資料結構報告

姓名:童呈偉

日期:2024/07/30

目錄

1.解題說明

a.Q1

題目翻譯

阿克曼函數 A(m, n) 定義如下：當 m = 0 時，A(m, n) = n + 1;當 n = 0 時，A(m, n) = A(m - 1, 1);否則，A(m, n) = A(m - 1, A(m, n - 1))。這個函數因為在 m 和 n 的小數值下增長非常快而受到研究。請編寫一個遞迴函數來計算這個函數。然後編寫一個非遞迴算法來計算阿克曼函數。

根據題意遞迴公式的程式碼如下:

```
//阿克曼函數
int ackermann(int m, int n)
{
    if (m == 0)
    {
        return n + 1;//如果m=0，則回傳n + 1
    }
    else if (n == 0)
    {
        return ackermann(m - 1, 1);//如果n=0，則回傳ackermann(m - 1, 1)的結果
    }
    else//如果是其他情況便開始遞迴，直到滿足m=0或n=0
    {
        return ackermann(m - 1, ackermann(m, n - 1));
    }
}
```

figure1.1:遞迴.cpp

非遞迴採用堆疊方式進行，程式碼如下:

```
int ackermann_non_recursive(int m, int n) {
    stack<pair<int, int> > s; //創建了一個名為s的堆疊

    s.push(make_pair(m, n));

    while (!s.empty())
        {
        pair<int, int> top = s.top();
        s.pop();//移除頂部的元素
        m = top.first;//first 和 second 分別代表這個 pair 中的第一個和第二個元素，top.first 的值賦給變量 m。
        n = top.second;//top.second 的值賦給變量 n。

        if (m == 0) //如果m=0，則回傳n+1
        {
            n = n + 1;
            if (!s.empty())
            {
                s.top().second = n;//將top.second 的值替換成新的n。
            }
        }
        else if (n == 0)//如果n=0，則回傳A(m - 1, 1)的結果
        {
            s.push(make_pair(m - 1, 1));
        }
        else //當n不為0且m也不為0時，需要計算 A(m - 1, A(m, n - 1))。
        {
            s.push(make_pair(m - 1, 0));
            s.push(make_pair(m, n - 1));
        }
    }
```

figure1.2:非遞迴.cpp

b.Q2

題目翻譯為:

如果 S 是一個包含 n 個元素的集合，那麼 S 的冪集就是 S 的所有可能子集的集合。

例如，如果 S = {a, b, c}，那麼 S 的冪集就是 {(), (a), (b), (c), (a, b), (a, c), (b, c), (a, b, c)}。

請寫一個遞迴函數來計算這個冪集。

根據題意遞迴公式的程式碼如下:

```cpp
// 遞迴函數用來生成集合 S 的冪集
void generatePowerSet(int S[], int n, int index, int current[], int currentSize)
//n 集合大小，index 是當前處理的元素索引，current[] 是暫時儲存當前子集的陣列，currentSize 是當前子集的大小。
{

    // 當前子集完成，輸出結果
    if (index == n)
    {
        cout << "{ ";
        for (int i = 0; i < currentSize; ++i)
          {
            cout << current[i] << " ";
        }
        cout << "}" << endl;
        return;
    }

    // 選擇不包含當前元素的情況
    generatePowerSet(S, n, index + 1, current, currentSize);

    // 選擇包含當前元素的情況
    current[currentSize] = S[index];
    generatePowerSet(S, n, index + 1, current, currentSize + 1);
}
```

figure1.3:Q2.cpp

2.設計與實作

a.Q1

```
//主函數
int main() {
    int m, n;
    cout << "輸入 m 跟 n: ";
    cin >> m >> n;
    cout << "Ackermann(" << m << ", " << n << ") = " << ackermann(m, n) ;
    return 0;
}
```

figure2.1:遞迴.cpp

```
int main() {
    int m, n;
    cout << "輸入 m 跟 n: ";
    cin >> m >> n;
    cout << "Ackermann(" << m << ", " << n << ") = " << ackermann_non_recursive(m, n) << endl;
    return 0;
}
```

figure2.2:遞迴.cpp

b.Q2

```
主函式
int main() {
    int n;
    cout << "輸入陣列大小: ";
    cin >> n;

    int S[n]; // 陣列大小由用戶輸入的數量決定
    cout << "輸入鎮列內容: ";
    for (int i = 0; i < n; ++i) {
        cin >> S[i];
    }

    int current[n]; // 保存當前子集的陣列
    cout << "The power set is:\n";

    // 計算冪集
    generatePowerSet(S, n, 0, current, 0);

    return 0;
}
```

figure2.3:Q2.cpp

3.效能分析

a.Q1

兩種方式皆為阿克曼函數的表示，因此時間複雜度接取決於m的大小

測試時的m值為1、2，故時間複雜度為:

m=1;O(n)

m=2;O(2^n)

b.Q2

時間複雜度主要有兩個部分：生成所有子集的決策樹的時間和輸出每個子集的時間。生成所有子集的時間為2^n每次調用的輸出操作為n。時間複雜度為O(2^n*n)。。

4.測試過程

a.Q1

```
C:\Users\User\Desktop\Ackermann.exe
輸入 m 跟 n: 1 2
Ackermann(1, 2) = 4
--------------------------------
Process exited after 4.226 seconds with return value 0
Press any key to continue . . .
```

figrue4.1:遞迴.exe:input1

```
C:\Users\User\Desktop\Ackermann.exe
輸入 m 跟 n: 2 4
Ackermann(2, 4) = 11
--------------------------------
Process exited after 2.568 seconds with return value 0
Press any key to continue . . .
```

figrue4.2:遞迴.exe:input2

```
C:\Users\User\Desktop\Ackermann2.exe
輸入 m 跟 n: 1 2
Ackermann(1, 2) = 4

--------------------------------
Process exited after 9.592 seconds with return value 0
Press any key to continue . . .
```

figrue4.3:非遞迴.exe:input1

```
C:\Users\User\Desktop\Ackermann2.exe
輸入 m 跟 n: 2 4
Ackermann(2, 4) = 11

--------------------------------
Process exited after 1.752 seconds with return value 0
Press any key to continue . . .
```

figrue4.4:非遞迴.exe:input2

根據阿克曼公式

m=1, n=2

A(1,2)=A(0,A(1,1))

A(1,2)=A(0,A(1,0))

A(1,0)=A(0,1)=2

A(1,1)=A(0,2)=3

A(1,2)=A(0,3)=4

m=2, n=4

A(2,4)=A(1,A(2,3))

A(2,3)=A(1,A(2,2))

A(2,2)=A(1,A(2,1))

A(2,1)=A(1,A(2,0))

A(2,0)=A(1,1)=3

A(2,1)=A(1,3)=5

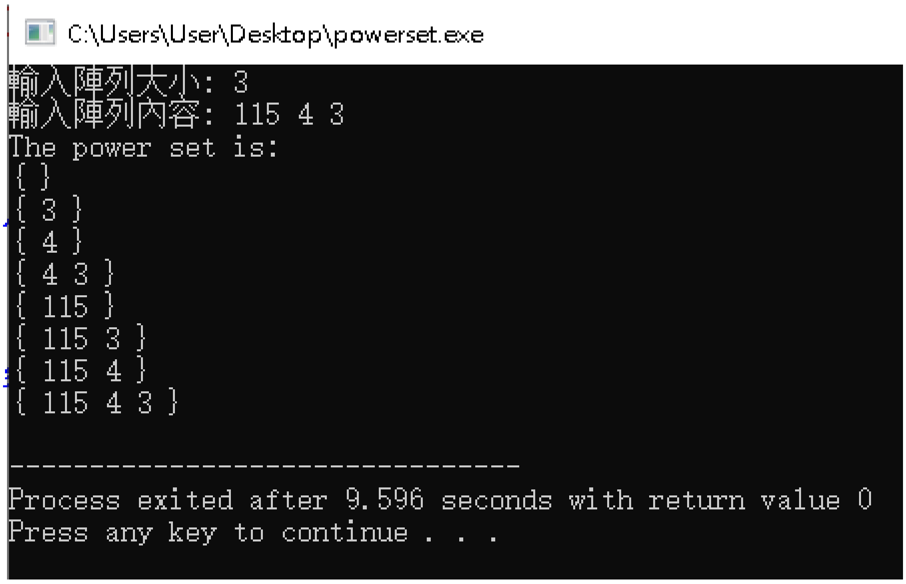
A(2,2)=A(1,5)=7

A(2,3)=A(1,7)=9

A(2,4)=A(1,9)=11

無論是遞迴還是非遞迴都會計算到m=0或者n=0，隨後回傳結果。

b.Q2



figrue4.5:powerset.exe:input1

先輸入陣列大小，接著輸入陣列內容
函數會記錄集合大小、當前處理的元素索引、當前子集合的陣列、子集合的大小。隨後從索引值0開始子集合排序，完成後輸出結果，再來開始遞迴，直到所有集合排序完畢。

{115,4,3}的所有子集合為
{{ },{115},{115,4},{115,3},{4,3},{115,4,3}}剛好符合。

程式碼皆由用chatGTP協助撰寫，由Dev C++編譯執行。