

​​ delete [] a[i];

​​ delete [] b[i];

​​ delete [] c[i];

​}

​ delete a;

delete b;

​ delete c;

​

​return 0;

}

(b)

一次矩陣相乘有 乘和 (n – 1)加

一次矩陣相加有 次加

(AC – BD) + i(AD + BC) 有 4 次矩陣相乘 2 次矩陣相加

addition : 4 (n – 1) + 2 = 4 – 2

multiplication :

1. (10%) The Tower of Hanoi is a classical problem which can be solved by recurrence. There are three pegs and N disks of different sizes. Originally, all the disks are on the left peg, stacked in decreasing size from bottom to top. Our goal is to transfer all the disks to the right peg, and the rules are that we can only move one disk at a time, and no disk can be moved onto a smaller one. We can easily solve this problem with the following recursive method: If N = 1, move this disk directly to the right peg and we are done. Otherwise (N >1), first transfer the top N − 1 disks to the middle peg applying the method recursively, then move the largest disk to the right peg, and finally transfer the N −1 disks on the middle peg to the right peg applying the method recursively. Let T(N) be the total number of moves needed to transfer N disks.

(a) Prove that T(N) = 2T(N −1) + 1 with T(1) = 1.

(b) Unfold this recurrence relation to obtain a closed-form expression for T(N). (T(N) is expressed in terms of function of N.)

**Answer:**

(a)

First we move N-1 disks to the middle peg which costs T(N-1) steps,then we move the last disks from left peg to right peg which costs 1 step , finally we move the N-1 disks from the middle peg to the right peg which costs T(N-1) steps,so the sum of steps is 2T(N-1)+1.

(b)

T(1) = 1

T(2) = 2\*T(1) + 1

T(3) = 2 \* T(2) + 1 = \* T(1) + 2 \* T(1) + 1

…

T(N) = + + +…. + 1 = = -1

1. (10%) For the polynomial represented by dynamic array of tuples,
2. Design an algorithm for performing multiplication of two polynomials.
3. Then analyze its time complexity using () notation, assuming the number of terms of the two polynomials are m and n, respectively.

**Answer:**

(a)

int multiply(int A[], int B[], int m, int n)

{

int \*product = new int [m + n – 1]; // Multiply two polynomials term by term

for (int i=0; i<m; i++) // Take every term of first polynomial

{

/\* Multiply the current term of first polynomial

with every term of second polynomial.\*/

for (int j=0; j<n; j++)

product[i+j] += A[i]\*B[j];

}

return product;

}

(b)

(mn)

1. (10%) Obtain an addressing formula for the element a[i1][i2]…[in] in an array declared as a[u1][u2]…[un]. Assume a column-major representation of the array with one word per element and a the address of a[0][0]…[0]. In a column-major representation, a two-dimensional array is atored sequentially by columns rather than by rows.

**Answer:**

Address of a[i1][i2]…[in] is :

+( \*\*….) + (\*\*….)

+( \*\*….) + ……. + (\*) + ()

1. (15%) A square band matrix An,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.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | 6  下方帶狀區域 | 7 | 8 | 0 | | 8 | 0 | 4 | 4 | | 9 | 3 | 2 | 9 | | 0 | 7 | 6 | 8 |   *A*4,3 | *a* 條對角線  上方帶狀區域   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | 0 | |  | |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  | | *n*列 |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  | |  | 0 | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |   *n* 行  *An*,*a*  主對角線 |

1. How many elements are there in the band of An,a?
2. What is the relationship between i and j for element Aij in the band of An,a?
3. Assume that the band of An,a is stored sequentially in an array b by diagonals starting with the lowermost diagonal. Thus A4,3 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 |
| A20 | A31 | A10 | A21 | A32 | A00 | A11 | A22 | A33 | A01 | A12 | A23 | A02 | A13 |

Obtain an addressing formula for the location of an element Aij in the lower band of An,a, e.g., LOC(A20) = 0, LOC(A31) = 1 in the example above.

**Answer:**

(a)

Main diagonal has n elements, and there are a-1 diagonals below and above the main diagonal, where there are n-x elements in the diagonal if there are x+1 elements between the diagonal and the main diagonal. Hence, there are

n + [(n-1)+(n-2)+…..+(n-a+1)]\*2 = n + (2n-a)\*(a-1) elements in the band.

(b)

The element Aij in the band of An,a. Through observation, we can find that the elements in the same diagonal will have the same quantity of (i-j), for when i increases/decreases k, j also increases/decreases k. Moreover, we can know that

| i-j | <= a-1 because we know i-j = 0 in the main diagonal, and the diagonal which is the farthest from main diagonal has | i-j | = a-1.

(c)

We can observe that for i–j = a–1, LOC is from 0 ~ a – 2.

For i–j = a-2, LOC is from a-2 ~ 2a-2……

we can conclude that LOC( Aij )=( )+ j , when i-j>=0.(lower band)

1. (15%) A generalized band matrix An,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.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *b*  *a* |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | 0 | |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| *n*列 |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | 0 | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

主對角線

*n* 行

*An*, *a*, *b*

1. How many elements are there in the band of An,a,b?
2. What is the relationship between i and j for element Aij in the band of An,a,b?
3. Obtain a sequential representation of the band of An,a,b in one-dimensional array c.

**Answer:**

(a)

The main diagonal has n elements, and there are [(n-1)+(n-2)+….(n-a+1)] elements in the left-and-lower region in band, and there are

[(n-1)+(n-2)+….+(n-b+1)] elements in the right-and-upper region in band.

Therefore, there are [(n-1)+(n-2)+….+(n-a+1)+(n-1)+(n-2)+….+(n-b+1)]+n

= n+(2n-a)\*(a-1)+(2n-b)\*(b-1)/2 elements.

(b)

If the element is in the upper-right region of band, -b < i-j < 0

If the element is in the main diagonal, i-j = 0

If the element is in the lower-left region of band, 0 < i-j <a

(c)

We can observe that for i–j = a–1, LOC is from 0 ~ a – 2.

For i–j = a-2, LOC is from a-2 ~ 2a-2……

we can conclude that LOC( Aij )=( )+ j , when i-j>=0.(lower band)

LOC(Aij )= () + i, when i–j < 0. ( upper band )