Práctica 5: Backtracking

Ejercitación básica

Ejercicio 1. Programar en Python las siguientes funciones. Usamos la letra A para denotar listas de elementos:

a) `variaciones_con_repeticion(A, n)`: devuelve la lista de todas las listas de longitud n que se pueden formar usando n elementos de A permitiendo repetidos. Por ejemplo, si los dígitos posibles son $A =$ `[0,1,2,3,4,5,6,7,8,9]` y un cuentakilómetros tiene $n = 3$ dígitos, `variaciones_con_repeticion(A, n)` corresponde a la lista de las mil posibles secuencias de dígitos que puede mostrar el cuentakilómetros:

```
[[0,0,0],[0,0,1],[0,0,2],...,[9,9,9]]
```

Notar que las listas posibles son de la forma $[x_1, \ldots, x_n]$ donde cada x_i es un elemento de A.

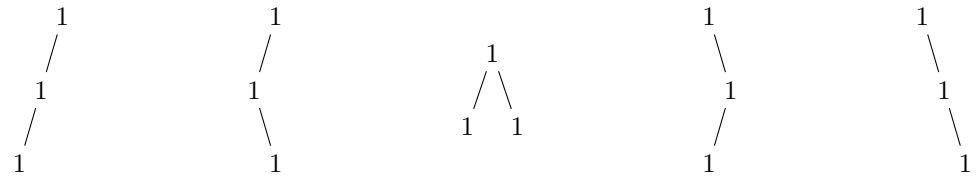
b) `variaciones_sin_repeticion(A, n)`: devuelve la lista de todas las listas de longitud n que se pueden formar usando n elementos de A **sin repetir**. Por ejemplo, si una sala tiene butacas numeradas $A =$ `[1,2,3,4,5,6]` y se sientan $n = 2$ personas en ellas, `variaciones_sin_repeticion(A, n)` corresponde a la lista que incluye todas las posibles maneras en las que esas dos personas pueden sentarse:

```
[[1,2],[1,3],[1,4],[1,5],[1,6],[2,1],[2,3],[2,4],[2,5],[2,6],...]
```

donde cada lista $[x, y]$ significa que la primera persona se sienta en la butaca x y la segunda en la butaca y. Notar que el orden es relevante, es decir, no es lo mismo `[1,2]` que `[2,1]`. Notar que las listas posibles son de la forma $[x_1, \ldots, x_n]$ donde cada x_i es un elemento de A y no hay elementos repetidos.

Ejercicio 2. Diseñar un algoritmo que reciba como entrada una lista A de enteros y un número k, y determine si es posible elegir algunos de los números de A cuya suma dé k. En caso de que sea posible, se debe devolver además una subsecuencia de A que sume k. Por ejemplo, si $A =$ `[8, 11, 11, 8, 12, 7]` y $k = 29$ la respuesta es positiva porque `[11, 11, 7]` es una subsecuencia de A que suma 29.

Ejercicio 3. Programar en Python una función `arbolesBinarios(k)` que dado un número $k \geq 0$ devuelva una lista que contiene a todos los árboles binarios de tamaño k, asumiendo que en todos los nodos se usa el número 1 como valor. Por ejemplo, los posibles árboles binarios de tamaño $k = 3$ son los siguientes cinco:



Ejercitación adicional

Ejercicio 4. Programar en Python una función `listasQueSuman(`k`)` que dado un número $k \geq 0$ devuelva una lista que contiene a todas las posibles listas de elementos estrictamente positivos que suman k. (Es decir, se desea listar todas las maneras de sumar k). Por ejemplo:

k	salida
0	`[[]]`
1	`[[1]]`
2	`[[2], [1, 1]]`
3	`[[3], [2, 1], [1, 2], [1, 1, 1]]`
4	`[[4], [3, 1], [2, 2], [2, 1, 1], [1, 3], [1, 2, 1], [1, 1, 2], [1, 1, 1, 1]]`

Ejercicio 5. Un **cuadrado mágico** de orden n es una matriz de $n \times n$ cuyos elementos son números entre 1 y n sin repetidos, y tales que todas las filas y columnas suman lo mismo. Programar una función que reciba un número n y genere todos los posibles cuadrados mágicos de orden n.

Ejercicio 6. Un mapamundi tiene n países. Se quiere colorear el mapamundi de tal manera que dos países limítrofes **no** estén pintados del mismo color. Diseñar un algoritmo que reciba como entrada una lista con los nombres de los n países $[p_0, p_1, \ldots p_{n-1}]$ y una lista de k posibles colores $[c_0, \ldots, c_{k-1}]$. La salida debe ser un diccionario C que a cada país le asigne uno de los colores, de tal manera que si p y q son países limítrofes entonces $C[p] \neq C[q]$. Se puede asumir dada una función `sonLimítrofes(`p`, `q`)` que recibe los nombres de dos países y devuelve un booleano que indica si son o no son limítrofes.