MAC0345 - Desafios 2
Editorial Lista 2

Enrique Junchaya

March 29, 2023

A. AlgoRace.
Tópicos: Grafos, DP

Podemos computar as respostas para todas as consultas possíveis com a seguinte recorrência de PD:

$dp[c][i][j]$ = custo mínimo para viajar da cidade i até a cidade j usando no máximo c trocas de carro

Podemos preencher a matriz dessa PD de forma similar ao algoritmo de Floyd-Warshall:

```
for(int k = 0; k < n; k++)
        for(int i = 0; i < n; i++)
                for(int j = 0; j < n; j++)
                        dp[c][i][j] = min(dp[c][i][j], dp[c-1][i][k] + dp[0][k][j]);
```

Assim, faltaria só computar $dp[0][i][j]$, o melhor caminho de i até j usando um único carro. Isso é fácil de achar se computamos todos os caminhos mínimos usando um único carro e, para cada par de cidades, vemos qual carro é o que minimiza o caminho entre elas.

B. Greg and Graph.
Tópicos: Grafos, DP

Às vezes é mais simples pensar o problema na ordem contrária. Essa é uma dessas vezes. Assim, na ordem contrária, o problema pede para computarmos a soma das distâncias dos caminho mínimos entre cada par de vértices após adicionar um novo vértice (e as arestas que saem/entram nele). Para isso, devemos calcular os caminhos mínimos que usam apenas os vértices adicionados enquanto vamos adicionando novos vértices.

Parece familiar? Sim, é o algoritmo de Floyd-Warshall! Lembra que esse algoritmo é uma DP que, na i-ésima iteração, calcula os caminhos mínimos entre todos os pares de vértices usando apenas os i primeiros vértices. Assim, basta apenas modificar a ordem dos vértices adicionados para que seja igual à do *input*.

C. Arbitrage.
Tópicos: Grafos

Seja c_e o custo de uma aresta e. Se convertemos o custo de toda aresta e em $\log c_e$, então o custo de um caminho seria igual ao logaritmo da produtório dos custos originais das arestas do caminho. Note que o custo do caminho é positivo se e somente se o produtorio dos custos originais é maior que 1. Portanto, negativando os custos das arestas, o problema se reduziria a achar ciclos negativos.

Resumindo, após converter o custo c_e de uma aresta e em $-\log c_e$, o problema se reduz a achar um circuito negativo nesse novo grafo. Para isso, é possível usar o algoritmo de Bellman-Ford ou o de Floyd-Warshall.

D. Short in Average.
Tópicos: Grafos, busca binária

Note que, no problema, se fala de *walks* (passeios) e não de *paths* (caminhos). Em outras palavras, o "caminho" pode repetir arestas.

Quando temos problemas de otimização cuja solução direta parece muito difícil, vale a pena pensar no problema de decisão associado a ele. O motivo disso é que depois podemos tentar fazer uma busca binária. Assim, pensemos no seguinte problema de decisão: existe um passeio cujo comprimento médio é menor que X?

Seja P um passeio e $c(e)$ o custo de uma aresta e, podemos escrever matematicamente o enunciado do problema de decisão anterior:

$$\exists P \text{ tal que } \frac{\sum_{e \in P} c(e)}{|P|} < X.$$

Ou, de outra forma,

$$\exists P \text{ tal que } \sum_{e \in P} c(e) - |P|X < 0.$$

Lembrando que o somatório tem exatamente $|P|$ parcelas, podemos subtrair X de cada uma das parcelas e obter o mesmo resultado

$$\exists P \text{ tal que } \sum_{e \in P} (c(e) - X) < 0.$$

Com isso, obtemos que a média dos custos do caminho não depende mais da quantidade de arestas no caminho. Para resolver o problema de decisão, basta criar um grafo auxiliar em que o custo de cada aresta e seja agora $c(e) - X$ e depois verificar se o caminho mínimo entre os vértices dados é negativo (lembre que, se é possível chegar do vértice inicial a um circuito negativo e de aí ao vértice final, então o custo do caminho mínimo é negativo, mas essa não é a única forma em que o caminho mínimo tenha custo negativo).

Agora, note que, se existe um passeio cujo comprimento médio é menor que X, com certeza existe um passeio cujo comprimento médio é menor que $X + \epsilon$. Assim, podemos fazer uma busca binária para encontrar o menor X que cumpra a condição do problema de decisão.

E. Shortest Routes II.
Tópicos: Grafos

Aplicação direta de Floyd-Warshall. Depois de computar todos os pares de caminhos mínimos com Floyd-Warshall em $O(n^3)$, podemos responder cada consulta em $O(1)$. É necessário tomar cuidado com *corner cases*: arestas múltiplas, laços, etc.

F. Pentagon.
Tópicos: Grafos, DP, Combinatória, Matrizes

Nota: onde está escrito "caminho" ou "circuito" é pra ser caminho simples e circuito simples. Ou seja, caminhos/circuitos em que nenhum vértice é visitado mais de uma vez.

O problema pede para contar o número de circuitos distintos de tamanho 5. Seja $D1$ o número de caminhos de tamanho 1. Obtemos $D1$ diretamente do *input*. Sejam $D2, D3$ definidos analogamente. Podemos calcular $D2$ com uma PD similar à do Floyd-Warshall:

$$D2 = D1[i][k] * D1[k][j], \forall k \in [1, n]$$

Contudo, perceba que não podemos computar $D3$ corretamente fazendo apenas

$$D3 = D2[i][k] * D1[k][j], \forall k \in [1, n]$$

pois estamos contando também os caminhos não simples da forma $i \to j \to k \to j$. Assim, basta desconsiderar esses caminhos na contagem:

$$D3 = (D2[i][k] - D1[i][j] * D1[j][k]) * D1[k][j], \forall k \in [1, n].$$

Para calcular a resposta, podemos fixar o primeiro e o terceiro vértices do circuito ($i \to a \to j \to b \to c \to i$) para poder usar $D2$ e $D3$. Contudo, de forma similar ao caso anterior, podemos estar contando circuitos não simples como $i \to a \to j \to a \to c \to i$ ou $i \to a \to j \to b \to a \to i$. De fato, esses são os únicos casos em que estamos contando mais do que deveríamos, mas devemos tomar cuidado ao desconsiderar esses casos porque poderíamos acabar tirando mais casos dos que deveríamos da contagem.

Finalmente, perceba que vamos a contar cada circuito uma vez por cada vértice que aparece nele (ou seja, 5 vezes). Inclusive, por cada vértice do circuito, estaríamos contando duas vezes o circuito, já que tem duas formas distintas de percorrer ele. Assim, o resultado final de nossa contagem deve ser dividido entre 10.