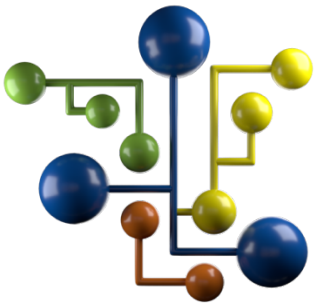

Data Science Academy

www.datascienceacademy.com.br

Introdução à Lógica de Programação

Vamos para a terceira e última semana do curso de Introdução à Lógica de Programação.

E para começar bem a semana, aqui vai uma dica de ouro! Um repositório completo com a implementação de diversos algoritmos em 6 linguagens de programação: C, C++, Java, JavaScript, Go e Python. Um trabalho realmente magnífico:

https://github.com/TheAlgorithms

Aula 11 – Algoritmos Gulosos (Greedy Algorithms)

Suponha que você tenha uma sala de aula e queira reservar o máximo de aulas possível nela. Assim, recebe-se uma lista das aulas.

Você não pode reservar todas as aulas na sala porque os horários de algumas delas coincidem.

Soa como um problema difícil, não? Na realidade, o algoritmo é tão simples que pode surpreender. Aqui temos o funcionamento dele:

1. Pegue a aula que termina mais cedo. Esta é a primeira aula que você colocará nessa sala.

2. Agora você precisa pegar uma aula que comece depois da primeira aula. De novo, pegue a aula que termine mais cedo. Esta é a segunda aula que você colocará na lista.

3. Continue fazendo isso e no final você terá a sua resposta!

Muitas pessoas dizem que esse algoritmo parece ser fácil. Mas ele é óbvio demais, logo, deve estar errado. No entanto essa é a beleza dos algoritmos gulosos (também chamados de algoritmos gananciosos): eles são fáceis!

Um algoritmo guloso é simples: a cada etapa, deve-se escolher o movimento ideal. Nesse caso, cada vez que você escolhe uma aula, deve escolher a que acaba mais cedo. Em termos técnicos: a cada etapa, escolhe-se a solução ideal, e no fim você tem uma solução global ideal. Acredite ou não, esse algoritmo simples acha a solução ideal para esse problema! Obviamente os algoritmos gulosos nem sempre funcionam, mas eles são simples de escrever!

Existem dúzias de algoritmos gulosos e aqui você encontra uma lista bastante completa:

https://www.geeksforgeeks.org/greedy-algorithms/

Alguns desses algoritmos são estudados nos cursos Análise em Grafos Para Big Data e Deep Learning II da Formação IA e no curso de Programação Para Blockchain da Formação Engenheiro Blockchain!

Por que os algoritmos gulosos são chamados de gulosos?

Chamamos algoritmos de "gulosos" ou "gananciosos" quando eles utilizam a propriedade “gananciosa”. A propriedade gananciosa é:

Neste exato momento, qual é a melhor escolha a ser feita?

Algoritmos gananciosos são..... gananciosos. Eles não olham para o futuro para decidir a solução ideal global. Eles estão preocupados apenas com a solução ideal localmente. Isso significa que a solução ideal geral pode diferir da solução escolhida pelo algoritmo.

Eles nunca olham para trás para o que fizeram para ver se poderiam otimizar globalmente. Essa é a principal diferença entre Algoritmos Gulosos e Programação Dinâmica (assunto da Aula 12).

Para ser mais claro, uma das perguntas mais pesquisadas no Google sobre algoritmos gananciosos é:

"Quais estratégias de solução de problemas não garantem soluções, mas fazem uso eficiente do tempo?"

A resposta é "algoritmos gananciosos". Eles não garantem soluções, mas são muito eficientes em termos de tempo.

Algoritmos gananciosos são rápidos. E eles são usados por isso, porque são rápidos.

Os algoritmos gulosos são amplamente empregados em Inteligência Artificial, especialmente em Reinforcement Learning, para descobrir o caminho mais rápido para uma solução e ajudar o agente inteligente a tomar uma decisão. E podemos usar Deep Learning com algoritmos gulosos para que o aprendizado do algoritmo ocorra de forma automática. Mostramos como fazer isso no curso de Inteligência Artificial Aplicada a Finanças, da Formação Engenheiro Blockchain e no Laboratório de Machine Learning no curso de Matemática Para Machine Learning da Formação Análise Estatística Para Cientistas de dados.

Vamos a um exemplo em Python.

Imagine que você esteja precisando fazer uma dieta e consumir apenas 1000 calorias por dia. Mas ao invés de procurar um Nutricionista especializado, você foi perguntar a um amigo que “aprendeu Nutrição no Facebook” quais alimentos você deveria comer e o “especialista” indicou os 10 alimentos abaixo, com os respectivos valores e calorias:

Alimentos = ['Frango', 'Milk Shake', 'Pizza', 'Hamburger', 'Batata Frita', 'Refrigerante', 'Maça', 'Laranja', 'Cenoura', "Alface"]

Valores (em Reais) = [79, 18, 45, 38, 25, 9, 15, 10, 22, 12]

Calorias = [114, 156, 359, 354, 365, 153, 97, 82, 79, 40]

Desconfiado que alguns desses elementos não o ajudarão na dieta, você então cria um algoritmo para buscar as melhores combinações de cardápio considerando duas regras (constraints): valor do cardápio para 1000 calorias e custo do cardápio para 1000 calorias.

Um algoritmo guloso (e eu juro que não foi trocadilho com o exemplo...kkkk) é então usado para esta tarefa. Aqui a solução em Python (cuidado com a identação e leia atentamente os comentários):

```
# Algoritmo guloso que busca a melhor combinação de alimentos para uma dieta de 1000 calorias.

# Classe para armazenar os alimentos
class Alimentos(object):

--# Construtor da classe
--def __init__(self, n, v, c):

----# Nome do alimento
----self.nome = n

----# Valor do alimento em Reais
----self.valor = v

----# Número de calorias do alimento
----self.calorias = c

--# Método para obter o valor de cada alimento
--def getValor(self):
----return self.valor

--# Método para obter o custo (1 dividido pelo número de calorias, calculado mais abaixo. Aqui retornamos apenas as calorias)
--def getCusto(self):
----return self.calorias

--# Método para imprimir nome/valor/calorias
--def __str__(self):
----return self.nome + ': <' + str(self.valor) + ', ' + str(self.calorias) + '>'
```

```
# Cria o Menu com a lista de alimentos"
def criaMenu(nomes, valores, calorias):

--# Cria a lista vazia
--menu = []

--# De acordo com o tamanho da lista de valores, inserimos todos os itens na lista de menu
--for i in range(len(valores)):
----menu.append(Alimentos(nomes[i], valores[i], calorias[i]))
--return menu


# Algoritmo Guloso
def greedy(items, maxCost, keyFunction):

--# Copia todos os itens
--itemsCopy = sorted(items, key = keyFunction, reverse = True)

--# Lista de resultados
--resultado = []

--# Valor total
--totalValor = 0

--# Custo total
--totalCusto = 0.0

--# De acordo com o tamanho da lista de itens, realiza os seguintes cálculos
--for i in range(len(itemsCopy)):
----if (totalCusto + itemsCopy[i].getCusto()) < = maxCost:

------# adiciona os itens a lista de resultados
------resultado.append(itemsCopy[i])

------# Calcula o custo total
------totalCusto += itemsCopy[i].getCusto()

------# Calcula o valor total
------totalValor += itemsCopy[i].getValor()

--return (resultado, totalValor)

# Função para executar o algoritmo guloso
```

```
def executaGreedy(items, constraint, keyFunction):
--result, val = greedy(items, constraint, keyFunction)
--print('Valor Total dos Itens para 1000 calorias =', val)
--for item in result:
----print(item)

# Função que gera o melhor cardápio pelo valor individual dos alimentos e pelo custo dos alimentos
def geraCardapio(alimentos, totCalorias):

--print('Usando o algoritmo guloso para buscar o melhor menu pelo valor dos alimentos para', totCalorias, 'calorias')
--executaGreedy(alimentos, totCalorias, Alimentos.getValor)

--print('\nUsando o algoritmo guloso para buscar o melhor menu pelo custo dos alimentos para', totCalorias, 'calorias')
--executaGreedy(alimentos, totCalorias, lambda x: 1/Alimentos.getCusto(x))


# Listas para testar o algoritmo
listaAlimentos = ['Frango', 'Milk Shake', 'Pizza', 'Hamburger', 'Batata Frita', 'Refrigerante', 'Maça', 'Laranja', 'Cenoura', "Alface"]

valores = [79, 18, 45, 38, 25, 9, 15, 10, 22, 12]

calorias = [114, 156, 359, 354, 365, 153, 97, 82, 79, 40]


# Cria o menu
menu_alimentos = criaMenu(listaAlimentos, valores, calorias)


# Busca o melhor menu para o consumo de 1000 calorias
geraCardapio(menu_alimentos, 1000)
```

Ao executar o script acima no Jupyter Notebook, teremos as seguintes saídas:

```
Usando o algoritmo guloso para buscar o melhor menu pelo valor dos alimentos para 1000 calorias
Valor Total dos Itens para 1000 calorias = 196
Frango: < 79, 114 >
Pizza: < 45, 359 >
Hamburger: < 38, 354 >
```

Cenoura: < 22, 79 >
Alface: < 12, 40 >

Usando o algoritmo guloso para buscar o melhor menu pelo custo dos alimentos para 1000 calorias
Valor Total dos Itens para 1000 calorias = 165
Alface: < 12, 40 >
Cenoura: < 22, 79 >
Laranja: < 10, 82 >
Maça: < 15, 97 >
Frango: < 79, 114 >
Refrigerante: < 9, 153 >
Milk Shake: < 18, 156 >

Observe que o algoritmo foi capaz de encontrar as soluções com base nas regras que foram apresentadas. Isso não significa que ele tenha encontrado as melhores soluções possíveis, mas ele apresentou as soluções rapidamente e em muitos casos, isso é tudo que queremos ou precisamos!

E se você observar ainda mais atentamente, vai perceber que a segunda opção é bem melhor que a primeira quando consideramos as duas regras apresentadas. Mas na prática, temos duas soluções!

No exemplo acima o algoritmo não sabe qual é a melhor solução e precisamos de seres humanos para a interpretação. Mas e se removermos os seres humanos da equação?

Os algoritmos gulosos quando associados a Deep Learning oferecem uma solução bem interessante de Inteligência Artificial, pois o agente inteligente aprende a melhor solução ao ser exposto a diversas combinações possíveis e ao ser recompensado ou penalizado pelas soluções certas ou erradas! Esse tipo de aprendizagem (Reinforcement Learning ou Aprendizagem Por Reforço) vem sendo usado com sucesso em Finanças, Medicina e Robótica.

Uma alternativa aos algoritmos gulosos é a programação dinâmica, tema da Aula 12.

#aula11