MINISTÉRIO DA EDUCAÇÃO

UNIVERSIDADE FEDERAL RURAL

DE PERNAMBUCO

UNIDADE ACADÊMICA DE GARANHUNS

Projeto e Análise de Algoritmos - 2018.2

"Problema bitônico do caixeiro viajante."

Discente: Antônio Adelino e Armstrong Lohãns

Docente: Tiago B. A. de Carvalho

Garanhuns – PE

Estratégia Gulosa:

Nosso algoritmo guloso funciona da seguinte maneira, a priori, ele tem como parâmetro único um objeto do tipo Matriz, a qual representa um plano contendo pontos nela, a partir disso é chamado uma função auxiliar "*resgatarPontos*" que recebe a dada matriz, a qual tem como retorno uma lista contendo apenas os pontos da mesma.

Após isso, é chamada uma outra função auxiliar, essa função recebe a lista dos pontos e a ordena, levando em consideração a posição da coluna de cada ponto em ordem crescente. Com isso, temos uma lista de pontos ordenada pelo número da sua coluna, ou seja, temos os pontos ordenados da esquerda da matriz para sua direita. Fazemos a cópia de tal lista para outra e em seguida adicionamos o primeiro ponto a uma lista auxiliar, pois, pela ordenação, ele é o ponto mais a esquerda.

Logo depois, iniciamos um laço controlado numericamente, tal estrutura varre a lista de pontos auxiliar dois a dois, dentro desse laço existe uma condição que faz uma escolha gulosa, uma vez que ele analisa a seguinte questão "Dado que eu estou na posição 1, o ponto que está na posição 3 está na linha mais acima, ou na mesma linha, em comparação ao ponto que está na posição 2", se essa condição for verdadeira, o algoritmo adiciona o ponto que está na posição 3 em uma lista auxiliar e o remove da lista de pontos auxiliar, se a condição não for satisfeita, o algoritmo adiciona o ponto que está na posição 2 na lista auxiliar, e o remove da lista de pontos auxiliar.

Agora, com o fim desse laço, temos duas listas, uma com os pontos que foram removidos e outra com os pontos ainda remanescentes, após isso invertemos a lista auxiliar e fazemos mais um laço de repetição, o qual irá percorrer a lista auxiliar adicionando cada elemento dele na lista de pontos auxiliar.

Logo, temos o resultado final do algoritmo dado como saída a lista de pontos auxiliares e seu tempo de execução, o qual retornamos ele na função.

Estratégia Backtracking:

Nosso algoritmo de resolução do problema dado usando como parâmetro a estratégia Backtracking funciona da seguinte maneira, primeiramente recebemos uma dada matriz, a qual, assim como a estratégia gulosa descrita anteriormente, representa um plano contendo pontos nela. A partir disso, criamos a lista que representa o caminho que deve ser seguido entre os pontos e adicionamos o ponto posicionado mais à esquerda como primeiro elemento dessa lista, além de marcarmos ele como verificado.

Em seguida, entramos em um laço de repetição controlado por duas condições, uma delas diz que enquanto o tamanho da lista caminho for diferente da lista de pontos deve continuar o laço. A outra condição diz que a variável *k,* a qual representa o marcador que varre a lista de pontos, deve ser menor que a própria lista de pontos.

Dadas e verificadas as condições anteriores, entramos no laço, a priori adicionamos o ponto que está na lista de pontos a lista de caminho, considerando que essa é a melhor escolha, em seguida avaliamos o seguinte, "O ponto que está na posição anterior da lista de caminhos está posicionado mais abaixo do que o ponto recém adicionado? Além disso, o ponto anterior não foi verificado? ".

Considerando como verdadeira a resposta para as questões anteriores, o algoritmo marca o ponto recém adicionado como verificado e remove da lista de caminho o ponto antecessor ao recém adicionado. Caso uma das questoẽs implique em uma não verdade, o algoritmo apenas incrementa o contador *k*.

Ao fim do laço, temos a lista de caminho contendo todos os pontos mais superiores da esquerda para a direita, levando em consideração a estratégia backtracking. Após isso, o algoritmo, por meio de uma função, elimina os pontos repetidos na lista de pontos e na lista de caminhos e os adiciona de trás para frente um a um na lista caminho.

Por fim, temos que a lista Caminho contém a solução do problema dado, solução a qual foi obtida por meio da estratégia backtracking, adicionando o caminho e considerando ele como válido até que seja provado a infelicidade na escolha.

Por fim, temos o resultado final dado como saída a lista do caminho escolhido e seu tempo de execução, o qual retornamos ele na função.

Dificuldades encontradas:

Nossa dificuldade foi ao buscar conteúdo relacionado ao percurso bitônico, já que ele é uma adaptação em tempo polinomial do problema original do caixeiro viajante, além de também não conseguirmos fazer uma adaptação deste mesmo algoritmo de forma dinâmica, o qual utilizamos o método backtracking como substituição, no geral, não tivemos maiores problemas.