資料結構報告

楊政愷

July 30, 2024

目錄

1. 解題說明

第一題先以遞迴的形式完成 Ackermann function，接著再利用 vector 的堆疊特性實現非遞迴

Ackermann's function $A(m,n)$ is defined as follows:

$$A(m,n) = \begin{cases} n+1 & , \text{if } m = 0 \\ A(m-1, 1) & , \text{if } n = 0 \\ A(m-1, A(m,n-1)) & , \text{otherwise} \end{cases}$$

This function is studied because it grows very fast for small values of m and n. Write a recursive function for computing this function. Then write a nonrecursive algorithm for computing Ackermann's function.

```
int Ack_r(int m, int n){
    if(m == 0){//當m = 0
        return n + 1;//回n + 1
    }
    else if(n == 0){//當n = 0
        return Ack_r(m - 1, 1);//call Ack_r(m - 1, 1)
    }
    else{
        return Ack_r(m - 1, Ack_r(m , n - 1));//其餘狀況call Ack_r(m - 1, Ack(m, n - 1))
    }
}
```

Figure1.1:R_Ackermann.cpp

```cpp
class Tnum {//建立類別來存輸入的值
public:
    int m;
    int n;
    Tnum(int m, int n) : m(m), n(n) {}//建構函式
};

int Ack_nr(int m, int n){
    vector<Tnum> stack;//vector建立與輸入相同類別的堆疊
    stack.push_back(Tnum(m, n)); //外部輸入進堆疊

    while (!stack.empty()) {//堆疊淨空即停止
        Tnum p = stack.back();
        stack.pop_back();//取出輸入值計算並清除

        if (p.m == 0) {//m = 0
            if (stack.empty()) {//無殘值返回結果
                return p.n + 1;
            } else {//有殘值將n + 1後放回
                Tnum pPrev = stack.back();
                stack.pop_back();
                stack.push_back(Tnum(pPrev.m, p.n + 1));
            }
        } else if (p.n == 0) {//n = 0,將(m - 1, 1)放入
            stack.push_back(Tnum(p.m - 1, 1));
        } else {
            stack.push_back(Tnum(p.m - 1, -1));//先將(m - 1, -1)放入以做標記
            stack.push_back(Tnum(p.m, p.n - 1));//再將(m, n - 1)放入
        }

        if (!stack.empty() && stack.back().n == -1) {//標記處理, 取出前值並將n + 1
            stack.pop_back();
            if (!stack.empty()) {
                Tnum pTemp = stack.back();
                stack.pop_back();
                stack.push_back(Tnum(pTemp.m, p.n + 1));
            }
        }
    }
    return -1;
}
```

Figure1.2:NR_Ackermann.cpp

第二題以 vector 來實現 sets

```
void powersetHelper(const vector<int>& S, vector<vector<int>>& result, vector<int>& current, int index) {
    if (index == S.size()) {
        result.push_back(current);
        return;
    }
    powersetHelper(S, result, current, index + 1);

    current.push_back(S[index]);
    powersetHelper(S, result, current, index + 1);
    current.pop_back();
}

vector<vector<int>> powerset(const vector<int>& S) {
    vector<vector<int>> result;
    vector<int> current;
    powersetHelper(S, result, current, 0);
    return result;
}

bool compareSets(const vector<int>& a, const vector<int>& b) {
    if (a.size() != b.size())
        return a.size() < b.size();
    return a < b;
```

Figure1.3:Subsets.cpp

!!!部分程式碼與解釋為與 chatGPT 互動之結果

2. 演算法設計與實作

```
int main(){
    int m, n;
    while(cin >> m >> n){//持續輸入m和n
        cout << "Recursive Ackermann : " << Ack_r(m, n) << endl;//輸出計算結果
    }
}
```

Figure2.1:R_Ackermann.cpp

```
int main() {
    int m, n;
    while(cin >> m >> n){//持續輸入m和n
        cout << "Non Recursive Ackermann : " << Ack_nr(m, n) << endl;//輸出結果
    }
}
```

Figure2.2:NR_Ackermann.cpp

```
int main() {
    vector<int> S = {1, 2, 3};

    vector<vector<int>> result = powerset(S);

    sort(result.begin(), result.end(), compareSets);

    cout << "Powersets of Set {1, 2, 3} :" << endl;
    for (const auto& subset : result) {
        cout << "{ ";
        for (int element : subset) {
            cout << element << " ";
        }
        cout << "}" << endl;
    }

    return 0;
}
```

Figure2.3:Subsets.cpp

3. 效能分析

F(n) = O(n)

時間複雜度

R_Ackermann.cpp:

NR_Ackermann.cpp:

Subsets.cpp:

我不會算

空間複雜度

R_Ackermann.cpp:

NR_Ackermann.cpp:

Subsets.cpp:

我不會算

4. 測試與過程

```
PS E:\資料結構> cd 'e:\資料結構\HW\code\output'
PS E:\資料結構\HW\code\output> & .\'R_Ackermann.exe'
3 2
Recursive Ackermann : 29
```

Figure4.1:NR_Ackermann.cpp

```
PS E:\資料結構> cd 'e:\資料結構\HW\code\output'
PS E:\資料結構\HW\code\output> & .\'NR_Ackermann.exe'
3 2
Non Recursive Ackermann : 29
3 1
Non Recursive Ackermann : 13
```

Figure4.2:NR_Ackermann.cpp

```
PS E:\資料結構> cd 'e:\資料結構\HW\code\output'
PS E:\資料結構\HW\code\output> & .\'Subsets.exe'
Powersets of Set {1, 2, 3} :
{ }
{ 1 }
{ 2 }
{ 3 }
{ 1 2 }
{ 1 3 }
{ 2 3 }
{ 1 2 3 }
PS E:\資料結構\HW\code\output>
```

Figure4.3:Subsets.cpp