PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Engenharia de Software
Fundamentos de Projeto e Análise de Algoritmos
Prof(a): João Caram Santos de Oliveira

Carlos Roberto Alves de Almeida Júnior
Jully Ketely Aves da Silva
Laura Lourdes Coutinho Rodrigues

Problema

Uma empresa de distribuição e logística possui uma frota composta por **N** caminhões. Semanalmente, esta empresa organiza suas entregas em **M** rotas, as quais devem ser distribuídas entre os caminhões disponíveis. A empresa deseja fazer a distribuição de maneira que cada caminhão cumpra a **mesma quilometragem**, evitando assim que ao final do período existam caminhões ociosos enquanto outros continuam executando várias rotas. Se não for possível cumprir a mesma quilometragem, que a **diferença** entre a quilometragem dos caminhões seja a **menor** possível, diminuindo o problema.

Solução

O grupo decidiu que cada membro implementaria uma solução:

Carlos Roberto	Programação Dinâmica
Jully Ketely	Backtracking
Laura Lourdes	Divisão e Conquista

Na geração de rotas fornecida pelo professor, os valores representados abaixo foram pré-definidas para início da solução do problema. Para cada iteração do *backtracking* a quantidade de rotas (*qtd_rotas*) é incrementado. Para resolução do problema é considerada a **existência de 3 caminhões**.

```
qtd_rotas = 6
qtd_conjunto = 10
dispersao = 75
whileTime = True
```

Backtracking

A função de backtracking foi implementada para distribuir rotas entre caminhões de forma a minimizar a diferença de quilometragem entre eles.

O *bakctracking* busca realizar uma exploração sistemática de todas as opções possíveis. A função do *backtracking* gera soluções parciais incrementais, tentando atribuir cada rota a um dos caminhões. Claro que, são realizados testes para verificar se a solução parcial é

viável ou se pode ser descartada. Se caso a solução parcial não atender aos critérios é realizado a poda, ou seja, o algoritmo abandona esse ramo da busca. A cada vez que uma solução completa é alcançada, o algoritmo verifica se essa solução é a melhor que a melhor solução encontrada até agora.

O critério de poda funciona da seguinte maneira:

- A poda recebe a distribuição atual, o índice da rota a ser considerada, a melhor diferença atual e o tamanho das rotas;
- Calcula a quilometragem total em cada caminhão;
- Verifica se a distribuição parcial já é pior que a melhor solução encontrada, **realizando a poda**;
    - Se o índice da rota for menor que o tamanho total das rotas, verifica se a diferença parcial já é menor ou igual à melhor diferença encontrada.
- Retorna *True* se a poda for válida, indicando que o ramo da árvore de busca pode ser podado.

Utilizando o *GeradorDeProblemas* fornecido pelo professor, o backtracking ultrappasou seu limite de 30 segundos, chegando a 181s, com o **tamanho de rotas, igual a 47**, e conferindo o seguinte conjunto de rotas, que posteriormente foram analisados pela Programação Dinâmica e Divisão e Conquista.

Conjunto resultante do Backtracking

```
[[438, 185, 726, 391, 579, 181, 462, 262, 960, 608, 54, 149, 115, 823, 880, 660,
640, 398, 809, 174, 29, 913, 511, 95, 19, 20, 196, 961, 104, 736, 680, 620, 921,
303, 439, 723, 839, 824, 804, 500, 721, 142, 224, 159, 668, 290, 548], [474, 434,
257, 973, 191, 638, 721, 51, 623, 197, 747, 524, 639, 986, 103, 655, 859, 814, 321,
369, 441, 17, 584, 254, 209, 133, 555, 110, 646, 383, 896, 792, 565, 532, 237, 488,
801, 654, 688, 226, 296, 67, 211, 128, 828, 165, 714], [877, 946, 498, 522, 193,
176, 699, 957, 57, 68, 673, 385, 304, 227, 458, 270, 355, 644, 837, 924, 789, 477,
57, 119, 771, 66, 960, 563, 193, 174, 817, 42, 509, 540, 131, 85, 929, 253, 751,
468, 619, 606, 346, 442, 216, 778, 706], [932, 346, 248, 810, 125, 346, 246, 113,
255, 768, 435, 903, 217, 397, 925, 238, 88, 495, 612, 509, 68, 924, 459, 161, 234,
963, 637, 478, 195, 409, 395, 152, 481, 750, 211, 660, 380, 289, 220, 356, 143, 45,
721, 595, 74, 300, 152], [126, 885, 489, 482, 441, 188, 791, 364, 238, 920, 315,
631, 173, 821, 826, 666, 988, 639, 940, 187, 604, 579, 888, 248, 101, 434, 890, 547,
513, 895, 234, 618, 497, 870, 486, 298, 27, 760, 451, 766, 906, 265, 266, 348, 457,
606, 240], [926, 962, 618, 981, 165, 948, 600, 932, 18, 440, 58, 54, 494, 764, 800,
529, 47, 448, 556, 416, 503, 842, 830, 726, 854, 113, 476, 520, 259, 98, 584, 784,
25, 722, 762, 429, 985, 236, 148, 380, 315, 257, 676, 105, 29, 504, 698], [325, 178,
733, 483, 891, 597, 778, 610, 37, 478, 487, 803, 154, 908, 733, 428, 202, 821, 905,
392, 654, 488, 733, 661, 14, 894, 655, 85, 407, 927, 720, 353, 556, 46, 316, 449,
783, 289, 727, 981, 640, 568, 602, 826, 225, 819, 714], [95, 648, 914, 356, 766,
967, 69, 540, 981, 944, 703, 385, 922, 65, 126, 405, 387, 489, 646, 54, 387, 388,
192, 676, 585, 734, 81, 587, 739, 805, 222, 154, 355, 257, 565, 129, 677, 502, 768,
974, 783, 607, 905, 732, 227, 784, 775], [792, 243, 341, 374, 932, 286, 528, 709,
446, 296, 68, 77, 394, 366, 476, 510, 526, 552, 862, 138, 494, 739, 557, 707, 858,
558, 934, 312, 475, 648, 389, 832, 702, 981, 722, 841, 976, 917, 80, 205, 926, 918,
481, 102, 95, 413, 932], [196, 308, 91, 124, 181, 295, 127, 726, 164, 521, 167, 908,
222, 546, 987, 655, 36, 313, 439, 711, 567, 967, 148, 498, 368, 443, 467, 768, 673,
34, 311, 206, 77, 35, 905, 834, 250, 867, 983, 545, 673, 356, 434, 605, 364, 684,
674]]
```

Divisão e Conquista

É utilizada a Divisão e Conquista para outra solução do problema apresentado, distribuir rotas entre caminhões. O algoritmo recebe duas entradas: uma lista de rotas e o número de caminhões.

A função *diferenca_entre_caminhoes(caminhoes)* retorna a diferença entre a maior e a menor quilometragem entre os caminhões.

A função *distribuir_rotas_divisao_conquista(rotas, num_caminhoes)* distribui as rotas entre os caminhões. Se não houver rotas, ela retorna uma lista vazia para cada caminhão. Caso contrário, ela classifica as rotas em ordem decrescente e chama a função *divide_rotas(rotas, num_caminhoes)*.

A função *divide_rotas(rotas, num_caminhoes):* divide as rotas entre os caminhões. Se não houver rotas, ela retorna uma lista vazia para cada caminhão. Se houver apenas um caminhão, ela retorna todas as rotas para esse caminhão. Caso contrário, ela divide as rotas entre os caminhões de tal forma que a diferença entre a maior e a menor quilometragem seja minimizada. **Ela faz isso verificando todas as possíveis divisões das rotas e escolhendo a que resulta na menor diferença de quilometragem.**

Resultado (Conjunto utilizado pelo backtracking)

```
divisaoconquista(conjuntoResultante, 3)
```

```
------------------ DIVISÃO E CONQUISTA ------------------

Distribuição encontrada para minimizar a diferença de quilometragem entre os caminhões:
Caminhão 1: [961, 960, 921, 913, 880, 839, 824, 823]
Caminhão 2: [809, 804, 736, 726, 723, 721, 680, 668, 660, 640, 620]
Caminhão 3: [608, 579, 548, 511, 500, 462, 439, 438, 398, 391, 303, 290, 262, 224, 196, 185, 181, 174, 159, 149, 142, 115, 104, 95, 54, 29,
 20, 19]
Diferença mínima de quilometragem: 666 km
Distribuição encontrada para minimizar a diferença de quilometragem entre os caminhões:
Caminhão 1: [986, 973, 896, 859, 828, 814, 801, 792, 747]
Caminhão 2: [721, 714, 688, 655, 654, 646, 639, 638, 623, 584, 565]
Caminhão 3: [555, 532, 524, 488, 474, 441, 434, 383, 369, 321, 296, 257, 254, 237, 226, 211, 209, 197, 191, 165, 133, 128, 110, 103, 67, 51
, 17]
Diferença mínima de quilometragem: 569 km
Distribuição encontrada para minimizar a diferença de quilometragem entre os caminhões:
Caminhão 1: [960, 957, 946, 929, 924, 877, 837, 817]
Caminhão 2: [789, 778, 771, 751, 706, 699, 673, 644, 619, 606, 563]
Caminhão 3: [540, 522, 509, 498, 477, 468, 458, 442, 385, 355, 346, 304, 270, 253, 227, 216, 193, 193, 176, 174, 131, 119, 85, 68, 66, 57,
57, 42]
Diferença mínima de quilometragem: 384 km
Distribuição encontrada para minimizar a diferença de quilometragem entre os caminhões:
Caminhão 1: [963, 932, 925, 924, 903, 810, 768]
Caminhão 2: [750, 721, 660, 637, 612, 595, 509, 495, 481, 478, 459, 435]
Caminhão 3: [409, 397, 395, 380, 356, 346, 346, 300, 289, 255, 248, 246, 238, 234, 220, 217, 211, 195, 161, 152, 152, 143, 125, 113, 88, 74
, 68, 45]
Diferença mínima de quilometragem: 607 km
Distribuição encontrada para minimizar a diferença de quilometragem entre os caminhões:
Caminhão 1: [988, 940, 920, 906, 895, 890, 888, 885, 870]
Caminhão 2: [826, 821, 791, 766, 760, 666, 639, 631, 618, 606, 604, 579]
Caminhão 3: [547, 513, 497, 489, 486, 482, 457, 451, 441, 434, 364, 348, 315, 298, 266, 265, 248, 240, 238, 234, 188, 187, 173, 126, 101, 2
7]
Diferença mínima de quilometragem: 233 km
```

Resultado (Conjunto de rotas teste)

```
divisaoconquista(rotas_teste, 3
```

```
------------------- DIVISÃO E CONQUISTA -------------------

Distribuição encontrada para minimizar a diferença de quilometragem entre os caminhões:
Caminhão 1: [40, 39, 39, 39, 38, 38, 38, 36]
Caminhão 2: [35, 35, 35, 33, 33, 32, 32, 32, 32]
Caminhão 3: [31, 31, 30, 30, 29, 29, 29, 29, 28, 28]
Diferença mínima de quilometragem: 13 km
Distribuição encontrada para minimizar a diferença de quilometragem entre os caminhões:
Caminhão 1: [55, 55, 53, 51, 51, 51]
Caminhão 2: [44, 43, 43, 43, 42, 42, 32, 32]
Caminhão 3: [32, 32, 30, 30, 29, 29, 29, 29, 27, 27, 25, 24]
Diferença mínima de quilometragem: 27 km
```

Programação Dinâmica

O Algoritmo recebe as rotas e o número de caminhões, a partir dessas informações calcula a Quilometragem Média (*kmMedia)* que nada mais é que a soma das rotas dividido pela quantidade de caminhões; tamanho da coluna (*tamColuna*) que é a *kmMedia* * intervalo superior (0.10).

A solução do Algoritmo foi baseado no problema da Soma de Subconjunto, utilizando V, F e “.”(ponto, que representa que a linha de cima é verdadeira). Abaixo a descrição do algoritmo:
- Verdadeiro (ponto) caso, M[i-1][j] seja V
- Verdadeiro caso, M[i-1][j-Vi] + Vi seja V
- Caso contrário, F

Além disso quando o valor da coluna, neste caso o valor de j, esteja no intervalo de *kmMedia* o algoritmo salva o valor de i e j em outras duas variáveis e encerra as iterações, pois uma solução foi encontrada.

Com o valor de i (linha) e j (coluna) salvos, é feita a busca pelas rotas que fazem parte da solução para o caminhão. A formação do resultado verifica se a posição atual é V, se sim adiciona o valor de rota[j] na solução e vai para M[i-1][j-rotas[j-Vi]]; caso não for V e seja ponto diminui valor de linha em 1, indo na próxima iteração para a linha e cima até achar V.

Resultado (Conjunto que inicia pelo último conjunto do backtracking):
- Na imagem abaixo está contido somente pequena parte do primeiro conjunto que é gerado pelo backtracking, como os últimos conjuntos ficam com quantidade de rotas enormes, não é possível colocar uma imagem de exemplo.

```
------------------- PROGRAMAÇÃO DINAMICA -------------------
Distribuição encontrada para minimizar a diferença de quilometragem entre os caminhões:
Caminhão 1: [579, 548, 511, 500, 462, 439, 438, 398, 391, 303, 290, 262, 224, 196, 185, 181, 174, 159, 149, 142, 115, 104, 95, 54, 29, 20, 19] -- Quilometragem: 6967 km
Caminhão 2: [736, 726, 723, 721, 680, 668, 660, 640, 620, 608] -- Quilometragem: 6782 km
Caminhão 3: [804, 809, 823, 824, 839, 880, 913, 921, 960, 961] -- Quilometragem: 8734 km
Distribuição encontrada para minimizar a diferença de quilometragem entre os caminhões:
Caminhão 1: [532, 524, 488, 474, 441, 434, 383, 369, 321, 296, 257, 254, 237, 226, 211, 209, 197, 191, 165, 133, 128, 110, 103, 67, 51, 17] -- Quilometragem: 6818 km
Caminhão 2: [714, 688, 655, 654, 646, 639, 638, 623, 584, 565, 555] -- Quilometragem: 6961 km
Caminhão 3: [721, 747, 792, 801, 814, 828, 859, 896, 973, 986] -- Quilometragem: 8417 km
Distribuição encontrada para minimizar a diferença de quilometragem entre os caminhões:
Caminhão 1: [522, 509, 498, 477, 468, 458, 442, 385, 355, 346, 304, 270, 253, 227, 216, 193, 193, 176, 174, 131, 119, 85, 68, 66, 57, 57, 42] -- Quilometragem: 7091 km
Caminhão 2: [778, 771, 751, 706, 699, 673, 644, 619, 606, 563, 540] -- Quilometragem: 7350 km
Caminhão 3: [789, 817, 837, 877, 924, 929, 946, 957, 960] -- Quilometragem: 8036 km
```

Análise geral

Mesmo o *backtracking* utilizando em seu critério de poda a relação de qual a melhor diferença entre a distribuição atual e a melhor atualmente, ainda, sim, na quantidade de

rotas 47, o *backtracking* não conseguiu executar em menos de 30 segundos, ultrapassando o tempo limite e chegando ao pico de 181 segundos. Os conjuntos anteriores, o *backtracking* conseguiu executar, alcançando o pico médio de 16 segundos.

Já na programação dinâmica foi utilizado uma condição de parada, para quando a coluna atual está dentro do intervalo. Este intervalo é definido pela soma das rotas dividido pela quantidade de número de caminhões (*kmMedia*), e os limites inferior e superior com diferença de 10%, ou seja, limite inferior: *kmMedia* * 0.90 e limite superior: *kmMedia* * 1.10.

Com essa condição de parada, o algoritmo para a quantidade de 47 rotas levou o tempo médio de 0.44 segundos, o tempo médio é calculado depois das 10 execuções do algoritmo. E o último conjunto com a quantidade 470 rotas levou em média 42.66 segundos.

Especificações das máquinas utilizadas

Dono: Carlos Roberto
Processador: Ryzen 7 5700G 3.8GHz 8 núcleos e 16 threads
RAM instalada: 2 memórias de 16 GB RAM
Tipo de sistema: Windows 10

Dono: Jully Ketely
Processador: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
RAM instalada: 12,0 GB (utilizável: 11,8 GB)
Tipo de sistema: Sistema operacional de 64 bits, processador baseado em x64

Dono: Laura Lourdes
Processador: 11th Gen Intel(R) Core(™) @ 2,30GHz 2304 MHz 8 Core(s) 16 Logical Processor(s)
RAM instalada: 16 GB RAM
Tipo de sistema: Windows 11