Ex2.4.1 : Solve the following recurrence relations.

1. x(n) = 2x(n − 3) for n > 1, x(1) = 1

x(n) = 2x(n-3) = 2\*2\*x(n-6) = 2\*2\*2x(n-9) = x(1) \* 2 \* 2 \* 2 \* …. \* 2 (n-1/3 phần tử)

= 2^(n-1/3)

=> O(2^n)

b. x(n) = x(n − 2) − 2 for n > 1, x(1) = 0

x(n) = x(n-2) - 2 = x(n-4) – 2 – 2 = x(1) – 2 – 2 – 2 - …. – 2 (n-1/2) phần tử

= 1 – (n-1)

=> O(n)

Ex2.4.2 :

Recurrence relations :

F(n) = 1 + F(n/2)

F(0) = 0

Solve :

F(n) = 1 + F(n/2) = 1 + 1 + F(n/4) = 1 + 1 + 1 + F(n/8)

= F(1) + 1 + 1 + 1 + … + 1 (log2(n) phần tử)

= log2(n)

* O(logn)

**Bài toán xếp balo :**

* **Phân tích bài toán :** bài toán yêu cầu tìm một cách chọn đồ vật sao cho tổng giá trị là lớn nhất. Như vậy ta có thể xem xét từng đồ vật xem trong trường hợp lấy và không lấy đồ vật thì giá trị sẽ thay đổi ra sao và lấy ra giá trị lớn hơn.
* **Thuật toán :** 
  + **Input :** 
    - n : số lượng đồ vật
    - p : trọng lượng các đồ vật
    - v : giá trị các đồ vật
    - m : sức chứa balo
  + **Output :** 
    - Các vật được chọn thỏa mãn tổng giá trị lớn nhất
  + Nếu n = 0 hoặc w = 0 => return 0 , []
  + Nếu trọng lượng vật thứ n lớn hơn sức chứa, ta bỏ qua vật này (gọi đệ quy chuyển tiếp tới đồ tiếp theo mà ko giảm sức chứa balo), Nếu vật thỏa mãn trọng lượng thì ta xét 2 trường hợp :
    - Cho vật vào balo (gọi đệ quy tới vật tiếp theo và giảm sức chứa balo)
    - Không cho vật vào balo (Gọi đệ quy tới vật tiếp theo và không giảm sức chứa balo)
  + So sánh 2 trường hợp trên và trả về các vật có giá trị lớn hơn
* **Chứng minh tính đúng :**
  + Tại trường hợp cơ sở ta có n = 0 hoặc m = 0, kết quả trả về sẽ là mảng rỗng (**Hiển nhiên**)
  + Giả sử thuật toán đã đúng với n , ta sẽ chứng minh nó đúng với n + 1.
  + Trường hợp đồ vật thứ n + 1 được cho vào balo, thuật toán sẽ tính toán đệ quy giá trị lớn nhất có thể nhận được từ n đồ vật còn lại với sức chứa trừ đi sức chứa của vật thứ n + 1.
  + Trường hợp đồ vật thứ n không được cho vào balo, thuật toán sẽ tính toán đệ quy giá trị lớn nhất có thể nhận được từ n đồ vật còn lại với sức chứa không đổi.
  + Vì ta đã giả sử thuật toán đúng với n vậy Sau khi so sánh 2 trường hợp ta nhận được kết quả cho n + 1 độ vật . Vậy thuật toán đúng với mọi số lượng đồ vật.
* **Độ Phức tạp :**
  + Với mỗi đồ vật ta đều xét 2 trường hợp ( đều gọi đệ quy ) vì vậy số lần gọi đệ quy sẽ là 2 ^ n
  + Vậy độ phức tạp thuật toán : O(2^n)
* **Triển Khai thuật toán :** file best\_items\_selection.py