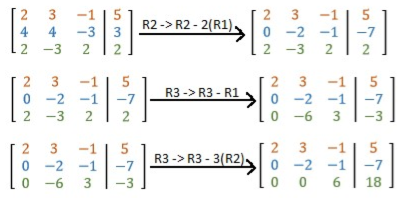
10/6 Ch. 2 Array

1. A hacker is transmitting a virus through the internet. The virus has a special pattern denoted as P, and the value is “11101101”. Now we want to detect whether a downloaded file includes this kind of virus.
2. We are going to use the KMP algorithm to detect, and need to build a failure function. Please fill out the blank below.

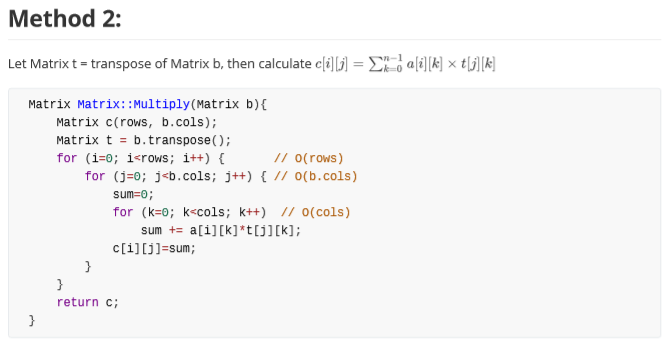
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P[i] | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| F(i) | -1 | 0 |  |  |  |  |  |  |

1. Please implement the failure function in C++.
2. Please evaluate the big-O of this program.
3. 給定一組N\*N 且儲存在二維陣列中的矩陣和一組向量，請求出N個未知數，計算其算法的時間複雜度。

****

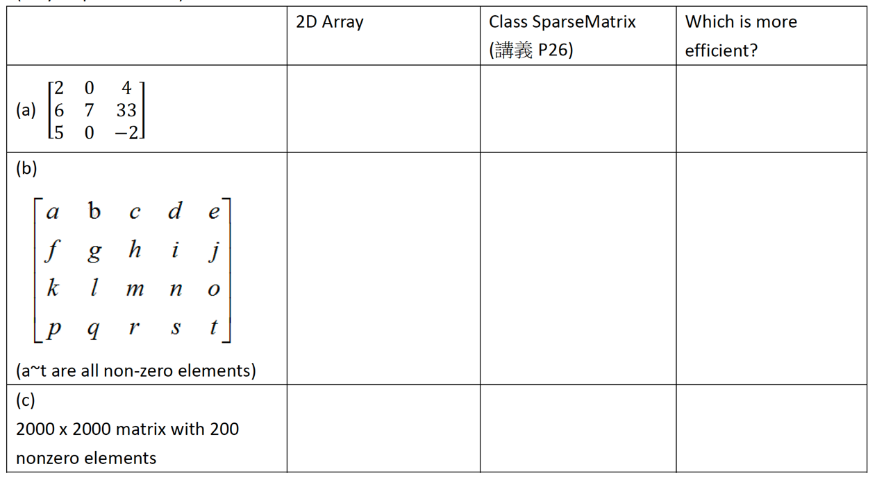
1. In the class, the teacher introduces two kinds of polynomial representation and its addition function.
2. Please analyze the advantages and disadvantages of these two methods from the aspect of memory, runtime and its algorithm of operations. (p.s. consider whether the polynomial is sparse or not)
3. After comparison of the two methods, which representation should we use for polynomial operation?
4. Compare the two following versions of Matrix Multiplication. Which version runs faster on the same computer when the size of the matrix is large?

****

****

1. Compare the time complexity of the two Methods.
2. Which method runs faster on a normal computer when n is large?
3. Please write down the pseudo code of the inner product of two -degree vectors (向量內積), while most vector elements have value zero. Please utilize the fact of sparsity and implement an efficient the code suitable for cases of large .
4. 計算下列failure function
5. aaab
6. babaa
7. abaabaabb
8. Consider the sparse matrix A as shown below. Suppose that each non-zero element in the matrix is stored by a triple <row, column, value>. And all the triples are stored in an array in the row-major order.
9. Draw the array that represents matrix A.

Consider the fast transpose algorithm. The goal is to obtain the transpose of sparse matrix A step by step.

1. Write down the array that stores <rowSize, rowStart> paris for all rows in .
2. Write down the array that represents (in which non-zero elements of are represented by <row, column, value> triples).
3. Explain what a sparse matrix is?
4. ****Please calculate the approximate memory requirements of the matrices in the following for the two given methods and tell which one is more efficient.
5. In the lecture video, we learned the basic concept of the Knuth-Morris-Pratt algorithm. To perform pattern matching using this algorithm, we need to find the (failure) function of the pattern first.
6. Now, please find the function of the following two patterns:
7. BCABCABAABCA
8. ABCDEFGZZXYJJ
9. If we want to perform pattern matching with these two strings using the Knuth-Morris-Pratt algorithm, which pattern will have significant advantages over the conventional one-by-one pattern matching method? Why? Compare these two algorithm’s (KMP and 1-by-1) time complexity in the worst case.
10. 上課投影片的稀疏矩陣儲存方式是一維陣列儲存三元組的<列，行，組>資料，例如: 可以下列方式儲存

|  |  |  |  |
| --- | --- | --- | --- |
|  | 列 | 行 | 值 |
| S[0] | 0 | 0 | 1 |
| S[1] | 0 | 2 | 3 |
| S[2] | 1 | 0 | 5 |
| S[3] | 2 | 1 | 7 |

而另一種稀疏矩陣儲存方式為，將稀疏矩陣非零的數值依序儲存，而原本的 矩陣有無非零數值，則以 1、0 表示，上方例子則變成

請比較兩者memory使用情形。

11. Please fill in the “TODO” part below to complete the constructor, assuming that you use the code below to store a polynomial.

|  |
| --- |
| < Note > This is not a complete code, just part of it  class Term{  friend Polynomial;  private:  float coef; //coefficient  int exp; //exponent  public:  Term():coef(0),exp(0){}  Term(float coef, int exp):coef(coef),exp(exp){}  }  class Polynomial{  private:  Term \*termArray;  int capacity; // capacity is the size of termArray  int terms; // terms is the number of elements  public:  public Polynomial(int coefficient[], int size){  // TODO  // Note that 0 should not be necessary to create Term instance  }  }  int main(){  int coefficient[5]=[2 0 0 4 2]; //This means 2x4+4x3+2  Polynomial poly(coefficient,5);  } |

12. When do we use Sparse Matrix to store a matrix ?

|  |
| --- |
| class SparseMatrix;  class MatrixTerm{  friend class SparseMatrix;  private:  int row,col,value;  };  class SparseMatrix{  private:  int rows,cols,terms,capacity;  MatrixTerm \*smArray;  } |

If we want to store the matrix below, what shall smArray contain?

1 0 3 0 1

0 0 0 1 0

0 0 1 0 0

0 0 0 1 1

13. The time complexity of a 2-dimensional matrix multiplication is O(n3). It costs a lot in computing when you want to multiply two really large matrices. Please discuss whether the order of the matrix multiplication influence the computing time. For example, you may try to change the order of M1\*M2 to M2\*M1 and compute the result.

14. In data analysis region, we often “resize” the matrix. For example, we resize the 4x4 matrix into 2x8 matrix( After resizing, the total elements must be the same ).

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1

1 0 0 0 0 0 1 0

0 1 0 0 0 0 0 1

→

The bottom side of the matrix would be move to the right-hand side of the upper submatrix, in another case, if we resize the 2x6 matrix into 3\*4 matrix, the example is below.

1 0 0

0 1 0

0 5 2

0 2 3

1 0 0 0 5 2

0 1 0 0 2 3

→

Can you write down the pseudocode of “resize(int i, int j)” which i and j means row and column after resizing?( Assume that the matrix has been stored in class SparseMatrix as we mentioned in problem 2 )

[ BONUS : Demo this function in class and explain ]

15. Use “single linear list” and “one liner list per row” to represent the sparse matrix below :

A[6][6]=

15 0 0 22 0 -15

0 11 3 0 0 0

0 0 0 6 0 0

0 0 0 0 0 0

91 0 0 0 0 0

0 0 28 0 0 0

16. Please write down the execution result of the code and answer the following question. Which parameter passing type (pass by value or pass by address) does the foo function use? Why not the other type?

#include <iostream>

using namespace std;

void foo(int a[]) {

a[0] += 1;

a[1] += 2;

a[2] += 3;

}

int main() {

int a[] = {1, 2, 3};

cout << a[0] << a[1] << a[2] << endl;

foo(a);

cout << a[0] << a[1] << a[2] << endl;

return 0;

}

17. Consider the following declaration of a two-dimensional array in C++: Assuming that the main memory is byte-addressable and that the array A is stored starting from memory address 0, what is the address of A[40][50] ? There are two possible answers based on different storage method. (Row-base and column-base)

# 18. How to find the size of array without using sizeof ? Finish the “TODO” part below.

#include <bits/stdc++.h>

using namespace std;

int main()

{

    int  arr[] = {1, 2, 3, 4, 5, 6};

    int size;

//TODO

    cout << "Number of elements in arr[] is "

         << size;

    return 0;

}

19. What is the difference between code A and B ? Explain why the difference?

A:

#include<iostream>

using namespace std;

int main(){

    char a[100];

    char b[100];

    cin.get(a , 100);

    cin.getline(b , 100);

    cout << a <<b;

}

B:

#include<iostream>

using namespace std;

int main(){

    char a[100];

    char b[100];

    cin.getline(a , 100);

    cin.get(b , 100);

    cout << a <<b;

}

20. We need to build a multiplier to compute the power of 11, but the output of the multiplier, for example 11100, often exceeds the valid integer range (-2,147,483,648 to 2,147,483,647). Please describe how to deal with this situation using array.