1a ACTIVITAT D'AVALUACIÓ

Algorítmica i Complexitat

Grau en Enginyeria Informàtica

22 d'abril de 2022

Laura Haro Escoi

Guillem Mora Bea

Índex

Pseudocodi de l'algoritme iteratiu 2

Pseudocodi de l'algoritme recursiu 2

Explicació dels algoritmes i els seus costos 3

Iteratiu 3

Recursiu 3

Costos experimentals 4

Iteratiu 4

Recursiu 5

Conversió d'iteratiu a recursiu 5

Pseudocodi de l'algoritme iteratiu

```
function bookerineManagement_iterativo(reserves):
    listReservats <- set()
    for reserva <- 0 in length(reserves):
        listReservats.add(int(reserva[1]))
    for index <- 0 in length(listReservats):
        if index not in listReservats:
            return index
    return index + 1
```

Pseudocodi de l'algoritme recursiu

```
function bookerineManagement_recursivoAux(reserves, llistaReserves,
index):
    if len(reserves) > 0:
        reserva <- reserves.pop()
        llistaReserves.add(int(reserva[1]))
        return bookerineManagement_recursivoAux(reserves,
llistaReserves)
    if index not in llistaReserves:
        return index
    return bookerineManagement_recursivoAux(reserves, llistaReserves,
index + 1)
```

```
function bookerineManagement_recursivo(reserves):
    listReservats <- set()    //diccionari
    index <- 0
    return bookerineManagement_recursivoAux(reserves, listReservats,
index)
```

Explicació dels algoritmes i els seus costos

Iteratiu

En l'algoritme iteratiu hem utilitzat un diccionari en el que guardarem com a valors els números de les taules reservades.

De les tuples que rebem de reserves, podem prescindir del primer element, ja que el nom de la persona és irrellevant.

Per a guardar els valors utilitzarem un iterador for en el que recorrerem tota la llista de reserves per a guardar-les dins del diccionari.

Aquest procediment, té cos O(n), ja que hem de recórrer n vegades la llista de reserves.

A continuació, fem ús d'un altre iterador en el que recorrem el nostre diccionari per a trobar la posició lliure amb la clau més petita. En aquest cas, també tenim un cos de O(n).

Finalment retornem el valor de la taula lliure més petita.

- El cost total seria de O(n) + O(n) => O(2n) => O(n)

Recursiu

En l'algoritme recursiu, hem fet ús d'una funció auxiliar, ja que necessitàvem més paràmetres d'entrada per a poder fer la recursivitat.

En la funció principal hem creat un diccionari de python buit, ja que el necessitàvem com a paràmetre i un índex inicialitzat a 0 que utilitzarem per a recorre'l.

A continuació, fem un return amb la funció recursiva auxiliar creada on li passarem la llista de reserves, el diccionari i l'índex.

En aquesta funció, primerament emplenarem recursivament el diccionari amb els valors de les taules reservades. Per a fer això, primer comprovem que la llargada de la llista de reserves sigui més gran que 0. Si això és complís, faríem un pop() el qual treu el valor de la llista i el diposita en una variable (redueix la mida de la llista). Després, afegim amb un add() el valor de la taula al diccionari. Finalment fem un return de la crida recursiva sense canvia els paràmetres d'entrada.

Si la condició de que la llargada de la llista de reserves fos més gran que 0 no es complís, comprovaríem si el índex que ens han passat no està dins del diccionari. Si això és complís, retornaríem aquest índex, ja que seria la taula amb el nombre més petit lliure. En canvi, si aquesta condició no fos certa, faríem una crida recursiva de la funció augmentant el valor de l'índex.

Aquesta funció, al seguir el mateix procediment que la iterativa, hauria de tenir el mateix cost, en aquest cas O(n).

Això és pot comprovar sabent que mentre entri la funció al primer if, el cost serà de O(n), ja que estem recorrent tota la llista de reserves. A continuació, quan entrés al else, també serà O(n), ja que recorrerà el diccionari comprovant si l'índex pertany a aquest. Això ho repetirà n vegades com a màxim.

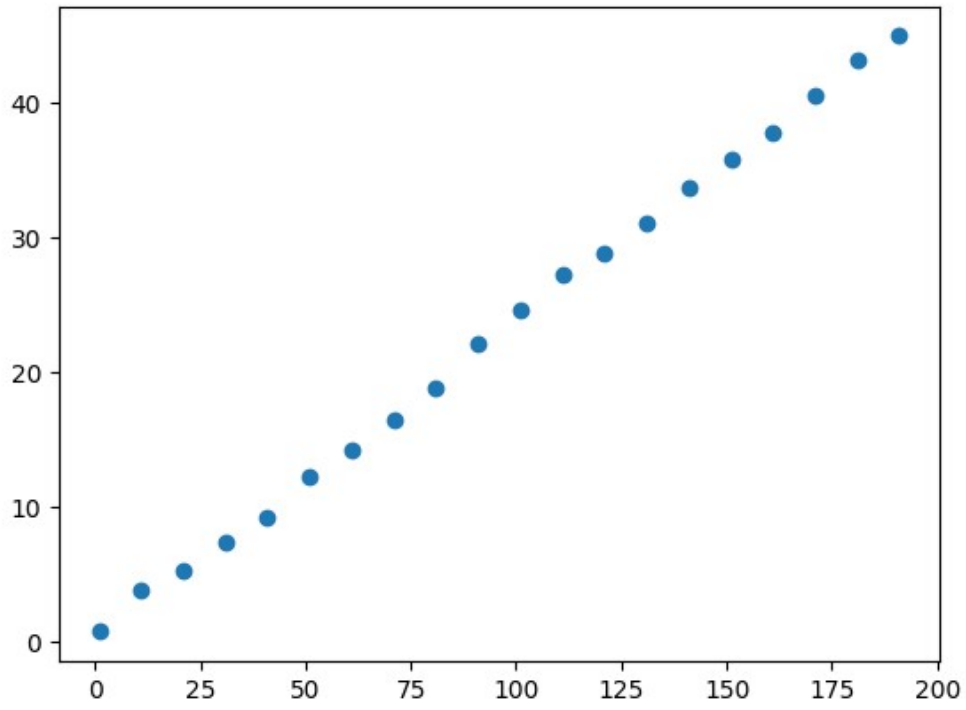
Per tant, podem veure com tots els elements de la funció tenen el mateix cost, i és sumen, ja que no estan uns dins d'altres.

- El cost total seria de O(n)

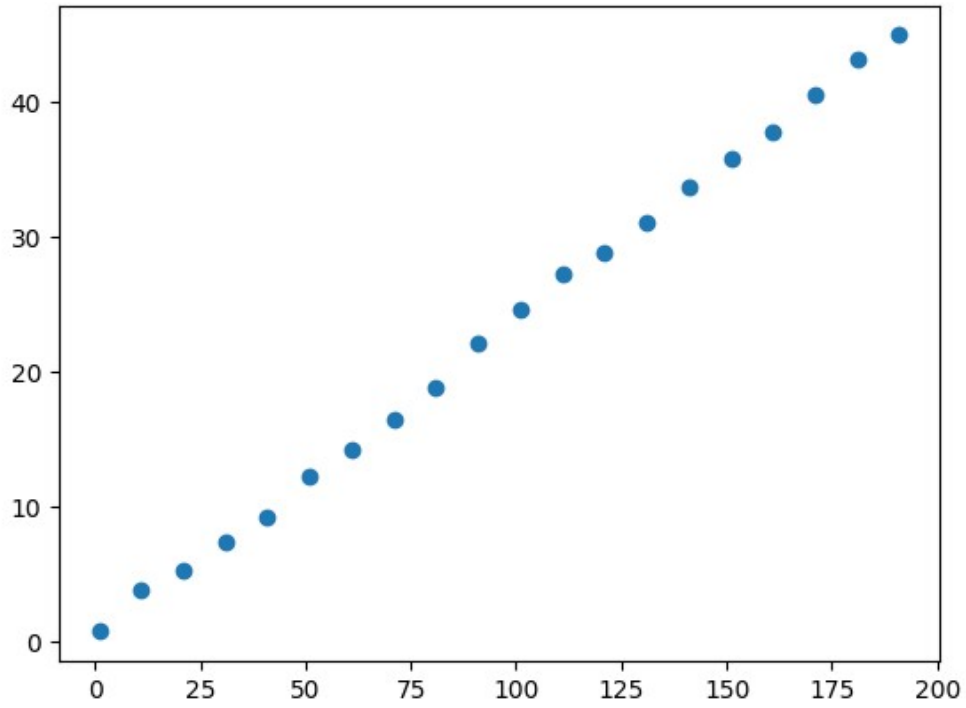
Costos experimentals

Amb la funció donada al main, hem pogut comprovar els nostres càlculs dels costos amb la gràfica que aquest ens retorna.

Iteratiu



Recursiu



Gràcies a aquest gràfics, podem veure la tendència que segueixen els nostres programes. S'aprecia que aquests tenen un creixement lineal, per tant, podem dir que la funció que representen es f(x) = n. Això demostra que els costos dels nostres algoritmes són de O(n).

Conversió d'iteratiu a recursiu

En els dos algoritmes, hem fet de la llista de reserves, d'un diccionari i d'un índex.

Aquests elements en el recursiu els inicialitzem al principi de la funció i en l'iteratiu en el for, amb la diferència de que en l'iteratiu, seguim amb el tractament dels elements amb diferents iteradors i en el recursiu cridem una nova funció auxiliar que és la que aplicarà les crides recursives.

El primer for de l'iteratiu es canvien per un if, els qual marca quan s'ha arribat al final de la llista de reserves i en els dos casos guardem el numero de la taula.

A continuació, hem canviat el següent for de l'iteratiu per la resta de la funció, ja que dins de l'if hi ha un return. En aquesta part es comprova tant en un com en l'altre si l'índex pertany al diccionari. Si és així el retornem, i sinó, seguirem recorrent el bucle fins arribar al final. Si això passa, retornem el valor de l'index + 1. En el recursiu, aquest return no és necessari ja que el + 1 el fem a la crida recursiva.