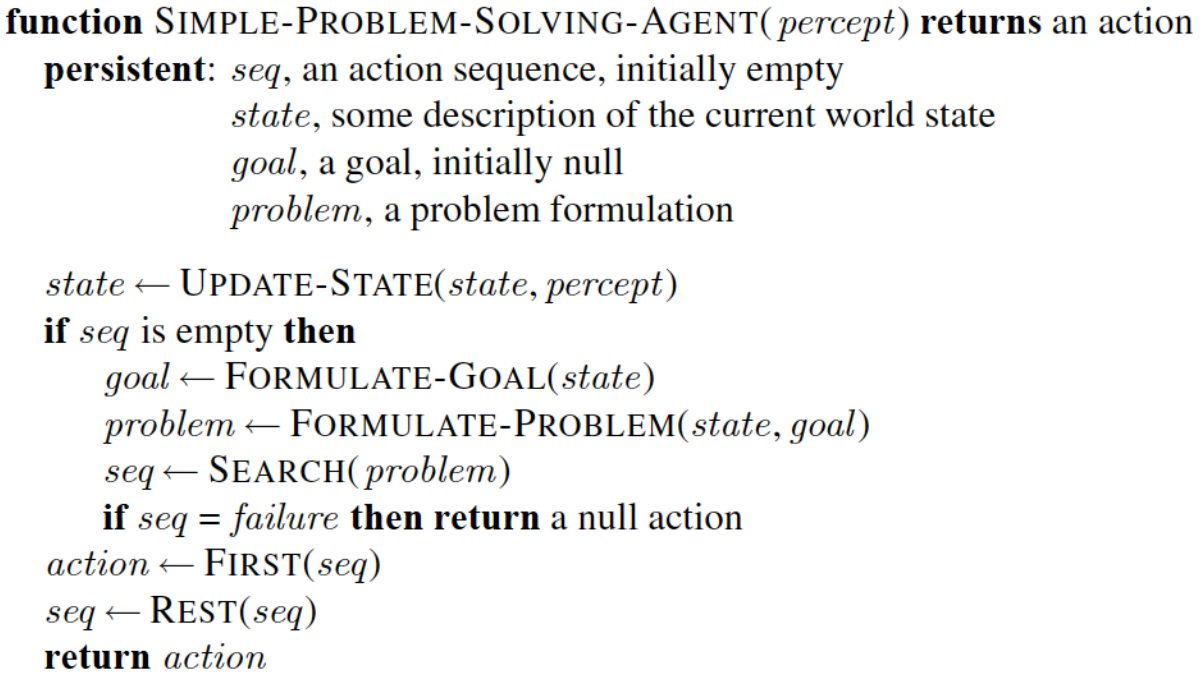
**Homework assignment#1 (Chap3)**

106971001 林上人

1. **Pseudo codes documentation**

Pages: 3



②

①

③

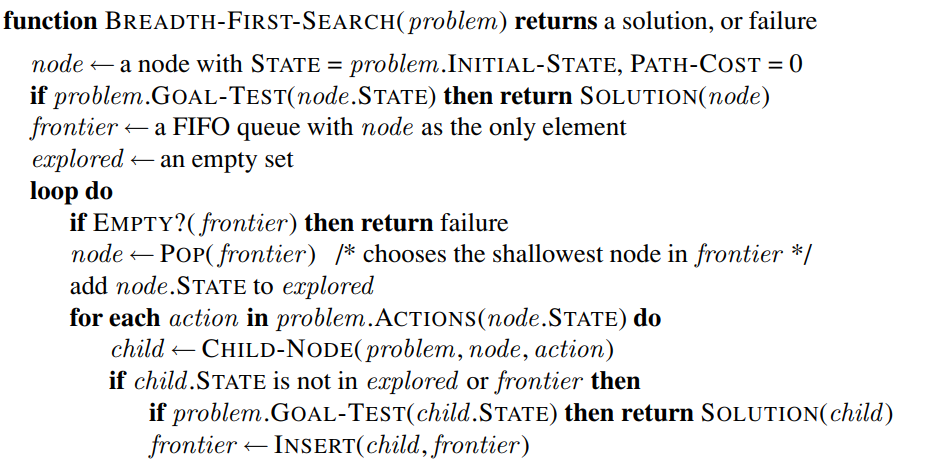
④

簡單的問題求解Agent設計

一開始persistent的部分先説明了後面函數會用到的名詞，包含seq用來存放動作序列，state為對目前的狀態描述，goal表示欲達成的目標，problem則代表問題的描述，意指在給定目標的情況下要考慮哪些狀態和動作的過程。

整個流程包含三個部分，分別為Formulate、Search、Execute，一開始①根據狀態和感知更新自己的狀態,後檢查動作序列是否為空，若不是空的表示目前其實是在一個解決問題的動作過程中,若爲空則進入②(Formulate)的階段,並制定目標和界定問題,然後在③(Search)尋找解決問題的動作序列,最後進入④(Execute)階段,從動作序列取出第一個動作後更新序列為剩餘的動作並將取出的動作回傳,function每次回傳一個動作,直到動作序列為空才會再次進入②(Formulate) 、③(Search)階段。

Pages: 47



③

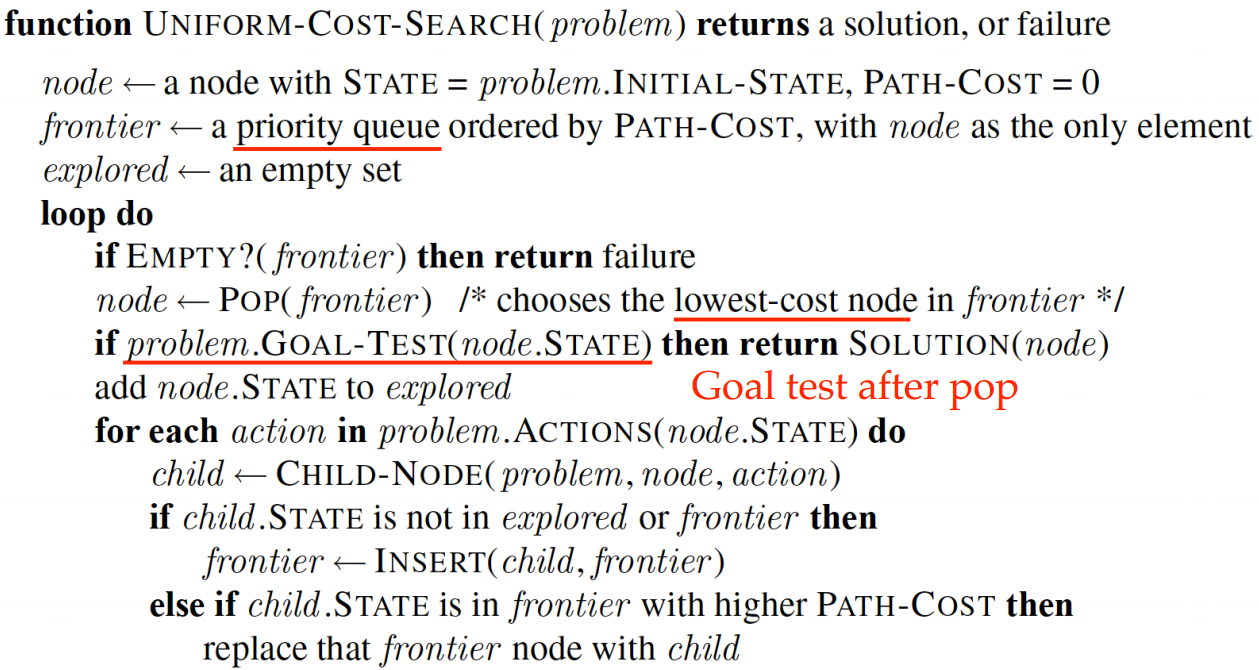
②

①

BFS演算法 - 先檢查最淺節點

此處使用BFS演算法來解決問題，①檢查問題的初始節點是否已經能夠通過GOAL-TEST,通過就回傳Solution，否則初始節點就為第一個frontier佇列元素，並利用FIFO的佇列結構實作frontier以達到BFS先檢查最淺節點的特性，explored集合用來存放已經遍歷過且不屬於frontier的節點,後續的步驟以迴圈執行， ②首先檢查frontier佇列内還有沒有元素，沒有則表示找不到解就回傳failure，否則就從frontier佇列中POP出一個節點，因爲放在佇列中的節點都是沒有通過GOAL-TEST，所以取出後也直接把它加入explored集合，然後③針對取出的節點找出子節點並檢查，如果子節點不屬於explored集合，也不屬於frontier佇列，就先對它做GOAL-TEST,若通過則表示找到解回傳Solution,沒通過就加入frontier佇列之中，整個迴圈會執行到frontier為空回傳failure（②）或child通過GOAL-TEST回傳Solution（③）爲止。

Pages: 61



②

④

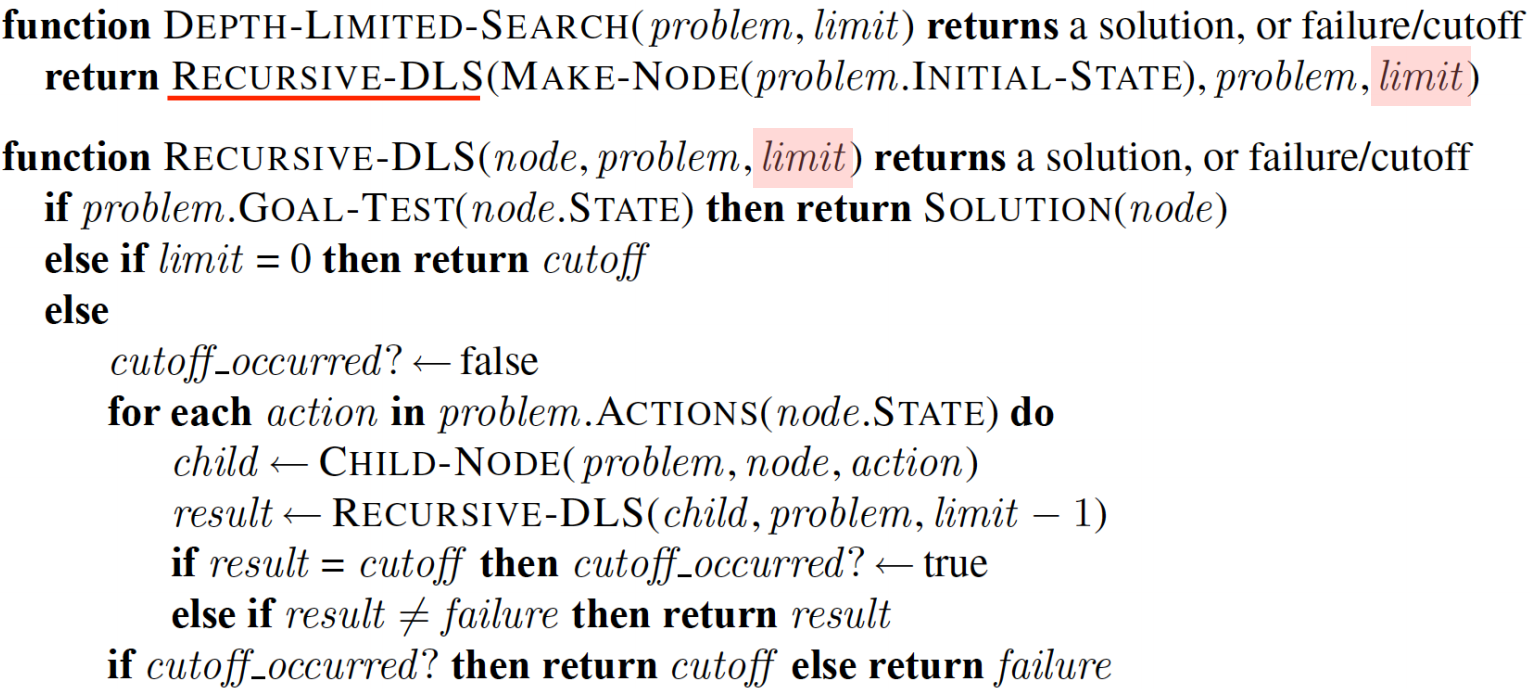
③

①

UCS演算法 – 改良BFS,考慮路徑成本

演算法類似BFS演算法，但①不檢查問題的初始節點而是初始節點直接就為第一個frontier佇列元素，並把frontier佇列改成以PATH-COST排列的優先權佇列，explored集合一樣存放走過的節點,後續的步驟以迴圈執行，②首先檢查frontier佇列内還有沒有元素，沒有則表示找不到解就回傳failure，否則就從frontier佇列中POP出PATH-COST最低的節點，對節點做GOAL-TEST，通過就回傳Solution，否則把它加入explored集合，然後③針對取出的節點找出子節點並檢查，如果子節點不屬於explored集合，也不屬於frontier佇列，就依PATH-COST加入frontier佇列之中，但是④如果子節點已經存在於frontier佇列中，就表示有其他路徑已經到達過此節點，此時就必須比較到達的PATH-COST，如果新的PATH-COST比較低就用這個子節點替換frontier佇列中的節點資料，因此即使目標節點已經先走到過也會在frontier佇列中不斷更新成最低成本的路徑，整個迴圈會因為無解而回傳failure或是執行到目標節點成為frontier佇列中成本最低的節點時通過POP和GOAL-TEST回傳Solution爲止（②）。

Pages: 85



④

⑤

③

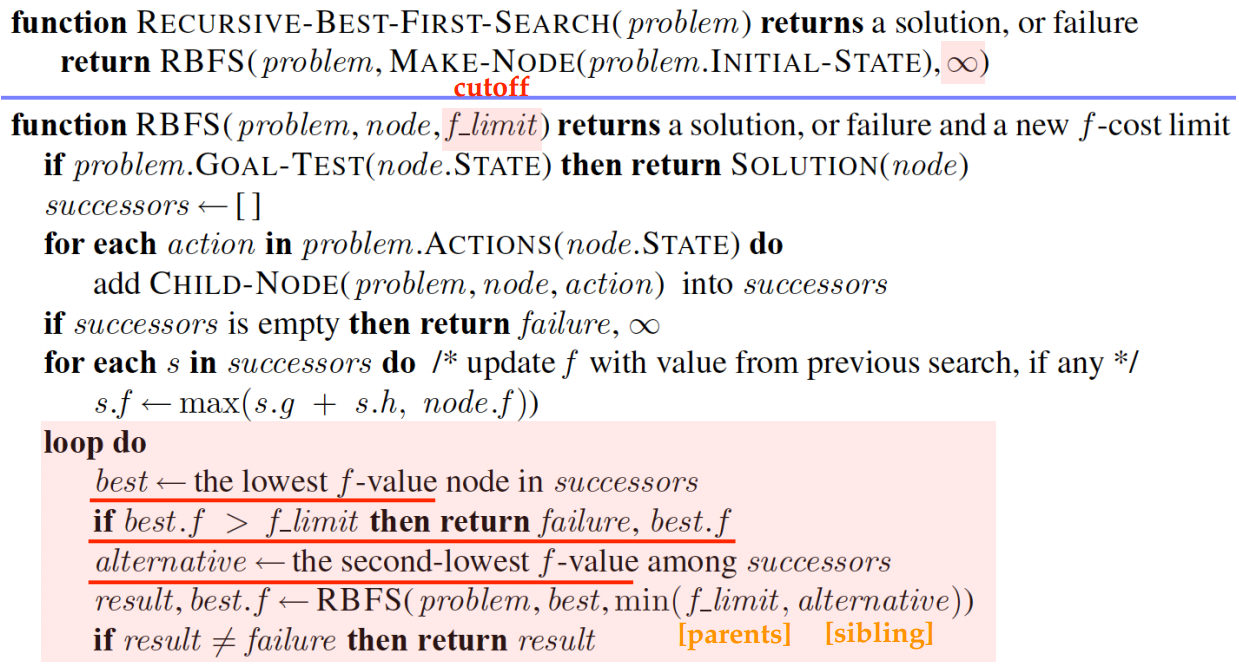
②

①

Depth limited search – 改良原DFS, 限制深度limit

此處以遞迴的方式實作Depth limited search，①Depth limited函式將初始節點和limit作為參數呼叫遞迴函式Recursive-DLS，而Recursive-DLS函式內則②先檢查傳入的參數節點是否可以通過GOAL-TEST，可以就回傳Solution，或③傳入的limit等於0則回傳cutoff，意指已經到達指定的深度，如果都不是則先宣告變數cutoff\_occurred等於false，然後④取出參數節點的子節點，並將子節點和深度-1作為參數呼叫自己，直到遞迴跑完可以取得result，因為Recursive-DLS有三種回傳值，所以result有三種可能分別為Solution、failure、cutoff，檢查result若為cutoff表示已經到達指定深度但是還未找到解，修改cutoff\_occurred值為true，如果result不是failure則代表是Solution，直接回傳result，最後⑤檢查cutoff\_occurred值若是true則回傳cutoff，若是還未達到指定深度但是找不到解就回傳false，這種可能就是子樹的規模比指定的深度小，而Depth limited函式的結果就和Recursive-DLS一樣分為找到解:Solution、未找到解但完成深度limit的搜尋:cutoff、未找到解且搜尋深度沒有達到limit:failure。

Pages: 137



⑤

④

③

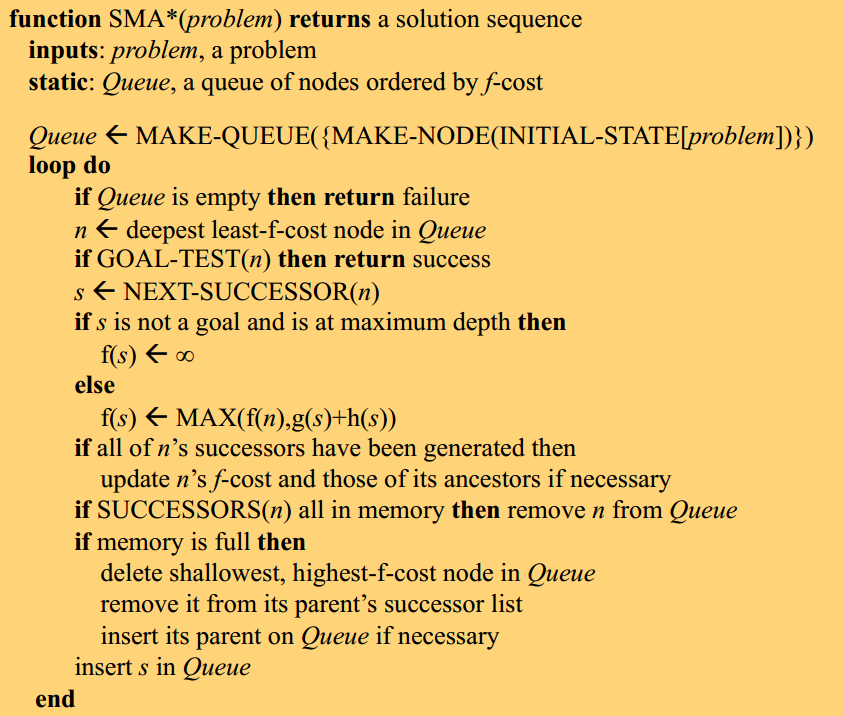
②

①

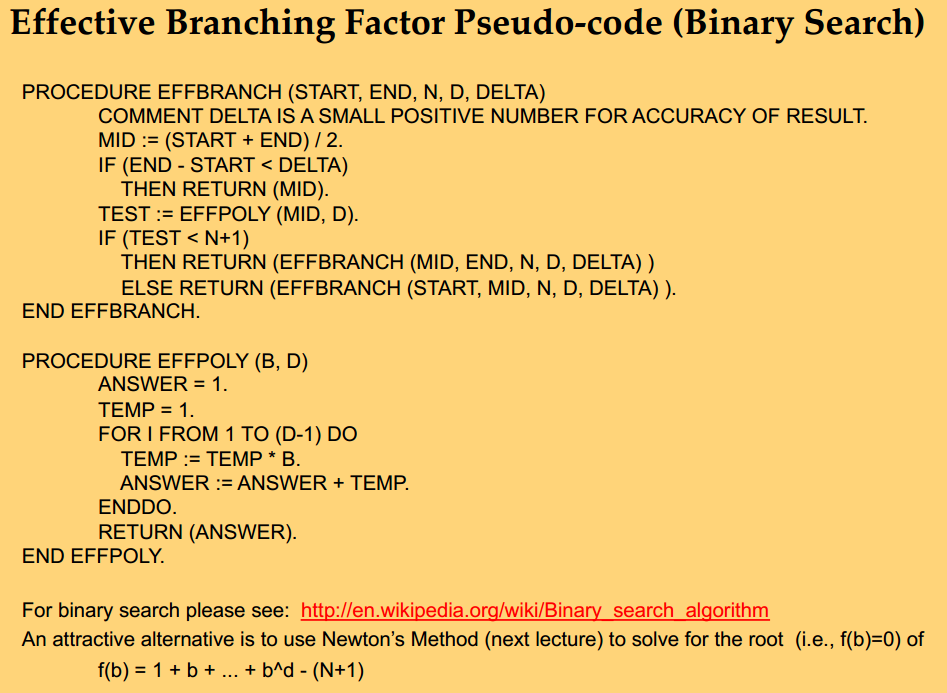
RBFS演算法 – 使用linear space和遞迴模仿standard best-first search

遞迴的方式實作best-first search，①以初始節點和作為參數呼叫遞迴函式RBFS，RBFS②首先檢查傳入的參數節點是否能通過GOAL-TEST，否則③創建一個successors陣列並放入所有的子節點，如果沒有子節點就表示現在這個節點已經沒有路可以走向目標節點，就回傳failure，接著④利用參數node計算successors的f值，f表示Solution經過節點的最低估計成本，所以這邊就代表要計算有通過successor的Solution中最低的估計成本，之後以迴圈方式執行後續動作，⑤先取得f-value最低的子節點與f\_limit相比，因為f\_limit記錄的是祖先節點的最佳估計代價，所以如果f\_limit比較小則表示有其他路較佳，因此僅回傳failure和此路徑知道的最小估計代價，儲存的successors陣列不保留以節省記憶體空間，如果f-value比較小則拓展這個子節點，以parents節點或sibling節點中較小的f-value作爲新的f\_limit參數呼叫遞迴函式RBFS，RBFS的回傳值有兩種可能，一個是Solution就直接回傳，另一個則是failure和一個數值，而這個數值即代表這個節點下的subtree的估計代價會大於等於這個數值，因此我們就以這個值替換原本的f-value,接著繼續跑迴圈直到整個

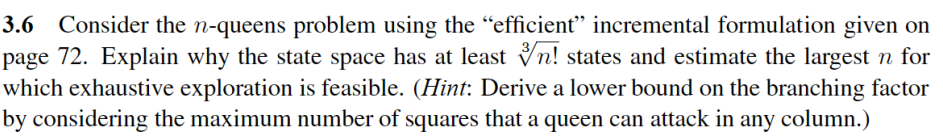
Pages: 146



Pages: 154



1. **Exercises**



Ans

