Nome: Jéssica Ap. Almeida dos Santos **RA:** 101992
UC: Projeto e Análise de Algoritmos **Docente:** Reginaldo Kuroshu

Exercício 3 de Algoritmos Gulosos: Paradas mínimas para abastecimento do carro entre a cidade A e B

a) A solução desenvolvida inicia-se com a comparação inicial acerca do último elemento do vetor de distâncias *D* entre os *n* postos, elemento o qual corresponde a distância do posto de combustível mais próximo a cidade B. Sabendo-se que a autonomia do carro é dado por *m*, retorna-se 0 caso a distância seja menor ou igual a *m*, o que dispensa a necessidade de parada em tal situação.

Verificado o último elemento do vetor *D*, percorre-se então a extensão das n-1 distâncias declaradas, as quais correspondem às distâncias de cada posto a partir do **ponto inicial da viagem(0 km)**. Desta forma, inicializa-se uma variável denominada *distancia_percorrida* com a primeira distância do vetor. Em seguida, realiza-se uma segunda comparação, a qual soma o valor contido na variável *distancia_percorrida* com a diferença entre a distância seguinte(*D[i+1]*) com a distância atual(*D[i]*), diferença a qual resulta na real distância entre o posto atual e a próximo posto de combustível. Se a soma da distância percorrida com a diferença calculada supera os *m* km de autonomia do carro, incrementa-se a variável *paradas* em um e *distancia_percorrida* recebe o valor da diferença. Caso contrário, soma-se à *distancia_percorrida* a diferença das distâncias entre os postos calculada. Repete-se tal procedimento n-1 vezes e após isso retorna-se o valor contido na variável *paradas*. A solução descrita encontra-se na função abaixo:

```
int calcula_Paradas(int n,int D[],int m){
   int i;
   int distancia_percorrida = D[1];
   int paradas = 0;

   if(D[N] <= m) //Se o posto mais próximo de B tiver distância menor ou igual a m
      return 0; //Não há necessidade de parada

   for(i=1;i<=n-1;i++){ //Caso contrário
//Se a distância percorrida somada a distância entre o posto atual e o próximo é maior que a autonomia m do carro
      if(distancia_percorrida + (D[i+1] - D[i]) > m){
         distancia_percorrida = D[i+1] - D[i]; //atualização da distância para não seguir viagem
         Paradas++; //realiza parada
      }
      else
         //Segue viagem incrementando a distância percorrida
         distancia_percorrida = distancia_percorrida + (D[i+1] - D[i]);
   }
```

```
    return paradas;
}
```

b) Prova de corretude da solução ótima

Seja S uma solução ótima do problema em questão. Supondo que o vetor de distâncias D é composto por d_1, d_2….d_n onde d_i é a parada correspondente a distância d_i e também que p é a primeira parada realizada pelo algoritmo guloso desenvolvido, supõe-se que existe solução ótima com a primeira parada p. A partir de tal suposição, temos que:

$$d_1 = p,$$

fazendo de S uma solução válida.

Supõe-se agora que $d_1 \neq p$. Já que o algoritmo termina sua execução no penúltimo posto possível, visto que o último foi verificado anteriormente, sabe-se então que d_1 é anterior a p.

Tendo-se S' como uma solução ótima válida e $S' = \langle p, d_2, d_3 \ldots . d_n \rangle$. Nota-se que $|S'| = |S|$. Por definição da escolha gulosa, sabe-se que é possível alcançar p. Com isso, já que S é solução ótima e a distância entre p e d_2 não é superior à distância entre d_1 e d_2, conclui-se que ainda haverá combustível necessário para que o carro siga viagem entre p e d_2. O restante de S' é como a solução S, o que a torna igualmente válida.