Trabalho Prático 3
Sistema de cobrança de metrô

Fernanda Carolina da Silva Pereira
Universidade Federal de Minas Gerais
Belo Horizonte – MG – Brasil
fernanda.pereira@dcc.ufmg.br

1. Introdução

João da Silva é um jovem universitário que gosta de conhecer outras cidades ao redor do mundo. Antes de agendar suas viagens ele planeja todos os custos de passagens, hospedagem, alimentação e transporte, pois geralmente seu orçamento é bem apertado, e também planeja os pontos turísticos que deseja conhecer.

O próximo destino de João é Nova York, mas infelizmente devido a recente valorização do dólar, ele terá de economizar no valor que havia planejado para o transporte. Para isso, ele resolveu utilizar o metrô da Big Apple, que recentemente passou por uma revisão no formato de cobrança dos bilhetes. Como o ponto de destino pode estar a várias estações de distância, João poderá precisar pegar vários metrôs subsequentemente.

2. Sistema de Cobrança

O novo sistema de cobrança dos bilhetes, que está em fase de teste de aceitação, funciona da seguinte forma: o passageiro paga por cada transição de linha de metrô realizada (definida como escala) e o bilhete para cada escala pode ter preço diferente. Foi implantada uma política de descontos cumulativos (Di, desconto cumulativo para i-ésima escala em diante) nas escalas realizadas dentro de um intervalo de T minutos, até um limite máximo de D escalas nesse intervalo. Transcorridos os T minutos desde o início da primeira escala (tempo >= T), o passageiro perde o desconto acumulado e inicia novamente a progressão de descontos na próxima escala que fizer. Caso dentro do intervalo T, o passageiro realize mais do que D escalas, a partir da escala D+1, o passageiro não faz mais jus ao desconto cumulativo, pagando então o preço cheio do bilhete ate o termino do tempo T.

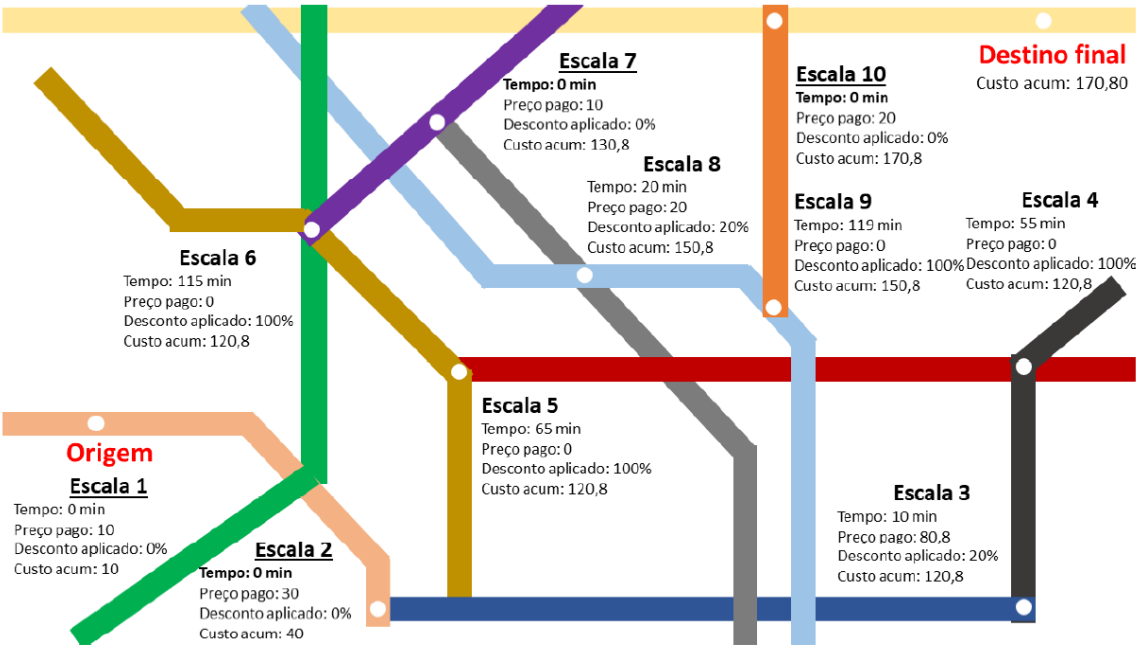

Figura 1. Exemplo do problema com 10 escalas

3. Objetivo

O objetivo nesta etapa é determinar o custo mínimo para João realizar o transporte para cada atração turística na ordem listada. No entanto, ele aceita esperar o tempo que for necessário nas transições de linhas de metrô (apesar de passar metrô a cada minuto) de forma a obter o menor custo de transporte possível em sua estadia em Nova York.

4. Modelagem Computacional

4.1. Programação Dinâmica

Para calcular o valor mínimo que podemos obter *com* o item i, primeiro precisamos comparar se o tempo máximo e a quantidade máxima de descontos ainda serão respeitados na capacidade dada pelo problema. Obviamente, se não, não incluímos e não é necessário performar os cálculos de custo. Nesse caso, a solução para o problema é simplesmente o menor valor que podemos obter *sem* o item i.

Assim, suponhamos que a adição do item i respeite os parâmetros de tempo e desconto máximo. Nesse caso nós o incluímos, alterando o novo custo para visitar a nova atração turística.

Assim, a decisão consiste em continuar da próxima escala até a final reiniciando o desconto, ou continuar da atual mantendo o desconto que já estamos aplicando.

4.2. Equação de Bellman

Para a aplicação do conceito de programação dinâmica utilizou-se na solução deste problema uma variação da Equação de Bellman

$$C(e_a,e_d)= \begin{cases} 999999999, \text{ se } (e_a-e_d) > \text{ou} = D \text{ e } tempo[e_a]-tempo[e_d] > \text{ou} = T \\ 0, \text{ se } e_a \text{ é a escala final} \\ \min(C(e_a+1, e_d+1),\ C(e_a, e_d)), \text{ de outra forma} \end{cases}$$

em que D é o limite de escalas, T o limite de tempo, e_a como a escala atual, e_d a escala onde o desconto começa, e $C(e_a,e_d)$ o custo de ir da escala atual até a escala final dado que o desconto começa em e_d.

4.3. Complexidade Assintótica

O algoritmo possui complexidade de espaço de $\Theta(n)$ - n sendo a quantidade de escalas - visto que utilizamos vetores para armazenar tempo, preços e custo acumulado com n indexes.

Além disso, a complexidade de tempo é O(n+d) - d sendo a quantidade máxima de descontos - pois, realizamos o trajeto das escalas e acumulamos e/ou reiniciamos os descontos se necessário.

5. Compilação e Execução

Para compilar o projeto, há um makefile na raiz do projeto do VSCode (IDE usada no desenvolvimento do projeto). Para executá-lo, basta digitar make dentro desta pasta. Para executar o projeto basta digitar ./tp03 . Foi criada uma pasta de testes para fins de organização do projeto, então para executá-los basta digitar ./tp03 < "nomeDoarquivo".in (sem aspas). Sendo os argumentos:

• < "nomeDoarquivo".txt >: Arquivo de entrada com a quantidade de escalas, quantidade máxima de escalas com descontos cumulativos no intervalo T, tempo

máximo para aplicação de descontos, além dos descontos fornecidos, tempo e preço de cada escala.

6. Implementação

6.1. Linguagem

Todo o algoritmo e os exemplos de código usados aqui foram desenvolvidos em C++17.

6.2. Estruturas de Dados e Algoritmos

Neste projeto foram utilizadas apenas estruturas simples de armazenamento, como *arrays* e estruturas de iteração como *for*. Os principais conceitos utilizados foram baseados nos estudos de programação dinâmica e equação de Bellman mencionados acima.

6.3. Pseudocódigo

```
Para i < N :
    custo ← custo_acumulado[i];
    temp ← 0;
    desconto ← 0;
    Para j < D:
        Se temp<T e i+j<N:
          desconto_acumulado← desconto_acumulado + descontos[j];
          custo← custo + preços[i+j] * (100 - desconto_acumulado);
          custo_acumulado[i+j+1]← min(custo_acumulado[i+j+1], custo);
    temp← temp+tempo[i+j];
```

7. Conclusão

A maior dificuldade se deu na decisão de escolha da implementação do conceito de programação dinâmica. Previamente me pareceu simples aplicar o conceito do algoritmo Knapsack, mas percebi que não seria tão trivial. Então utilizei o que consegui aproveitar, que fora a Equação de Bellman, fazendo as adequações necessárias para o problema que tinha em mãos.

8. Bibliografia

Referências

- Notas de aula das Prof.ª Olga e Jussara. DCC, UFMG.