Introducción

En esta entrada supondremos que conoces las nociones básicas de sucesiones. También supondremos que conoces a grandes rasgos el teorema de recursión en los números naturales. De no ser así, puedes revisar el siguiente material:

- Material de sucesiones en https://blog.nekomath.com/srp/
- Material de la Unidad 1 en https://blog.nekomath.com/as2/

El teorema de recursión en los números naturales nos dice, a grandes rasgos, que podemos crear funciones en los números naturales «haciendo referencia a términos anteriores». Esta noción la podemos llevar al contexto de sucesiones, a la de resolución de problemas de conteo y a la de creación de algoritmos. En esta entrada discutiremos algunas de estas aplicaciones.

Sucesiones recursivas

Una sucesión en un conjunto X consiste simplemente en definir, para cada entero $n\geq 0$ un elemento x_n en X. Una manera de hacer esto es **recursivamente**, es decir, explicando cómo es el valor de un término x_n en función de los términos anteriores $x_0,x_1,\ldots,x_{n-1}$. Si tenemos una **sucesión recursiva** así de general, basta con definir el término x_0 y la regla recursiva para construir toda la sucesión.

Ejemplo. Consideremos la sucesión recursiva dada por

$$\begin{aligned} y_0&=1\\ y_n&=y_0+y_1+y_2+\ldots+y_{n-1}+1. \end{aligned}$$

Su primer término es $y_0=1$. Si queremos encontrar y_1, podemos usar la regla recursiva en $n=1$ para obtener

$$y_1=y_0+1=1+1=2.$$

Para encontrar su segundo término, usamos de nuevo la regla recursiva, ahora para $n=2$ para obtener

$$y_2=y_0+y_1+1=1+2+1=4.$$

Calculando un par de valores más tenemos

$$\begin{aligned} y_3&=y_0+y_1+y_2+1=1+2+4+1=8\\ y_4&=y_0+y_1+y_2+y_3+1=1+2+4+8+1=16. \end{aligned}$$

Si bien tenemos una fórmula recursiva, en este caso parece aparecer un patrón que nos da una fórmula cerrada: las potencias de dos. Para demostrar que $y_n=2^n$ se puede utilizar inducción fuerte. $\square$

En algunas ocasiones queremos que nuestra sucesión dependa de los k últimos valores que definimos, es decir, que x_{n+k+1} esté en términos de $x_{n+1},\ldots,x_{n+k}$. Cuando este es el caso, usualmente debemos definir los primeros k valores de la sucesión para de ahí definir los demás. En este caso decimos que la sucesión es **recursiva de orden** k. Un ejemplo que ya hemos visto es la sucesión de Fibonacci: los primeros dos términos son 0 y 1. Luego, cada término depende de los dos anteriores. Así, la sucesión de Fibonacci es una sucesión recursiva de orden 2. Veamos otro ejemplo.

Problema. Considera las sucesiones a_n y b_n dadas por las siguientes reglas.

$$\begin{aligned} a_0&=0, a_1=0, a_2=1\\ a_{n+3}&=a_{n}a_{n+1}a_{n+2}-a_{n}-a_{n+1}-a_{n+2}. \end{aligned}$$

y

$$\begin{aligned} b_0&=0, b_1=1, b_2=1\\ b_{n+3}&=b_{n}b_{n+1}b_{n+2}-b_{n}-b_{n+1}-b_{n+2}. \end{aligned}$$

Demuestra que una de ellas nunca excede el valor 2, mientras que la otra toma valores tan grandes como queramos.

Solución. Ambas son sucesiones recursivas de orden 3. Tienen exactamente la misma regla de recurrencia. Sin embargo, sus valores

iniciales difieren: $a_1=0$, mientras que $b_1=1$. Explorémos ahora estas sucesiones de manera computacional. Esto lo hacemos en las siguientes celdas de código, en donde mostramos los primeros 12 valores de cada sucesión.

```
x,y,z=0,0,1
an=[]
for j in range(12):
    an.append(x)
    x,y,z=y,z,x*y*z-x-y-z

print(an)
```

```
[0, 0, 1, -1, 0, 0, 1, -1, 0, 0, 1, -1]
```

```
x,y,z=0,1,1
bn=[]
for j in range(12):
    bn.append(x)
    x,y,z=y,z,x*y*z-x-y-z

print(bn)
```

```
[0, 1, 1, -2, -2, 7, 25, -380, -66152, 628510507, 15799345654000345, -6568929816410241531974738217 80]
```

De acuerdo a la exploración computacional, parece ser que a_n es la que nunca excede 2. De hecho, podemos conjeturar algo más fuerte: que a_n es cíclica con un ciclo de periodo 4 que es $0,0,1,-1$. Esto puede demostrarse por inducción.

También, parece ser que b_n es la que sus valores pueden ser tan grandes como queramos. Esto también puede demostrarse por inducción, aunque la prueba no es tan directa. $\square$

El problema anterior nos muestra que sucesiones recursivas con la misma regla de recursión y puntos de partida muy parecidos pueden tener comportamientos muy diferentes. Además, la segunda sucesión debe sugerirnos que una regla recursiva no siempre tiene una fórmula cerrada sencilla de expresar.

Recursión para resolver problemas

Cuando definimos recursivamente nos basamos en «casos más pequeños que ya hayamos hecho». Esto funciona muy bien no sólo para definir cosas, sino también para resolver problemas. Además, tiene la ventaja de combinarse bien con la técnica de buscar un patrón, pues para ambas resulta de mucha utilidad resolver casos pequeños.

Hay algunos problemas de conteo que se prestan para realizar soluciones recursivas. Veamos un ejemplo:

Problema. ¿Cuántas palabras de 100 letras A, B o C hay, en las cuales nunca aparece AB en la palabra?

Solución. Para una letra hay tres posibles palabras: A, B y C. Para dos letras hay 8 posibles palabras: AA, AC, BA, BB, BC, CA, CB y CC. En cuanto tenemos tres letras los casos se empiezan a complicar. Para resolver el problema hay que introducir dos ideas clave: partir nuestra cuentra en tres partes y establecer una recursión.

Así, sean a_n, b_n y c_n las palabras con n letras que nos interesan y que, respectivamente, terminan en A, B o C. Por los primeros casos que hicimos tenemos:

$$\begin{align*} a_1=1, b_1=1, c_1=1\\ a_2=3, b_2=2, c_2=3. \end{align*}$$

La ventaja de partir el problema y estudiarlo más en general es que ahora podemos traducir la hipótesis de que nunca aparece AB en una condición algebraica recursiva. Tenemos que:

$$\begin{align*} a_{n+1}&=a_n+b_n+c_n\\ b_{n+1}&=b_n+c_n\\ c_{n+1}&=a_n+b_n+c_n. \end{align*}$$

La primera igualdad se debe a que una palabra de $n+1$ letras que nos interesa y que termine en A puede ser formada a partir de agregar A al final a una de n letras que termine en a, o una de n letras que termine en B, o una de n letras que termine en C. La explicación es análoga para la tercera igualdad. Sin embargo, si una palabra de $n+1$ letras de las que nos

termine en \(C\). La explicación es análoga para la tercera igualdad. Sin embargo, si una palabra de \(n+1\) letras de las que nos interesan termina en \(B\), no pudo haber sido al agregarle una \(B\) al final a una que terminara en \(A\), pues en ese caso se crearía el patrón \(AB\) prohibido.

Ya argumentada la validez de la recursión, podemos encontrar computacionalmente los valores \(a_{100},b_{100},c_{100}\) que nos interesan para sumarlos y dar el resultado pedido. Hacemos esto a continuación.

```
a,b,c=1,1,1
for j in range(99):
    a,b,c=a+b+c,b+c,a+b+c
print(a+b+c)
```

```
73454486715781809323490890211044929642335 1
```

Veamos ahora un problema que viene de datos reales. \(\square\)

Problema. En el siguiente código se carga la información de la inflación por año en México desde 1970 a 2020.

```
import pandas as pd

df_inflacion=pd.read_csv('Inflacion.csv',index_col="Year")
display(df_inflacion.head(3))
print("...")
display(df_inflacion.tail(2))
```

Year	Inflation
1970	4.69
1971	4.96
1972	5.56

...

Year	Inflation
2019	2.83
2020	3.15

Se quiere saber cuál es la máxima cantidad de años que se pueden tomar, de menor a mayor, pero no necesariamente seguidos, para los cuales la inflación disminuyó. Por ejemplo, se puede verificar que \(1980, 1989, 1999, 2004, 2009\) nos da cinco años en los cuales la inflación disminuyó, pues tuvo los siguientes valores:

```
df_inflacion.loc[[1980,1989,1999,2004,2009]]
```

Year	Inflation
1980	29.84
1989	19.69
1999	12.32
2004	5.19
2009	3.57

¿Será esta la lista más larga de años que podemos conseguir? La idea para resolver el problema es pensar de manera recursiva. La lista más larga que termina en \(1970\) consiste de únicamente ese año. ¿Cuál es la lista más larga que termina en el año \(n\)?

- Si todas las inflaciones anteriores son más chicas, entonces la lista consistirá sólo del año \(n\).
- Si en algún año \(k\) anterior la inflación es mayor, entonces entonces el año \(n\) puede alargar la lista más larga que termina en \(k\).

Así, la lista más larga que termina en el año \(n\) será de \(1\) año, o de lo máximo que podamos extender una que termine en una inflación menor.

Estas ideas nos llevan a la siguiente solución en código. De momento no discutiremos los detalles y los veremos con un poco más de profundidad en la parte 4 del libro cuando hablemos de heurísticas para la creación de algoritmos.

```
lista=df_inflacion['Inflation'].to_list()
max_decreciente=[]
padres=[]

for j in range(51):
    mejor=0
    padre=None
    for k in range(j-1):
        if lista[k]>lista[j] and max_decreciente[k]>mejor:
            padre=k
            mejor=max_decreciente[k]
    padres.append(padre)
    max_decreciente.append(mejor+1)

mejor_lista=[]
padre=49
while padre!=None:
    mejor_lista=[padre]+mejor_lista
    padre=padres[mejor_lista[0]]
mejor_lista=[x+1970 for x in mejor_lista]
print(mejor_lista)

df_inflacion.loc[mejor_lista]
```

```
[1973, 1977, 1979, 1989, 1991, 1997, 1999, 2002, 2004, 2006, 2009, 2016, 2019]
```

Year	Inflation
1973	21.37
1977	20.66
1979	20.02
1989	19.69
1991	18.79
1997	15.72
1999	12.32
2002	5.70
2004	5.19
2006	4.05
2009	3.57
2016	3.36
2019	2.83

De este modo, la mejor lista de años en los cuales la inflación ha disminuido tiene longitud \(13\). \(\square\)

Sucesiones recursivas lineales

Las sucesiones recursivas pueden ser tan extrañas como uno quisiera. Por ejemplo, podríamos pedir que $a_0=1, a_1=2, a_2=3$ y que $a_n=a_{n-1}+a_{n-2}^2-\log a_{n-3}$. En general es difícil entender totalmente una sucesión recursiva arbitraria y este tipo de preguntas nos llevan a áreas muy interesantes de las matemáticas como la dinámica y los sistemas caóticos. Por esta razón, es de mucha utilidad clasificar a las recursiones en distintos tipos e identificar aquellos que podamos estudiar de manera sencilla.

Definición. Para un entero k decimos que una sucesión es **recursiva lineal de orden** k si:

- Se dan sus primeros k términos $a_0,\ldots,a_{k-1}$ y
- Satisface una **recursión lineal**, es decir existen k constantes $c_0,\ldots,c_{k-1}$ tales que para todo $n\geq 0$ se cumple que

  $$a_{n+k}=c_0a_{n}+c_1a_{n+1}+\ldots+c_{k-1}a_{n+k-1}.$$

En términos de álgebra lineal, debe existir una k fija y k coeficientes fijos tales que cada término es combinación lineal de los k anteriores con esos coeficientes.

Ejemplo. Las sucesiones recursivas lineales de orden 1 son simplemente las sucesiones geométricas pues se nos da el primer término a_0 y luego la recursión que se cumple es $a_{n+1}=c_0a_n$. Una sencilla prueba inductiva muestra que

$$a_n=a_0\cdot c_0^n.$$

Así, a partir de una fórmula recursiva es muy sencillo dar una fórmula cerrada para ellas. $\square$

Ejemplo. Ya nos hemos encontrado con una sucesión recursiva lineal de orden 2: la sucesión de Fibonacci. Se dan los primeros dos términos como 0 y 1. Luego, cada término es la suma de los dos anteriores, es decir es la combinación lineal de ellos con coeficiente 1.

Otra sucesión recursiva lineal de orden 2 es la sucesión de Lucas. Los primeros dos términos son 2 y 1. Luego, cada término es la suma de los dos anteriores. Los primeros números de la sucesión de Lucas son:

$$2,1,3,4,7,11,18,29,47,76,\ldots.$$

$\square$

Las sucesiones recursivas lineales se conocen muy bien. Un primer resultado del cual no es difícil convencerse mediante un argumento inductivo es el siguiente.

Lema. Si dos sucesiones recursivas lineales de orden k coinciden en sus primeros k elementos y además tienen la misma regla recursiva, entonces coinciden en todos sus elementos.

A veces esto es suficiente para resolver un problema.

Problema. Demuesta la siguiente fórmula para los números de Fibonacci:

$$F_n=\frac{\left(\frac{1+\sqrt{5}}{2}\right)^n-\left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}.$$

Solución. Llamemos f_n al lado derecho de la fórmula por demostrar. Una verificación rápida muestra $f_0=0=F_0$ y que $f_1=1=F_1$. Por el lema anterior, bastaría ver que f_n es recursiva lineal de orden 2, con la misma regla de recursión que la sucesión de Fibonacci.

Escribamos $\varphi=\frac{1+\sqrt{5}}{2}$ y $\varphi'=\frac{1-\sqrt{5}}{2}$. Haciendo las operaciones, se puede verificar que $(x-\varphi)(x-\varphi')=x^2-x-1$, de modo que φ y φ' son las raíces del polinomio de la derecha. Esto nos dice que $\varphi^2=\varphi+1$ y que $\varphi'^2=\varphi'+1$. Al multiplicar por φ^n y φ'^n, respectivamente, obtenemos las igualdades:

$$\begin{aligned} \varphi^{n+2}&=\varphi^{n+1}+\varphi^n,\\ \varphi'^{n+2}&=\varphi'^{n+1}+\varphi'^n.\\ \end{aligned}$$

Restando la segunda a la primera y dividiendo entre $\sqrt{5}$ obtenemos que $f_{n+2}=f_{n+1}+f_n$. Así, la sucesión F_n y la f_n tienen los mismos dos términos y la misma recursión lineal de orden 2. De este modo, por el lema deben coincidir.

\(\square\)

El problema anterior es lindo, pero no explica de dónde sale la fórmula. La clave está en el polinomio \(x^2-x-1\) que se menciona durante el problema. Si tenemos \(a_n\) una sucesión recursiva lineal de orden \(2\) con recursión \(a_{n+2}=c_0a_n+c_1a_{n+1}\), entonces podemos asociarle el **polinomio característico** \(x^2-c_1x-c_0\). Este polinomio está muy conectado con la sucesión. Por ejemplo, puede mostrarse que si tiene dos raíces distintas \(r_1\) y \(r_2\), entonces existen constantes \(s_1\) y \(s_2\) tales que \(a_n=s_1r_1^n+s_2r_2^n\). Estas constantes pueden encontrarse con un sistema de ecuaciones usando los primeros valores de la sucesión. Veamos cómo podemos poner esto en acción.

Problema. Encuentra una fórmula cerrada para la sucesión \(x_n\) tal que \(x_0=3\), \(x_1=5\) y para \(n\geq 0\) cumple que

\[x_{n+2}=2x_{n+1}+3x_n.\]

Solución. Por lo discutido arriba, debemos estudiar el polinomio \(x^2-2x-3\). Sus raíces son \(3\) y \(-1\), que son distintas. Así, deben existir constantes \(a\) y \(b\) tales que \(x_n=a\cdot 3^n + b\cdot (-1)^n\). Los casos \(n=0\) y \(n=1\) nos llevan al sistema de ecuaciones:

\[\begin{split}\begin{cases}a+b&=3\\3a-b&=5. \end{cases}\end{split}\]
La solución es \(a=2\), \(b=1\). De este modo, concluimos que una fórmula cerrada para la sucesión es

\[x_n=2\cdot 3^n + (-1)^n.\]

\(\square\)

Hay que reflexionar un poco para convencerse de que el método anterior es válido matemáticamente. Sin embargo, por el momento podemos calcular computacionalmente los primeros valores de la sucesión y de la fórmula para ir convenciéndonos de su validez.

```
a,b=3,5
valores=[]
for j in range(10):
    valores.append(a)
    a,b=b,2*b+3*a
print("Los primeros 10 valores usando la recursión son {}".format(valores))

valores=[]
for j in range(10):
    valores.append(2*3**j+(-1)**j)
print("Los primeros 10 valores usando la fórmula son {}".format(valores))
```

```
Los primeros 10 valores usando la recursión son [3, 5, 19, 53, 163, 485, 1459, 4373, 13123, 39365]
Los primeros 10 valores usando la fórmula son [3, 5, 19, 53, 163, 485, 1459, 4373, 13123, 39365]
```

Tarea moral

Los siguientes problemas te ayudarán a practicar lo visto en esta entrada. Para resolverlos, necesitarás usar herramientas matemáticas, computacionales o ambas.

1. Se tienen 5 piratas, que van a repartir un tesoro de 1000 monedas entre ellos. Están ordenados por rangos. Repartirán el tesoro haciendo varias rondas de propuesta-votación. Cada una de ellas consiste de lo siguiente:
   - De entre los piratas que queden, el de mayor rango hace una propuesta de cómo repartir las monedas.
   - Se vota la propuesta entre los piratas.
   - Si los votos por aprobarla son estrictamente mayor a la mitad de los piratas restantes, entonces así se hace la repartición.
   - Si la propuesta no se aprueba con más de la mitad de los votos, entonces tiran al pirata que la propuso por la borda.

   La prioridad número uno de un pirata es sobrevivir. La prioridad número dos es conseguir la mayor cantidad posible de monedas. Finalmente, si debe decidir entre dos opciones en sobrevive con la misma cantidad de monedas, entonces elegirá en la que más piratas se avienten por la borda. Los piratas conocen estas reglas y por el honor pirata las seguirán al pie de la letra.

   ¿Qué es lo mejor que le puede pasar a cada pirata? ¿Cómo debe votar? En caso de que le toque proponer, ¿cómo debe proponer? Como sugerencia, estudia primero qué pasa con pocos piratas.

Como sugerencia, estudia primero qué pasa con pocos piratas.

2. Hay algunos problemas que quedaron pendientes en el texto. Están recopilados a continuación. Resuélvelos.
   - Ver que en efecto $y_n=2^n$.
   - Ver que la sucesión de a_n es cíclica.
   - Ver que la sucesión de b_n puede ser tan grande como se quiere. Como sugerencia, pon a b_{n+4} en términos de b_{n},b_{n+1},b_{n+2} para mostrar que $|b_{n+4}|\geq 2|b_n|$.
3. En el problema de la cuenta de palabras que no tenían AB encontramos el valor que nos pidieron. Sin embargo, todo lo obtenido permite encontrar una fórmula cerrada para n letras. Encuéntrala y demuéstrala.
4. Adapta la solución de la lista de años de inflación decreciente para ahora mostrar la lista más larga de años con solución creciente.
5. Lee la siguiente entrada de blog para conocer más de las sucesiones recursivas y de las recursiones lineales: Sucesiones recursivas y recursiones lineales .

‹
Anterior
Principio de inducción

Siguiente
Principio extremo
›

Por Leonardo Ignacio Martínez Sandoval