LAS VEGAS

ALGORITMOS PROBABILÍSTICOS

RELEMBRANDO O ALGORITIMO MONTECARLO

- Os algoritmos Monte Carlo tem como principal característica executarem funções em tempo determinístico mas dão uma resposta probabilística isto significa que o tempo entre diferentes execuções nunca se altera mas existe a possibilidade da resposta estar incorreta.
- Considerando o problema ao lado: Dado um vetor com números inteiros o índice da posição que contém um valor específico podemos construir um algoritmo Monte Carlo para este programa como segue a seguir:

- def Monte_Carlo(v,x)
  - I = randint(0,len(v))
  - return I;

COMO FUNCIONA O ALGORITMO LAS VEGAS

- Agora se pegarmos o algoritmo anterior e retornarmos a nossa resposta única e exclusivamente se a mesma estiver correta temos um algoritmo de Las Vegas a diferença do algoritmo Las Vegas para o de Monte Carlo é que o algoritmo Las Vegas somente nos retorna nos retorna uma resposta verdadeira

- Def Las_Vegas (v,x)
  - while true:
    - I = ranint(0,len(v))
    - If v[i] == x;

      return i;

LAS VEGAS

- O algoritmo Las Vegas apenas retorna uma resposta quando encontra uma solução correta e portanto nunca nos trará uma resposta incorreta de a probabilidade de erro e deixando esta probabilidade exclusiva para inversão de bit não detectada.
- porém o tempo de execução é totalmente incerto o algoritmo pode ter sorte de encontrar a solução correta já na primeira tentativa ou tentar múltiplas vezes até encontrar a solução
- Um algoritmo Las Vegas é um complemento do algoritmo de Monte Carlo
- O algoritmo Las Vegas tem tempo de execução probabilístico mas a sua resposta é determinística

TODO ALGORITMO MONTE CARLO COM UMA ETAPA DE VERIFICAÇÃO É UM ALGORITMO LAS VEGAS?

- Nem todo o algoritmo Las Vegas é um método de Monte Carlo com uma etapa de verificação.
- O algoritmo Las Vegas mais utilizado é o Algoritmo de ordenação Quick Sort.
- Este escolhe um elemento aleatório no vetor (elemento Pivo), e separa o vetor entre os elementos menores e maiores que o pivô, chamando o algoritmos recursivamente em cada metade.
- Seu pivô escolhido dividido de forma excessivamente desbalanceado vetor o algoritmo executa em O(n^2).

O QUICK SORT É LAS VEGAS?

- No entanto para vetores grandes é esperado que é na média para escolha aleatória acabe selecionando pivôs relativamente bons sendo assim o tempo médio de execução do algoritmo é dado por O(n log(n)).
- Detalhe bem importante neste algoritmo que não existe uma verificação resultado resultado mas a característica final é de um algoritmo de Las Vegas pois o vetor resultante sempre estará ordenado porém o tempo de execução é totalmente incerto pois varia a cada execução mesmo que tenhamos a mesma entrada.

QUALQUER ALGORITMO DE ORDENAÇÃO É UM ALGORITMO LAS VEGAS.

- Quando os algoritmos de ordenação são apresentados para os estudantes da computação muitas vezes temos o algoritmo chamado "Bozo Sort" Com o vetor de entrada, gera-se uma pergunta aleatória dos elementos e verifica se o vetor resultante está ordenado se não estiver repete-se o processo até que o mesmo esteja ordenado .
- Ao contrário do Quick Sort, que é um algoritmo eficiente na pratica de ordenação mesmo, considerando o pior caso. O Bozo Sort é extremamente ineficiente. Ainda assim o mesmo é um algoritmo Las Vegas, pois só temos o retorno com a solução correta.

BUSCA DE CAMINHOS VÁLIDOS APLICAÇÃO DE LAS VEGAS

- Imagine que você quer encontrar um caminho entre dois vértices em um grafo.
- Um algoritmo Las Vegas poderia:
- Escolher caminhos aleatórios.
- Verificar se o caminho é válido (conecta os vértices sem repetir arestas ou vértices, conforme o problema).
- Retornar **somente** quando encontrar um caminho válido.

```
def caminho_las_vegas(grafo, origem, destino):
    while True:
        caminho = gerar_caminho_aleatorio(grafo, origem, destino)
        if verificar_caminho_valido(caminho):
            return caminho
```

COLORAÇÃO DE GRAFOS

- Problema: colorir os vértices de um grafo com o menor número de cores, sem que vértices adjacentes tenham a mesma cor.
- O algoritmo tenta atribuições aleatórias de cores.
- Verifica se a coloração é válida.
- Só retorna quando encontra uma coloração correta.

PROBLEMA DO CLIQUE MÁXIMO

- Um clique é um subconjunto de vértices onde todos estão conectados entre si.
- O algoritmo pode gerar subconjuntos aleatórios.
- Verifica se formam um clique.
- Retorna apenas quando encontra um clique válido (e possivelmente o maior encontrado até então).

AS OITO RAINHAS

- Em um tabuleiro NxN, desejamos colocar N rainhas sem que as mesmas se ataquem
- Dada uma configuração com N rainhas, é simples verificar se ela é valida, mas encontrar uma solução não é trivial.
- O algoritmo eficiente coloca em rainhas em posições aleatórias e verifica se esta é uma solução.
- Porém podemos tornar essa solução mais eficiente considerando que 2 rainhas nunca podem ocupar a mesma linha ou coluna assim podemos posicionar as rainhas uma por uma.
- Uma por coluna escolhendo uma linha aleatória para cada rainha colocada verificamos então se esta nova está sobre ataque das anteriores se estiver retornamos ao início e testamos novamente.
- https://www.brainmetrix.com/8-queens/