1. Let’s assume that there are two functions A and B, and the time complexity of each is:

TA(n): 2𝑛^2+4𝑛

TB(n): 12𝑛

(1) Does function A run faster when 𝑛 = 3 ?

(2) Is function B faster when 𝑛 = 10 ?

(3) What is the Big-O(O()) of the two functions?

(4) For overall performance, which function is more efficient?

1. Please rank 1~5 for the following cases from the best time complexity (the fastest) to the worst when n is big enough (ex: n = 100000)

( ) f1 = θ(n)

( ) f2 = θ(2^n)

( ) f3 =θ(log n)

( ) f4 = θ(n^2)

( ) f5 = θ(nlogn)

1. Given the following definitions of two asymptotic notations:

Big-O: 𝑓(𝑛) = 𝑂(𝑔(𝑛)) iff there exist 𝒄, 𝒏𝟎 > 𝟎 such that 𝒇(𝒏) ≤ 𝒄𝒈(𝒏) for all 𝒏 ≥ 𝒏𝟎 Omega: 𝑓(𝑛) = Ω(𝑔(𝑛)) iff there exist 𝒄, 𝒏𝟎 > 𝟎 such that 𝒇(𝒏) ≥ 𝒄𝒈(𝒏) for all 𝒏 ≥ 𝒏𝟎.

Please find out c and n0, for each corresponding complexity notations.

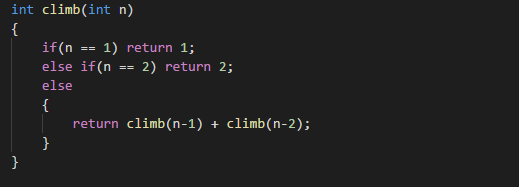
(a) 𝑓

(b)

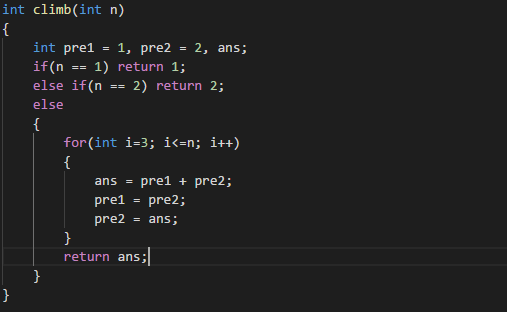
1. According to the lecture video, a recursive code usually performs poorly, in memory usage, than a pure iterative code, while the time complexity appears to be the same. Explain why we often still adopt recursions in coding?
2. 計算以下兩種算法的 Time Complexity in Big-O

爬樓梯，一次一階或兩階，爬 n 階樓梯有幾種方法

(1)



(2)



1. Please write down the Pseudo Code of Fibonacci series using the iterative method.

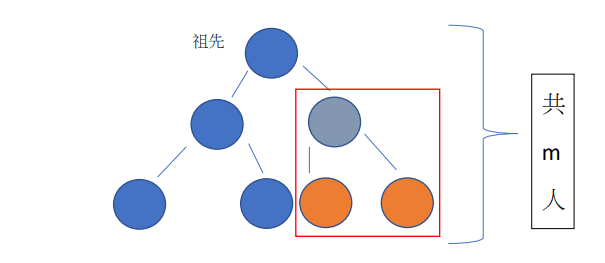
(2) Please write down the Pseudo Code of Fibonacci series using the recursive method.

(3) Please calculate their performance using big-O notation, and which one is more efficient?

1. 給定一張族譜，其中只有一個祖先，每人都有 2 個小孩，配偶不必納入族譜。族譜總人數為 m 人，試找出此家族中 n 個成員彼此之間親等關係的時間複雜度(Big O, upper bound)。

親等的計算方式為:

假設我們想要計算兩個橘色的親等關係，我們先向上溯源到共同的祖先(灰色)， 由於灰色和兩個橘色的距離都是 1，所以兩個橘色之間的親等關係為: 1+1 = 2



1. Which of the following statements is correct and why the other is wrong?
   1. is the most efficient among the following time complexity functions: .
   2. Suppose that A, B, C are matrices of size , the time complexity for the matrix multiplication C=A\*B is .
   3. A pair of iterators define a sequence which starts from the first element and ends at the last element.
   4. A feasible recursive algorism may have increased parameter values.

9.

(1) Please write down the Pseudo Code of binary search using “while loop”?

(2) Please write down the Pseudo Code of binary search using “recursive”?

(3) Please analyze the time complexity of the above two Pseudo Code.

10.

|  |
| --- |
| **#include <iostream>**  **#define n 8**  **int nums[n] = {7,3,1,2,6,8,4,5};**  **void merge(int l, int m, int r){**  **int i,j,k;**  **int n1=m-l+1, n2=r-m;**  **int L[n1], R[n2];**  **for(i=0; i<n1; i++)**  **L[i]=nums[l+i];**  **for(j=0;j<n2;j++)**  **R[j]=nums[m+1+j];**  **i=j=0;**  **k=l;**  **while(i<n1 && j<n2){**  **if(L[i] >= R[j])**  **nums[k++]=L[i++];**  **else**  **nums[k++]=R[j++];**  **}**  **while(i<n1)**  **nums[k++]=L[i++];**  **while(j<n2)**  **nums[k++]=R[j++];**  **}**  **void sort(int l, int r){**  **if(l>=r) return;**  **int m=(l+r)/2;**  **sort(l,m);**  **sort(m+1,r);**  **merge(l,m,r);**  **}**  **int main(){**  **sort(0, n-1);**  **for(int i=0; i<n; i++)**  **std::cout << nums[i] << " ";**  **return 0;**  **}** |

Please analyze the code above, and answer the questions.

1. Calculate the Space complexity of the sorting algorithm and show the results in Big-O (𝑂) notation.

𝑆(𝑃 ) = 𝐶 + 𝑆𝑃 (𝐼) where

• 𝐼 = 𝑛 (number of elements to be sorted)

• 𝐶 = 𝑐𝑜𝑛𝑠𝑡𝑎𝑛𝑡

1. Calculate the Time complexity of the sorting algorithm and show the results in Big-O (𝑂), Omega (Ω), Theta (Θ) notations.

𝑇 (𝑃 ) = 𝐶 + 𝑇𝑃 (𝐼) where

• 𝐼 = 𝑛 (number of elements to be sorted)

• 𝐶 = 𝑐𝑜𝑛𝑠𝑡𝑎𝑛𝑡

1. Write down the output result of the code above

11. STL is the abbreviation of Standard Template Library. It is a very powerful tool to implement data structure in C++. Here we will simply explain Stack and Queue data structures with examples of codes. Please make sure that you know the concept of these data structures well and answer the questions. Details of these data structures will be introduced in future chapters.

(A) Stack

|  |
| --- |
| **[ Concept ]**    **You can consider stack as a box on the ground. Push means that you put something into this box, and Pop means that you take something out of the box. Pop always takes the top-most one out of the box, and the top-most one is also the last one pushed into the box. As a result, Stack is also named LIFO( Last In First Out ) algorithm.** |
| **[ Sample Code ] : Please give the output of this code, and explain why the result.**  **#include<iostream>**  **#include<stack>**  **using namespace std;**  **int main(){**  **stack<int> s;**    **for(int i=0 ; i<5 ; i++){**  **s.push(i\*10);**  **}**  **s.pop();**  **s.pop();**    **while(s.size()!=0){**  **cout<<s.top()<<endl;**  **s.pop();**  **}**  **}** |

(B) Queue

|  |
| --- |
| **[ Concept ]**    **You can consider a queue as a group of people who are getting in line to buy something. The one who is going to get in line is to “Enqueue”, and the person is placed at the end of the line. The one who came earliest in the line is at the front of the line and the front-most one will be the first to take action for the purpose of the queue and then leave the line afterward. This is the concept of “Dequeue”. However, you should be aware that in C++ queue STL “Enqueue” is named “push”, and “Dequeue” is “pop”.** |
| **[ Sample Code ] : Please give the output of this code, and explain why the result.**  **#include<iostream>**  **#include<queue>**  **using namespace std;**  **int main(){**  **queue<int> q;**  **for(int i=0;i<5;i++){**  **q.push(i\*10);**  **}**  **while(q.size()!=0){**  **cout<<q.front()<<endl;**  **q.pop();**  **}**  **}** |

12.

|  |
| --- |
| **#include<iostream>**  **#include<vector>**  **using namespace std;**  **int main(){**  **vector<int> first; // Empty vector of ints**  **vector<int> second(4,100); // Question A**  **vector<int> third(second.begin(), second.end()); //Iterating through second**  **vector<int> fourth(third); // A copy of third**    **// The iterator constructor can also be used to construct from arrays**  **int myints[]={16,2,77,29};**  **vector<int> fifth(myints,myints+sizeof(myints)/sizeof(int)); // Question B**    **cout<<"The contents of fifth are : ";**  **vector<int>::iterator it=fifth.begin();**  **for( it ; it!=fifth.end() ; it++ ){**  **cout<<\*it<<" "; // Question C**  **}**  **cout<<endl;**  **}** |

Answer the following questions based on the code above.

(A) What does this vector constructor “**vector<int> second(4,100);**” do?

(B) What does this vector constructor “**vector<int> fifth(myints,myints+sizeof(myints)/sizeof(int));**” do?

(C) Why do we need to dereference “it” for cout?

13. Please design an algorithm to compute Binomial Coefficient C(n,m), and analyze the space complexity and time complexity of your algorithm.

14. A two-dimensional matrix multiplication is given below. A and B are two *n*x*n* two-dimensional matrices, and C is the output of A\*B.

|  |
| --- |
| **void matrixmult(int n, const number A[][], const number B[][], number C[][])**  **{**  **int i. j, k;**  **for( i=1 ; i<=n ; i++ ){**  **for( j=1 ; j<=n ; j++){**  **C[i][j]=0;**  **for( k=1 ; k<=n ; k++){**  **// "to fill in"**  **}**  **}**  **}**  **}** |

Try to answer the questions below :

(A) Please fill in the part marked “to fill in”.

(B) Please compute time complexity in Big-O notation and explain why.

(C) [Optional] This is not the most efficient algorithm of matrix multiplication. You can search for faster algorithm of matrix multiplication and share the idea in class.

[ Note : those who provide good answers to question C will receive bonus. ]

15.There are two possible implementations, recursive or iterative, for Fibonacci Sequence. Here we provide the pseudocode of these two versions.

|  |  |
| --- | --- |
| Recursive version(Top-down) | Iterative version(Bottom-up) |
| int fib(int n){  if(n<=1)  return n;  else  return fib(n-1)+fib(n-2)  } | **int fib(int n){**  **int f[n];**  **f[0]=0;**  **if(n>0){**  **f[1]=1;**  **for(int i=2;i<=n;i++)**  **f[i]=f[i-1]+f[i-2];**  **}**  **return f[n];**  **}** |

Try to answer the questions below :

(A) Without computing time complexity of these two algorithms, can you tell and explain which one is faster?

(B) Discuss the merits and drawbacks of each version.

16.Given the pseudocode of Exchange Sort as below.

|  |
| --- |
| void exchangesort(int n, int S[]){  int i,j;  for( i=1; i<=n; i++ )  for( j=i+1; j<=n; j++ )  if( S[j]<S[i] )  Exchange the value of S[i] and S[j] // Assume T(n)=1  } |

Try to answer the questions below :

(A) Prove that time complexity T(n)=

(B) Prove that T(n)=

[ Hint : For question B, you may start from the definition of Ω ]

17. One day, Billy Gates wants to design a C++ code to sort his employee’s salary. However, he is not quite familiar with C++ STL sorting algorithm tool. He doesn’t know how to implement the function “salaryComparison” highlighted in yellow color below. Please help him to complete the function and you may get a chance to work in his company, Macrosoft.

|  |
| --- |
| #include <iostream>  #include <algorithm>  #include <vector>  #include <stdlib.h>  #include <time.h>  #define employee\_number 50  #define hourlyRate\_max 100  #define time\_max 100  using namespace std;  class employee{  private:  int index; //員工編號  int time; //工作時間  int hourlyRate; //時薪  public:  employee():index(0),time(0),hourlyRate(0){}  employee(int index,int time,int hourlyRate):index(index),time(time),hourlyRate(hourlyRate){}  int getSalary(){  return time\*hourlyRate\*0.95;  }  int getIndex(){  return this->index;  }  };  int main () {  srand((int)time(NULL));    vector<employee> Microsoft;  for( int i=0 ; i<employee\_number ; i++ ){  Microsoft.push\_back(employee(i,rand()%time\_max,rand()%hourlyRate\_max));  }    cout<<"列出所有員工的薪水"<<endl;  for( vector<employee>::iterator it=Microsoft.begin(); it!=Microsoft.end() ; ++it ){  cout<<"員工編號: "<<it->getIndex()<<" ; 薪水: "<<it->getSalary() <<endl ;  }  cout<<endl;    cout<<"將員工薪水做排序"<<endl;  sort (Microsoft.begin(), Microsoft.end(), salaryComparison );  for( vector<employee>::iterator it=Microsoft.begin(); it!=Microsoft.end() ; ++it ){  cout<<"員工編號: "<<it->getIndex()<<" ; 薪水: "<<it->getSalary() <<endl ;  }  cout<<endl;  return 0;  } |

18. Please analyze how many program-steps the following code implementation requires given an int argument *n*?

|  |
| --- |
| float rsum ( float list[] , int n){  if(n)  return rsum(list,n-1) + list[ n-1];  return list[0];  } |

19. Analyze the time complexity (θ) of the following code.

|  |
| --- |
| for(i=0;i<N;i++)  for(j=0;j<i\*i;j++)  for(z=0;z<j;z++)  k++; |

20. Answer TRUE or FALSE to each question and explain why:

(1) *n*log*n*=O()

(2) =O(nlog2n)

(3) O() + O(logn) = O()

21. Rank the following functions ( F(n) ) by the order of growth rate:

log(*n*!)、*n*\*()、(log*n*)!、、*n*!、、log(log*n*)、*n*

22. Finish the following functor (class A) to make the code’s output confined from 11 to 15:

|  |
| --- |
| #include <iostream>  #include <vector>  #include <algorithm>  class A{  };  int main()  {  std::vector<int> v = { 1, 2, 3, 4, 5 };  int val = 10;  for\_each(v.begin(), v.end(), A(val)); // 11 12 13 14 15  } |