Logística Urbana para Entrega de Mercadorias

Desenho de Algoritmos

Alunos - G57

David Marques - up201905574
Gonçalo Marques - up202006874
Rui Soares - up202103631

Orientadores

Ana Paula Tomás
Rosaldo Rossetti
Gonçalo Leão

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Descrição do Problema

Uma empresa com base tecnológica pretende inovar, criando uma plataforma eletrónica de crowdsourcing para a entrega de mercadorias em zonas urbanas.

A empresa tem o seu próprio **armazém**, onde recebe e mantém as **mercadorias** enviadas pelos fornecedores, que ficam a aguardar o transporte para o destino final.

A empresa realiza dois tipos de serviços, nomeadamente a entrega **normal** e a entrega **expresso**.

Neste contexto surgiram **3** abordagens às formas de entrega:

- **Cenário 1**: otimização do número de estafetas (entrega normal);
- **Cenário 2**: otimização do lucro da empresa (entrega normal);
- **Cenário 3**: otimização das entregas expresso.

Cenário 1 - Formalização

- **Input:**
  - lista de carrinhas = {c1,c2,c3,...,cn};
  - lista de encomendas = {e1,e2,e3,...,em};
- **Output:** lista de carrinhas com encomendas;
- **Restrições**:
  - peso e volume máximo de cada carrinha não podem ser ultrapassados;

Cenário 1 - Descrição de Algoritmos

Neste cenário foi usado um algoritmo ***greedy***:

1. **ordenar** o vetor de carrinhas pela **soma** dos valores *maxPeso* e *maxVol* por ordem **crescente;**
2. **ordenar** o vetor de encomendas pela **soma** dos valores peso e volume por ordem **decrescente**;
3. **percorrer** o vetor das carrinhas até ao final do vetor ou até o vetor das encomendas ficar vazio;
4. em cada **iteração** percorrer o vetor das encomendas até ao final do vetor ou até que o peso ou o volume das **encomendas** dentro da carrinha seja igual à capacidade da carrinha:
    a. se o peso e o volume da **encomenda** em conjunto com o peso e volume de todas as **encomendas dentro** da carrinha forem **menor ou igual** ao peso e volume máximo da **carrinha** esta é inserida dentro da **carrinha** e removida do vetor das **encomendas**.

Cenário 1 - Análise da Complexidade

- vetor de carrinhas: **n** elementos;
- vetor de encomendas: **m** elementos;

1. ordenar os vetores usando a função *sort* que tem complexidade temporal **O(nlog(n))**;
2. um ciclo ***while*** com **m** elementos num ciclo ***for*** com **n** elementos resulta em complexidade O(n*m);

Assim, temos um complexidade temporal total de O[nlog(n)+ mlog(m) + n*m] = **O(n*m)**.

Em relação à complexidade espacial como não são criadas novas estruturas de dados a complexidade espacial é **O(1)**.

Cenário 1 - Resultados da Avaliação Empírica

volMax	pesoMax	total
42	100	142
19	115	134
28	95	123
20	89	109
35	55	90

volume	peso	total
9	9	18
3	19	22
3	19	22
3	19	22
3	20	23
3	23	26
16	13	29
16	13	29
6	26	32
6	26	32
20	13	33
19	16	35
25	13	38
22	16	38
23	16	39
16	26	42
13	29	42
19	28	47
26	22	48
25	28	53

Carrinha	volEnc	volCar - volEnc	pesoEnc	pesoCar - pesoEnc	Eficiencia de Volume	Eficiencia de Peso
142	9	33	9	91	88%	99%
	3	30	19	72		
	3	27	19	53		
	3	24	19	34		
	3	21	20	14		
	16	5	13	1		
134	3	16	23	92	100%	31%
	16	0	13	79		
123	6	22	26	69	100%	82%
	6	16	26	43		
	16	0	26	17		
109	20	0	13	76	100%	15%
90	19	16	16	39	91%	82%
	13	3	29	10		

Encomendas Entregues:	70%

Eficiencia Média	
96%	62%

Na tabela acima podemos ver os resultados ideais do cenário. Usando o algoritmo descrito anteriormente obtemos valores que estão dentro das margens de erro aceitáveis. Existem algoritmos que podem dar respostas melhores ao problema (e.g. *brute-force*) mas têm complexidades temporais muito maiores do que o nosso algoritmo.

Cenário 2 - Formalização

- **Input:**
  - lista de carrinhas = {c1,c2,c3,...,cn};
  - lista de encomendas = {e1,e2,e3,...,em};
- **Output**:
  - lucro máximo;
  - carrinhas usadas;
  - lucro por carrinha;
- **Restrições**:
  - peso e volume máximo de cada carrinha não podem ser ultrapassados;
  - não utilizar carrinhas (ou encomendas) que causem prejuízo.
- **Função Objetivo**: max (lucro).

Cenário 2 - Descrição do Algoritmo

Para este problema usamos um algoritmo ***greedy***:

1. ordenar **encomendas** por ordem de decrescente de **recompensa**;
2. ordenar **carrinhas** por ordem crescente de **custo**;
3. enquanto houver encomendas, ou existir peso e volume livre na carrinha, encher a carrinha:
   a. quando uma encomenda ultrapassa a capacidade restante da carrinha, tentar a seguinte até uma conseguir ser inserida ou chegar ao fim da lista;
4. calcular o lucro gerado por essa carrinha;
5. **repetir** até uma carrinha dar **prejuízo**, porque a partir dessa nenhuma carrinha dará lucro.
6. calcular o lucro final com a seguinte equação:

> máximo de lucro com o mínimo de custos.

$$\sum_{x=0}^{encomendasUsadas.size} x.recompensa - \sum_{x=0}^{carrinhasUsadas.size} x.custo$$

Cenário 2 - Análise de Complexidade

- vetor de carrinhas: **n** elementos;
- vetor de encomendas: **m** elementos;

1. ordenar os vetores usando a função *sort* que tem complexidade temporal **O(nlog(n))**;
2. um ciclo ***while*** com **m** elementos num ciclo ***for*** com **n** elementos resulta em complexidade **O(n*m)**;

Assim, temos um complexidade temporal total de O[nlog(n)+ mlog(m) + n*m] = **O(n*m)**. .

Para além disso, não são criadas novas estruturas de dados, pelo que a complexidade espacial é **O(1)**.

Cenário 2 - Resultados da Avaliação Empírica

volMax	pesoMax	custo
19	115	2 025,00 €
28	95	2 378,00 €
42	100	3 796,00 €
20	89	4 015,00 €
35	55	5 226,00 €

volume	peso	recompensa
6	26	1 991,00 €
19	16	1 967,00 €
3	20	1 645,00 €
26	22	1 628,00 €
16	26	1 590,00 €
20	13	1 542,00 €
13	29	1 511,00 €
3	19	1 432,00 €
3	23	1 410,00 €
25	28	1 370,00 €
6	26	1 291,00 €
25	13	1 251,00 €
9	9	1 137,00 €
16	13	1 121,00 €
22	16	982,00 €
19	28	946,00 €
23	16	868,00 €
3	19	754,00 €
3	19	585,00 €
16	13	367,00 €

Carrinha	volEnc	volCar - volEnc	pesoEnc	pesoCar - pesoEnc	Recompensa Carrinha	Lucro Carrinha
2025	6	13	26	89	7 232,00 €	5 207,00 €
	3	10	20	69		
	3	7	19	50		
	3	4	23	27		
	3	1	19	8		
2378	19	9	16	79	3 843,00 €	1 465,00 €
	6	3	26	53		
	3	0	19	34		
3796	26	16	22	78	3 218,00 €	-578,00 €
	16	0	26	52		
4015	20	0	13	76	1 542,00 €	-2 473,00 €
5226	13	22	29	26	2 648,00 €	-2 578,00 €
	9	13	9	17		

Custo Total	4 403,00 €
Recompensa Total	11 075,00 €
Lucro Total	6 672,00 €

Estas tabelas demonstram os dados utilizados para os testes empíricos e as respostas ideais.
O algoritmo utilizado conseguiu aproximar-se de uma resposta ideal.
Outros tipos de algoritmos que conseguissem alcançar uma resposta mais precisa, seriam mais lentos.

Cenário 3 - Formalização

- **Input**:
  - lista de encomendas = {e1, e2, e3, …, en};
- **Output**: tempo médio mínimo;
- **Restrições**:
  - 1 encomenda por entrega;
  - no máximo 28800 segundos por dia (8 horas = 1 dia de trabalho).
- **Função Objetivo**: min (tempo médio).

Cenário 3 - Descrição do Algoritmo

Para este problema usamos um algoritmo ***greedy***:

1. ordenar as **encomendas** por ordem **<u>crescente</u>** de **duração** da entrega.
2. enquanto houverem encomendas ou enquanto o tempo da entrega não passar as **8 horas** de trabalho diárias, são guardados os tempos das encomendas numa lista (*time*) para serem usados no cálculo do tempo médio das entregas.

Finalmente, para calcular o tempo médio das entregas, são usados os tempos na lista referida anteriormente da seguinte forma:

$$\sum_{i=1}^{time.size} time[time.size - 1] * i$$

Cenário 3 - Análise de Complexidade

- vetor de encomendas: **n** elementos;

1. ordenar o vetor usando a função *sort* tem complexidade temporal **O(nlog(n))**;
2. um ciclo ***for*** com **n** elementos tem complexidade **O(n)**;
3. outro ciclo ***for*** com **n** elementos tem complexidade **O(n)**;

Assim, temos um complexidade temporal total de O[nlog(n) + n + n] = **O(nlog(n))**. .

Para além disso, não são criadas novas estruturas de dados, pelo que a complexidade espacial é **O(1)**.

Cenário 3 - Resultados da Avaliação Empírica

Estas tabelas demonstram os dados utilizados para os testes empíricos e as respostas ideais.
O algoritmo utilizado conseguiu alcançar aproximadamente a resposta ideal.
Outros tipos de algoritmos que conseguissem alcançar uma resposta mais precisa, seriam mais lentos.

Ou seja, para estas 20 encomendas expresso, que só serão entregues em 6 dias, foram obtidos estes tempos médios mínimos de entrega para cada dia.

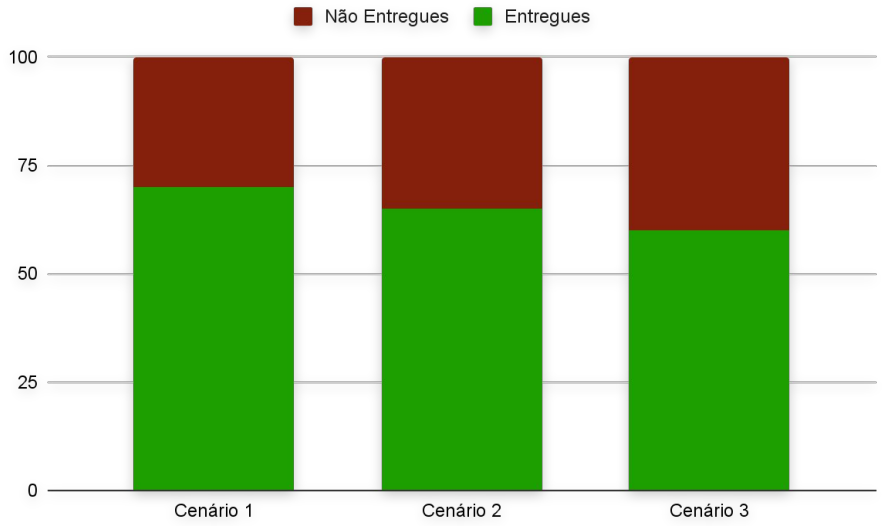
volume	peso	duração (s)
20	13	1280
22	16	2290
3	20	3160
6	26	3180
13	29	3250
25	13	3840
25	28	5350
23	16	6080
16	13	6410
16	13	7030
16	26	7280
3	19	8270
19	16	8650
26	22	9250
3	19	9840
9	9	10190
19	28	10210
3	23	10280
6	26	10350
3	19	10510

Segundos em 1 Dia de Trabalho: 28800

Dia	Sum. Aux	Somatório	Média	Enc. Entregues	Tempo Total
1	1280	102430	12803,75	40,00%	28430
	3570				
	6730				
	9910				
	13160				
	17000				
	22350				
	28430				
2	6410	40570	13523,33	25,00%	20720
	13440				
	20720				
3	8270	51360	17120	33,33%	26170
	16920				
	26170				
4	9840	29870	14935	33,33%	20030
	20030				
5	10210	30700	15350	50,00%	20490
	20490				
6	10350	31210	15605	100,00%	20860
	20860				

Funcionalidades Extra

Acrescentamos também a capacidade de medir a **eficiência de entregas** em cada cenário, em termos do quociente entre o número de pedidos **entregues** e os **não entregues**.



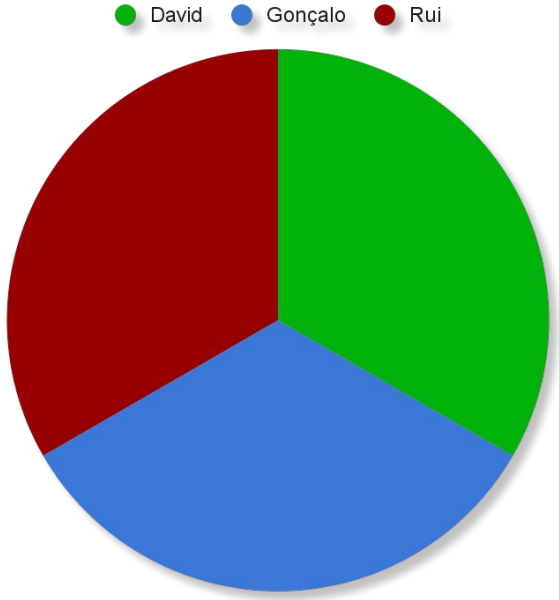
Solução Algorítmica de Destaque

Achamos que vale a pena destacar o algoritmo que obtivemos para o cenário 3, pois apesar de, tal como os outros cenários, ser um algoritmo *greedy*, conseguimos obter nesse caso uma complexidade temporal melhor do que nos outros algoritmos.

$$O(nlog(n)) > O(n^2)$$

Dificuldades Encontradas e Contributos

- Encontrar algoritmos com mais precisão em relação à solução ótima do que os do tipo *greedy* sem sacrificar tempo de execução.



FIM


usar
algoritmos
não-greedy
OR
draw
25
nós
imgflip.com


greedy
greedy
greedy
THIS IS FINE.
imgflip.com