Projeto 1 - Algoritmos Avançados

Marcus Vinícius Medeiros Pará - 11031663

Outubro de 2019

1 Problema dos trens

Sabendo os horários de chegada e de saída dos trens de uma estação, gostaríamos de saber quantas plataformas são necessárias para evitar a sobreposição de trens na mesma plataforma.

Entrada	Saída
6	2
02:00 02:10 03:00 03:20 03:50 05:00	
02:30 03:40 03:20 04:30 04:00 05:20	

Tabela 1: Entrada e saída do problema dos trens.

Entrada: A primeira linha da entrada é o valor inteiro que indica quantos horários serão lidos. As linhas seguintes contém, respectivamente, os horários de chegada e de saída dos trens. Cada horário está no formato hh:mm. Por exemplo: “05:50”, “15:20”, “16:10”.
Saída: A saída é um inteiro indicando quantas plataformas são necessárias.

1.1 Solução

O número de plataformas necessárias na estação é igual ao número máximo de trens que ficam concomitantemente na estação. Se houver mais plataformas, haverá plataformas que poderão nunca ser utilizadas. Se houver menos platarformas, uma estação teria que comportar mais de um trem, o que causaria atrasos.

Tomemos um instante i com n_i trens na estação. Se o horário mais próximo que vier for de chegada, então em $i+1$, haverá $n_{i+1} = n_i + 1$ trens na estação. Se o horário mais próximo for de chegada, em $i+1$, haverá $n_{i+1} = n_i - 1$ trens. Então, para resolver o problema, basta inicializar um contador em zero, e percorrer os horários disponíveis, sempre pegando o horário mais próximo,

ou seja o mínimo dos horários. Se a entrada não estiver ordenada, basta fazer as n inserções de horários numa *heap-min* e depois fazer n remoções. Segue que o algoritmo será $O(n \cdot lg(n))$.

2 Código Huffman

Dado um conjunto de caracteres $S = \{c_1, ..., c_n\}$ e f_i a frequência do caracter c_i, queremos saber como codificá-los de maneira a minimizar a quantidade de bits utilizados. O código Huffman utiliza sequências de bits de tamanhos distintos para cada caracter, de modo a utilizar sequências menores para codificar caracteres mais frequentes.

Entrada	Saída
5	11
a 32	10
b 25	00
c 20	011
d 18	010
e 5	

Tabela 2: Entrada e saída do problema do código Huffman.

Entrada: A primeira linha é um valor inteiro N que indica quantos caracteres serão codificados. Cada uma das próximas N linhas contém um caracter e um inteiro indicando sua frequência absoluta, respectivamente.
Saída: A saída contém M linhas. Cada linha contém um caracter. Os caracteres são ordenados de acordo com a tabela ASCII.

2.1 Solução

Para representar uma codificação de um dado conjunto de n caracteres, podemos utilizar uma árvore binária com n folhas. Cada uma das folhas representa um caracter. Para obter a respectiva codificação, percorremos a árvore desde a raiz até a folha desejada. Sempre que vamos para o nó filho da esquerda, escrevemos um '0' no final da codificação, se vamos ao nó filho da direita, escrevemos um '1'.

A árvore que representa a codificação ótima é feita de modo que se uma folha i tem profundidade menor que uma folha j, então a frequência do caracter i é maior ou igual à frequência caracter j. É fácil notar que isso preenche o requisito de usar códigos de menos bits para representar caracteres mais frequentes.

O algoritmo para resolver o problema pode ser descrito da seguinte forma.

```
Huffman(S)
if Tamanho de S = 2 then
    Codifique na árvore T um dos caracteres como 0, e outro como 1
else
    Remove caracteres u e v de menor frequência de S
    Insira caracter w em S, com frequência f_w = f_u + f_v
    Huffman(S)
    Na árvore T, insira os caracteres u e v como filhos de w
end
```

Podemos notar que, a cada chamada recursiva, o tamanho de S decresce em uma unidade. Portanto, fazemos $O(n)$ remoções e insreções em S. Se utilizarmos uma *heap-min* para representar o conjunto S de caracteres, teremos que o algoritmo é $O(n \cdot lgn$.