Ch1-Algorithm, Recursion, Time Complexity

Algorithm

完成特定功能之有限個(指令/敘述)所成之集合，且滿足下列5個性質(不包含Correctness)

1.Input: 外界提供 >= 0個輸入Data量

2.Output: 至少需有 >= 1個輸出結果

3.Definiteness(明確性): 每個指令/敘述須是Clear and Unambiguous

4.Finitneness(有限性): 在執行有限次的Step後，必須能夠終止

5.Effectiveness(有效性): 每個指令/敘述須足夠基本可讓人用紙筆即可追蹤Algorithm

有限性是Program與Algorithm的一個差異，即Program不一定滿足有限性，可以Infinite execution，如:人為設計Error、實際應用所需

Recursion

定義:以Direct Recursion為例，Algorithm(or Program)本身含有Self-calling敘述稱之

種類:

1.Direct Recursion

2.Indirect Recursion: 多個模組之間形成Calling Cycle存在

3.Tail Recursion: 在呼叫自己本身遞迴之前，運算已完成，No Stack，屬於Direct Recursion一種

|  |  |
| --- | --- |
| Recursive | Non-Recursive |
| 程式碼較精簡 | 較冗長 |
| 節省Storage空間 | 耗費較多 |
| 表達力較Powerful | Weak |
| 暫存/區域變數較少or無 | 較多 |
| 執行時間長，效能較差 | 執行時間短，效能較好 |
| 需要額外Stack支援 | 無須 |

Compiler對Recursive call之處理機制

1.須先保存當時執行狀況

Push參數值、區域/暫存變數、返回位址(Return Address)到Stack空間

2.控制權轉移

3.某次recursive call結束後，檢查Stack是否為空

是，則exit

否，則pop參數值、區域/暫存變數、返回位址 from stack then go to返回位址

Performance評估

Algo/Program之效能

評估有兩個Criteria

1.Space Complexity

2.Time Complexity

Space Complexity:

令S(P)是Program P之空間需求，則S(P) = C + S(I)，C是正常數

C => 固定空間(fixed space)

包含:

1.Instruction(code) space

2.變數空間

3.常數空間

4.Fixed size結構型變數，如:array、struct、Object

S(I) => 變動空間(Variable Space)

包含

1.參數是結構型且是Call-by-value傳遞

2.Stack空間 for recursion

Time Complexity

令T(P)代表Algo/Program P之時間需求，則T(P) = Development time + Execution Time

運用Analysis技巧，統計Algo/Program中指令執行次數總和，作為預估未來執行時間之基礎