UFV - Universidade Federal de Viçosa
DPI - Departamento de Informática
Prof. André Gustavo dos Santos
INF 230 - Matemática Discreta - 2021/1

Trabalho 1
Para segunda, 18/out/21

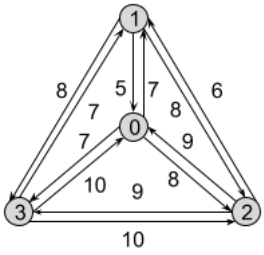
Permutações e combinações

A seção 6.6 do livro texto apresenta algoritmos para gerar permutações e combinações numa certa ordem. Use essas ideias ou outros algoritmos (não precisa gerá-las em ordem) para resolver os seguintes problemas.

1. TSP - Travelling Salesman Problem

Problema do Caixeiro Viajante (PCV), do inglês *Travelling Salesman Problem (TSP)*, é um problema que busca determinar a menor rota para percorrer uma série de cidades (visitando uma única vez cada uma delas), retornando à cidade de origem. É um problema de otimização combinatória bastante importante na ciência da computação teórica, classificado como $\mathcal{NP}$-difícil, o que indica que não se conhece algoritmo eficiente, que o resolva em tempo polinomial em relação ao tamanho da entrada. Também tem grande importância prática, pois o problema e suas variações ocorre em muitas situações do cotidiano.

O nome do problema é inspirado em uma das aplicações originais: o caixeiro, ou mercador ambulante, deve visitar um conjunto de clientes para vender ou entregar produtos, de porta em porta. Ele quer descobrir a menor rota que visita todos os clientes. Considere o exemplo ao lado, com apenas 3 cidades $\{1, 2, 3\}$ além da cidade do caixeiro, 0, onde a rota inicia e termina. A solução para esse exemplo é 28, obtida com a rota $0 \to 3 \to 2 \to 1 \to 0$.



Uma forma de resolver o problema é gerar todas as permutações das cidades, calcular a distância[1] total da rota correspondente a cada permutação e retornar o valor da menor delas. Esta forma não é eficiente pois requer avaliar $n!$ rotas. No entanto, será a forma usada neste trabalho de implementação.

- Implemente um código que gera todas as permutações dos números de 1 a n
- Use esse código em um programa que lê uma matriz de distâncias entre os pontos 0 a n, calcula a distância da rota de cada permutação e informa a menor distância (e uma rota com esta distância)
- Use o programa com as entradas fornecidas no Classroom
- Considere que o caixeiro mora em Viçosa e deve visitar as capitais do Sudeste e do Sul; encontre a solução para este caso (*use algum ambiente, como OpenStreetMap ou GoogleMaps, para criar uma entrada com as distâncias entre as cidades*)
- Verifique se é possível fazer o mesmo com as capitais do Sudeste, do Sul e do Centro-Oeste
- Faça uma análise do tempo gasto pelo seu programa para vários valores de n e informe até que valor o algoritmo encontra a solução em um tempo razoável (por exemplo, 10 minutos ou 1 hora)

2. OP - Orienteering Problem

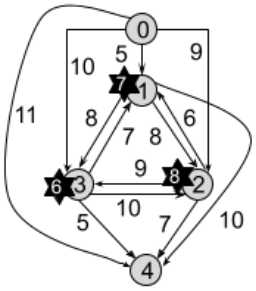
Orienteering Problem (OP) é um problema baseado no esporte de mesmo nome, *Orienteering*, em que cada participante recebe um mapa no ponto inicial do jogo, com uma série de pontos de controle que podem ser visitados apenas uma vez, cada um com um valor (*score*) associado, e um ponto final, em

[1] Os valores podem representar distância (rota mais curta), tempo (rota mais rápida), custo (rota mais barata), entre outros.

que deve chegar dentro de um tempo limite estabelecido. Vence quem chegar ao ponto final dentro do tempo limite com o maior *score* acumulado de pontos de controle visitados. Diferentemente do PCV, a rota não é circular e não necessariamente visita todos os pontos. O objetivo não é encontrar a menor rota, mas a rota com *score* máximo.

O problema também é classificado como NP-difícil, sem algoritmo eficiente conhecido. Como aplicações práticas podemos citar: um caixeiro que não tem tempo suficiente para visitar todos os clientes, que deve então escolher os clientes que darão maior lucro; um turista que tem tempo limitado em uma cidade e quer escolher quais, dentre os vários pontos de interesse, visitar durante o tempo que tem.

Considere o exemplo ao lado, com ponto de partida 0 e chegada 4, em que há 3 pontos de interesse. Os valores nas estrelas junto aos pontos indicam seu *score* e nas ligações entre eles o tempo gasto para ir diretamente de um para outro. Considere ainda um tempo limite $T = 19$. A rota mais rápida que passa por todos os pontos é $0 \to 1 \to 2 \to 3 \to 4$, de tempo 25, maior que o limite, portanto não é possível obter *score* total. A solução para esse exemplo tem *score* 13 e consiste em se visitar os pontos de controle 1 e 3, seguindo a rota $0 \to 1 \to 3 \to 4$, de tempo 18. Note que visitar os pontos 1 e 2 produz maior *score*, mas não é possível visitá-los em tempo menor que o limite.



O problema pode ser resolvido gerando-se combinações e permutações. Para n pontos de interesse, todos os subconjuntos de 0 a n pontos são gerados, ou seja, todas as r-combinações de $r = 0, 1, \ldots, n$ pontos. Para cada uma delas, calcula-se a rota mais rápida que visita todos os pontos selecionados. O cálculo da rota pode ser feito gerando-se todas as permutações destes pontos, como feito no TSP. Ao final, retornar o *score* máximo que pode ser obtido dentro do tempo limite, ou seja, a combinação de pontos visitados e a permutação (ordem em que devem ser visitados) que maximiza o *score* total possível no jogo.

- Implemente um código que gera todas as r-combinações dos números de 1 a n
- Use esse código para gerar todos os subconjuntos dos números de 1 a n (r-combinações, $r = 0 \ldots n$)
- Use o código do problema anterior para gerar todas as permutações de cada um dos subconjuntos
- Avalie todas as permutações de todos as combinações, verifique quais rotas correspondentes possuem tempo total dentro do tempo limite estipulado, e destas retorne a de maior *score* total
- Use o programa com as entradas fornecidas no Classroom
- Escolha alguns pontos de interesse na sua cidade, dando um valor para cada um deles, bem como uma origem e um destino (por exemplo a rodoviária); encontre a solução para algum tempo limite apropriado, por exemplo, algumas horas (*use algum ambiente, como OpenStreetMap ou GoogleMaps, para criar uma entrada com os tempos de percurso a pé entre os pontos*)
- Faça uma análise do tempo gasto pelo seu programa para vários valores de n e limites de tempo T e informe até que valor o algoritmo encontra a solução em um tempo razoável

Exemplo de aplicação na pandemia de COVID-19

- TSP: a prefeitura de uma cidade recebeu um novo carregamento de vacinas e quer abastecer todos os postos de vacinação da cidade no menor tempo possível.
- OP: a prefeitura recebeu um estoque de vacinas no limite do prazo de validade e tem 1 hora para entregá-las aos postos de vacinação; para cada posto é conhecida a demanda diária por vacinas; determinar os postos que devem ser visitados para maximizar o total estimado de vacinas usadas dentro do prazo.