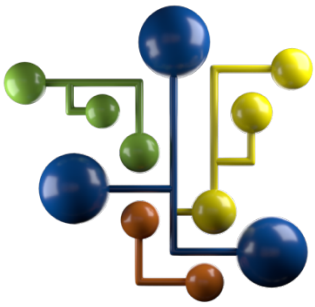

Data Science Academy

www.datascienceacademy.com.br

Introdução à Lógica de Programação

E para concluir a semana 2 do curso de Introdução à Lógica de Programação, mais um importante e útil algoritmo: Backtracking.

O Backtracking é uma forma de recursão. Mas envolve escolher apenas uma opção dentre todas as possibilidades. Começamos escolhendo uma opção e voltamos atrás, se chegarmos a um estado em que concluímos que essa opção específica não fornece a solução necessária. Repetimos essas etapas passando por cada opção disponível até obter a solução desejada.

O Backtracking é uma técnica algorítmica para resolver recursivamente os problemas, tentando criar uma solução de forma incremental, uma peça de cada vez, removendo as soluções que falham em satisfazer as restrições do problema a qualquer momento.

Por exemplo, considere um algoritmo de solução do jogo Sudoku, no qual tentamos preencher os dígitos um por um. Sempre que descobrimos que o dígito atual não pode levar a uma solução, nós o removemos (backtrack) e tentamos o próximo dígito. É melhor do que uma abordagem ingênua (gerando todas as combinações possíveis de dígitos e depois tentando todas as combinações uma por uma), pois ela libera um conjunto de permutações sempre que recua.

Aliás, se nunca jogou Sudoku (imagem abaixo), saiba que é um excelente exercício para o cérebro. Você encontra várias apps para seu Smartphone e se quiser desenvolver a solução do jogo em Python usando Backtracking, aqui tem um exemplo:

https://medium.com/datadriveninvestor/solving-sudoku-in-seconds-or-less-with-python-1c21f10117d6

Abaixo um exemplo de como encontrar toda a ordem possível de arranjos de um determinado conjunto de letras. Quando escolhemos um par, aplicamos Backtracking para verificar se esse par exato já foi criado ou não. Se ainda não tiver sido criado, o par será adicionado à lista de respostas, caso contrário, será ignorado.

```
# Função para o algoritmo Backtracking
def permuta(combinacoes, lista):

--# Verificamos se o número de combinações é igual a 1
--# e retornamos a própria lista, pois não há combinações a fazer
--if combinacoes == 1:
----return lista

--# Se quisermos mais de 1 combinação dos elementos, começamos a recursão
-- else:

----# Usamos list comprehension com recursão (chamada à própria função)
----return [ y + x for y in permuta(1, lista) for x in permuta(combinacoes - 1, lista) ]
```

```
# Executa a função buscando diferentes combinações e aplicando a técnica de Backtracking
print(permuta(1, ["a","b","c"]))
print(permuta(2, ["a","b","c"]))
print(permuta(3, ["a","b","c"]))
```

Bom final de semana para você e na próxima semana tem mais!

#aula10