Universidade do Minho
Escola de Engenharia
Departamento de Informática

Trabalho Prático

Métodos de Resolução de Problemas e de Procura

Mestrado Integrado em Engenharia Informática

SISTEMAS DE REPRESENTAÇÃO
DE CONHECIMENTO E RACIOCÍNIO
(2º semestre, 2020/21)

Joana Afonso Gomes
(a84912)

Braga
Junho 2021

Conteúdo

1

Introdução

No âmbito da Unidade Curricular de **Sistemas de Representação de Conhecimento e Raciocínio** foi-me proposto o desenvolvimento deste trabalho prático, consistindo no desenvolvimento de um sistema que permita importar os dados relativos aos diferentes circuitos e representá-los numa base de conhecimento da forma que julgue mais adequada e sistema de recomendação de circuitos de recolha para o caso de estudo.

No presente relatório é descrita a implementação do caso prático proposto as diversas decisões tomadas.

Importante referir que, sendo essa hipótese presente no enunciado do projeto, optei por resolver a **versão simplificada** do problema.

2

Descrição do Trabalho & Análise de Resultados

2.1 Tratamento de Dados

Com vista a concretizar o *parser* da informação do ficheiro *.xlsx* fornecido pelos docentes, que inclui dados dos circuitos de recolha de resíduos urbanos do concelho de Lisboa, e desta forma realizar o processamento dos dados, foi desenvolvido um programa em **Python**.

O *script* começa por converter o *dataset* original num ficheiro *.csv* com vista a facilitar o tratamento dos dados.

De seguida, recorri a um conhecido módulo de *Python*, muito útil no tratamento de dados, com que já me havia sido familiarizada previamente, o módulo ***networkx***.

Escolhi inicialmente prosseguir deste modo pois achei interessante haver uma maior aleatoriedade dos dados e de nodos ajdacentes, tornando a situação um pouco mais "real"quando falamos de dados recolhidos numa situação como a que se baseia o enunciado (um *dataset* de recolha de resíduos), e também por achar que a criação deste tipo de grafo seria um bom caso aplicacional para este módulo. Concluí mais tarde que talvez não fosse o método mais adequado, pois dificultou-me os testes das soluções, mas acabei por não alterar a base de conhecimento e os grafo que tinha inicialmente criado.

De destacar também que nem toda a informação do *dataset* original foi processada para a base de conhecimento. Os nodos das arestas são os IDs que se encontravam na coluna *PONTO_RECOLHA_LOCAL* do *dataset* original.

Foi adicionado ao dataset original uma **Garagem** com ID do Ponto de Recolha 15804 e um Local de **Deposição** (21950).

Apesar de nem toda a informação ser usada ao longo do projeto, a maioria das colunas foi processadas, com exceção apenas da coluna *PONTO_RECOLHA_FREGUESIA* (pois no *dataset* fornecido a freguesia mantinha-se igual em todos os casos).

De seguida apresento o *script* de *Python* desenvolvido e ainda um parcial da base de conhecimento criada (*baseConhecimento.pl*) e do grafo (*grafo.pl*).

2.1.1 parser.py

```
ruas = dict()

def excelToCSV():
    read_file = pd.read_excel('dataset-original.xlsx', engine='openpyxl', nrows= 421)
    read_file.to_csv (r'dataset.csv', encoding='utf-8', index = None, header=False)

def createDataset():
    old_stdout = sys.stdout
    log_file = open("output.txt","w")
    sys.stdout = log_file

    with open('dataset.csv') as file:
            data = csv.reader(file)
            for row in data:
                print ("ponto("+row[0]+", "+ row[1]+","+ row[2]+",'"+
                                (row[4].strip().split(":"))[0]+"','"+
                                row[5]+"','"+ row[6]+"',"+ row[7]+","+
                                row[8]+","+ row[9]+").")
                ruas[(row[4].strip().split(":"))[0]] = []

    sys.stdout = old_stdout

    log_file.close()

def getRandPontoRecolha():
    return ruas_list[(rand.randint(0,len(ruas_list)-1))]
```

```
def createGraph():

    old_stdout = sys.stdout
    log_file = open("outputGrafo.pl","w")
    sys.stdout = log_file

    G = net.Graph()
    G.add_nodes_from(ruas_list)

    while not net.is_connected(G):
        G.add_edge(getRandPontoRecolha(),getRandPontoRecolha())

    dictPoints = net.to_dict_of_lists(G)

    print("grafo([",end = '')

    count = 0

    for i in dictPoints:
        if(count==0):
            print("'" + i +"'",end = '')
            count=1
        else:
            print(",'"+i + "'",end = '')

    print("],[",end = '')

    count2 = 0

    for i in dictPoints:
        for j in dictPoints[i]:
            if(count2==0):
                print("aresta('" + i + "','" + j + "')",end = '')
                count2=1
            else:
                print(",aresta('" + i + "','" + j + "')",end = '')

    print("]).",end = '')

    sys.stdout = old_stdout

    log_file.close()

# Call functions

excelToCSV()
createDataset()
```

```
ruas_list = list(ruas.keys())
createGraph()
```

2.1.2 baseConhecimento.pl (parcial)

```
% Extencao do predicado ponto: Latitude, longitude, ObjectID, Ponto Recolha Local, Contentor Residuos,
↪  Contentor Tipo, Contentor Capacidade, Contentor Quantidade, Contentor Total Litros -> {V,F,D}

% Base de Conhecimento
ponto(-9.14330880914791, 38.7080787857024,1,'15804','Garagem','-',0,0,0).
ponto(-9.14330880914792, 38.7080787857025,355,'15805','Lixos','CV0090',90,1,90).
ponto(-9.14330880914792, 38.7080787857025,356,'15805','Lixos','CV0240',240,7,1680).
ponto(-9.14330880914792, 38.7080787857025,357,'15805','Lixos','CV0090',90,1,90).
ponto(-9.14330880914792, 38.7080787857025,358,'15805','Papel e Cartao','CV0240',240,6,1440).
ponto(-9.14330880914792, 38.7080787857025,359,'15805','Papel e Cartao','CV0090',90,1,90).
ponto(-9.14337777820218, 38.7080781891571,364,'15806','Lixos','CV0240',240,1,240).
ponto(-9.14337777820218, 38.7080781891571,365,'15806','Lixos','CV0140',140,6,840).
ponto(-9.14337777820218, 38.7080781891571,366,'15806','Lixos','CV0120',120,1,120).
ponto(-9.14337777820218, 38.7080781891571,367,'15806','Lixos','CV0090',90,2,180).
ponto(-9.14337777820218, 38.7080781891571,368,'15806','Lixos','CV0240',240,1,240).
ponto(-9.14337777820218, 38.7080781891571,369,'15806','Lixos','CV0140',140,6,840).
ponto(-9.14337777820218, 38.7080781891571,370,'15806','Lixos','CV0120',120,1,120).

(...)
```

2.1.3 grafo.pl (parcial)

```
g(grafo(['15804','15805','15806','15807','15808','15809','15810','15811','15812','15813','15814','15815','15818','15819','15820','15821',
'15822','15823','15824','15825','15827','15828','15830','15831','15832','15833','15834','15836','15838','15841','15842','15843','15844',
'15845','15846','15847','15848','15849','15850','15851','15852','15853','15855','15856','15857','15858','15859','15860','15861','15862',
'15863','15864','15865','15866','15867','15868','15869','15871','15873','15875','15876','15877','15878','15879','15880','15881','15882',
'15883','15884','15885','15886','15887','15888','15889','15890','15891','15892','15893','15894','15896','15897','15898','15899','19365',
'19216','19280','19301','19315','19300','19407','19282','19257','19293','19310','19319','19236','19295','19371','19390','19278','19415',
'19337','21843','21844','21957','21959','21911','21952','21969','21854','21938','21961','21866','21966','21967','21944','21932','21889',
'21925','21927','21939','21949','21950'],[aresta('15804','15828'),aresta('15804','15838'),aresta('15804','15862'),aresta('15804','15844'),
aresta('15804','21844'),aresta('15804','19310'),aresta('15804','15864'),aresta('15805','19315'),aresta('15805','21969'),
aresta('15805','15809'),aresta('15805','15864'),aresta('15805','15828'),aresta('15805','15886'),aresta('15806','19257'),
aresta('15806','15869'),aresta('15807','19371'),aresta('15807','15851'),aresta('15807','19337'),aresta('15807','15815'),
aresta('15807','15897'),aresta('15807','21925'),aresta('15808','19300'),aresta('15808','19415'),aresta('15808','15878'),
aresta('15808','21938'),aresta('15808','15812'),aresta('15808','15843'),aresta('15809','15884'),aresta('15809','15805'),
aresta('15809','19415'),aresta('15809','19301'),aresta('15809','15811'),aresta('15809','15860'),aresta('15809','15841'),
aresta('15809','15825'),aresta('15810','15828'),aresta('15810','15851'),aresta('15811','21969'),aresta('15811','19407'),
aresta('15811','15856'),aresta('15811','15809'),aresta('15811','15812'),aresta('15811','15831'),aresta('15811','15827'),
aresta('15812','15818'),aresta('15812','19407'),aresta('15812','15811'),aresta('15812','15885'),aresta('15812','15808'),
aresta('15812','15824'),aresta('15813','19315')
(...)
```

2.2 Estratégias de procura

Durante a elaboração do projeto foi procurado implementar as diversas formas de procura sugeridas no enunciado, passando por estratégias de pesquisa **não informada** (*Depth First Search*, *Breadth First Search*, Busca Iterativa Limitada em Profundidade) e **informada** (Gulosa, A*).

Estratégia	Tempo (s)	Espaço	Encontra a melhor solução?
DFS	$O(b^m)$	O(bm) linear	Não, devolve a primeira solução que encontra.
BFS	$O(b^d)$	$O(b^d)$	Sim, se o custo de cada passo for 1.
Gulosa	$O(b^m)$	$O(b^m)$	Não encontra sempre a solução ótima.
A* (A estrela)	Exponencial	Todos os nós em memória	Encontra a melhor solução.

2.3 Elaboração do caso prático

Seguindo o enunciado, a elaboração do caso prático deve considerar que um circuito se devide em:

- Inicial – Percurso entre a garagem e o primeiro ponto de recolha;
- Ponto de Recolha - Remoção de resíduos num local de paragem;
- Entre Pontos – Percurso entre dois pontos de Recolha;
- Transporte – Percurso entre o último ponto de Recolha e o local de deposição de resíduos;
- Final – Percurso entre o local de deposição e a garagem.

Para apoiar o desenvolvimento das vertentes pedidas, foram desenvolvidas numerosos predicados que auxiliam a criação dos vários casos pedidos, tanto relativos diretamente ao caso prático em estudo, como outro tipo de funções auxiliares necessárias para uma correta implementação de predicados em *Prolog*, como por exemplo no que toca ao tratamento de listas.

Estas funções encontram-se anexadas em A.4 e serão referidas ao longo da apresentação dos vários predicados do ficheiro principal (*indiv.pl*,A.3).

Alguns pontos devem ser destacados na lógica global do meu projeto, continuando a explicação iniciada na secção de Tratamento de Dados (2.1) e no início desta secção.

Foi decidido que a garagem tem ligação apenas a alguns nodos, mas que todos os nodos têm ligação ao local de deposição e que este volta a ter a ligação à garagem. Assim, no meu caso prático e nos meus testes, costumo considerar que a recolha de lixo começa da garagem e faz o seu circuito, nas várias circunstâncias, até um ponto de deposição.

Gerar circuitos de recolha

Foram criados alguns predicados com vista a gerar, caso existam, circuitos de recolha que cubram um determinado território.

O predicado **caminho**, apresentado de seguida, calcula um caminho entre dois nodos.

```
caminho(G,X,Y,C):-
    caminhoAux(G,X,[Y],C).


caminhoAux(_,X,[X|T],[X|T]).
caminhoAux(G,X,[Y|T],P) :-
    adjacente(Prox_nodo,Y,G), nao(membro(Prox_nodo,[Y|T])),
    ↪  caminhoAux(G,X,[Prox_nodo,Y|T],P).


% Auxiliar adjacente
% Verifica se dois nodos sao adjacente
adjacente(X,Y,grafo(_,L_arestas)) :- member(aresta(X,Y),L_arestas).
```

Foi criado ainda uma variância do predicado anterior que calcula um caminho e a distância percorrida.

```
caminhoK(G,A,B,P,D) :-
  caminhoAuxK(G,A,[B],P,D).
```

```
caminhoAuxK(G,A,[A|P1],[A|P],0,[]).

caminhoAuxK(G,A,[Y|P1],P,D1) :-
  adjacente(X,Y,Di,G),
  nao(membro(X,[Y|P1])),
  distanciaEntreNodos(X,Y,Di),
  caminhoAuxK(G,A,[X,Y|P1],P,K),
  D1 is D + Di.

% Auxiliar distanciaEntreNodos
distanciaEntreNodos(X,Y,D) :- pontosID(X,Ps), Ps = [P|T], P =
 ↪  ponto(Lat1,Lon1,_,X,_,_,_,_,_),
                              pontosID(Y,Ps2), Ps2 = [P2|T2], P2 =
                               ↪  ponto(Lat2,Lon2,_,Y,_,_,_,_,_),
                              distancia(Lat1,Lon1,Lat2,Lon2,D).
```

Para encontrar circuitos com a estratégia ***Depth First*** foi criado o predicado **dFirst**.

Muitas vezes seguem em comentário alguns testes ao predicado em funcionamento.

```
dFirst(G,X,Y,R) :- dFirst(G,X,Y,[X],R).

dFirst(G,X,Y,V,[aresta(X,Y)]) :- adjacente(X,Y,G).

dFirst(G,X,Y,V,[aresta(X,Z)|R]) :-
 adjacente(X,Z,G),
 \+ membrochk(aresta(X,Z),V),
 \+ membro(Z,V),
 dFirst(G,Z,Y,[Z|V],R),
 Z \= Y.

% T
test_dFirst(R) :- g(G), getIdGaragem(Gar), getIdDeposicao(D),
```

```
dfFirst(G,Gar,D,R).
```

Na mesma lógica foi feita a implementação de um predicado ***bFirst***, que encontra o circuito numa pesquisa em largura.

```
bFirst(G,X,Y,Visited) :- bFirstAux(G,Y,Successors,
                []),
                RevVisited),
                sucessor(G,X,Successors),
                inverso(RevVisited, Visited).

bFirstAux(G,Y,[], History, []).

bFirstAux(G,Y,[aresta(P,Y)|_], History, [aresta(P,Y)|History]).

bFirstAux(G,Y,[aresta(X,Z)|RestQ], History, RevVisited) :-
                            sucessor(G,Z,Cats),
                                                    addElement(RestQ, Cats,
                                                    ↪  Queue),
                                                    bFirstAux(G,Y, Queue, [aresta(X,Z)|History],
                                                    ↪  RevVisited).
%Auxiliar sucessor
% Obter sucessores de um nodo
sucessor(grafo(_,Es),X,R):- solucoes(aresta(X,Y),membro(aresta(X,Y),Es),R).

%Auxiliar inverso
inverso(Xs, Ys):-
        inverso(Xs, [], Ys).

inverso([], Xs, Xs).
inverso([X|Xs],Ys, Zs):-
        inverso(Xs, [X|Ys], Zs).


%Auxiliar addElement
addElement(X, [], [X]).
addElement(X, [Y | Rest], [X,Y | Rest]) :- X @< Y, !.
addElement(X, [Y | Rest1], [Y | Rest2]) :- addElement(X, Rest1, Rest2).
```

Foi também coberto o caso de uma pesquisa seletiva, ou seja, em que é suposto o circuito percorrer apenas pontos que tenham um determinado tipo de resíduo.

Para este fim, desenvolvi primeiramente um predicado **pontoTipo** que, dado como argumento um tipo de resíduo, devolve os pontos de um nodo que têm esse tipo de resíduo,

caso existam.

```
pontoTipo(aresta(X,Y),T) :- ponto(_,_,_,X,Tp,_,_,_,Ctotal),
 ↪  ponto(_,_,_,Y,Tp,_,_,_,Ctotal).
```

Com este predicado, prossegui ao desenvolvimento da estratégia *depthFirst* para circuitos de recolha seletivos perante um tipo específico de resíduos. Foi assim desenvolvido o predicado **dFTipo**, apresentado de seguida.

```
dFTipo(G,T,X,Y,P) :- dFTipo(G,T,X,Y,[X],P).


dFTipo(G,T,X,Y,V,[aresta(X,Y)]) :- adjacente(X,Y,G),
                        pontoTipo(aresta(X,Y),T).


dFTipo(G,T,X,Y,V,[aresta(X,Z)|P]) :-
 adjacente(X,Z,G),
 pontoTipo(aresta(X,Z),T),
 \+ membrochk(aresta(X,Z),V),
 \+ membro(Z,V),
 dFTipo(G,T,Z,Y,[Z|V],P).
```

Identificar circuitos com mais pontos de recolha (por tipo de resíduo)

Para esta funcionalidade, criei o predicado **nrPontosRecolhaT**, que calcula o número de pontos de recolha de um determinado tipo num circuito.

```
nrPontosRecolhaT([],T,Count,R).
nrPontosRecolhaT([A],T,Count,R) :- edgePointsTipo(A,Tp,S),
                        length(S,L),
                        Count2 is Count + L,
                        nrPontosRecolhaT([],Count2,R).
nrPontosRecolhaT([H|T],Count,T,R) :- edgePointsTipo(H,Tp,S),
                        length(S,L),
```

```
                        Count2 is Count + L,
                        nrPontosRecolhaT(T,Count2,R).

% T
teste_nrPontosRecolha(R) :-
 ↪  edgePointsTipo(aresta('15875','21844'),'Lixos',0,S),

% Aux
edgePointsTipo(aresta(X,_),Tp,R):-solucoes(ponto(Lat,Lon,O,X,Tp,Ct,Ccap,Cqt,Ctotal),
    ponto(Lat,Lon,O,X,Tp,Ct,Ccap,Cqt,Ctotal),R).
```

Escolher o circuito mais rápido (critério de distância

Para obter qual o circuito mais rápido, ou seja, o circuito em que se percorre a menor distância, recorri à estratégia *depth first* e criei o método **dFDistancia**, que concebe os percursos juntamente com a análise da distância percorrida.

Com este predicado, foi possível desenvolver o **maisRapido**, que indica qual o circuito que envolve uma menor distância para percorrer.

```
dFDistancia(G,X,Y,R) :- dFDistancia(G,X,Y,[X],R,Km).

dFDistancia(G,X,Y,V,[aresta(X,Y)],0) :- adjacente(X,Y,G).

dFDistancia(G,X,Y,V,[aresta(X,Z)|R],D) :-
 adjacente(X,Z,G),
 Di = distanciaEntreNodos(X,Z),
 \+ membrochk(aresta(X,Z),V),
 \+ membro(Z,V),
 D2 = distanciaEntreNodos(Z,Y),
 dFDistancia(G,Z,Y,[Z|V],R,D2),
 D is D2 + Di.

% Percurso mais rapido (menor distancia)
```

```
maisRapido(G,X,Y,P,D):-solucoes((P,C),(dFDistancia(G,X,Y,P,D),!,C=D),L),
                                        minimo(L,(P,C)), D=C.
```

Comparar circuitos de recolha tendo em conta a distância média percorrida entre pontos de recolha

Distância Média Percorrida

Para saber qual de mais do que um circuito é o mais rápido, primeiramente tive de criar um predicado que calculasse a distância num circuito, ou seja, a soma das distancias entre todos os nodos no circuito.

Esse predicado **calcDist** é apresentado de seguida, juntamente com os predicados auxiliares necessários.

```
calcDist([],Di,Di).
calcDist([A],Di,Di) :- edgePoints(A,A1), [A2|T2]=A1, edgePoints2(A,B), [B1|B2]=B,
                       distanciaEntrePontos(A2,B1,Dist2),
                       ZZ is Dist2 + Di,
                       calcDist([],ZZ,Dist).
calcDist([A|X],Di,Dist):- edgePoints(A,A1), [A2|T2]=A1, edgePoints2(A,B),
 ↪  [B1|B2]=B,
            distanciaEntrePontos(A2,B1,Dist2),
            ZZ is Dist2 + Di,
            calcDist(X,ZZ,Dist).

% -- Auxiliares --
% Calcula a distancia entre dois pontos
distanciaEntrePontos(ponto(Lat1,Lon1,_,X,_,_,_,_,_),
    ponto(Lat2,Lon2,_,Y,_,_,_,_,_),R) :- distancia(Lat1,Lon1,Lat2,Lon2,R).

% Devolve os pontos do primeiro elemento de uma aresta
edgePoints(aresta(X,_),R):- solucoes(ponto(Lat,Lon,O,X,T,Ct,Ccap,Cqt,Ctotal),
    ponto(Lat,Lon,O,X,T,Ct,Ccap,Cqt,Ctotal),R).

% Devolve os pontos do segundo elemento de uma aresta
```

```
edgePoints2(aresta(_,Y),R):- solucoes(ponto(Lat,Lon,0,Y,T,Ct,Ccap,Cqt,Ctotal),
    ponto(Lat,Lon,0,Y,T,Ct,Ccap,Cqt,Ctotal),R).
```

Com este predicado criado, desenvolvi o método para comparar circuitos no que toca à distância média percorrida entre pontos de recolha.

```
compararDistanciaMedia(L,R) :- compararDistanciaMediaAux(L,'0',0).

compararDistanciaMediaAux([],Ci,Dm).
compararDistanciaMediaAux([H],Ci,Dm) :- calcDist(H,0,D),
                                        (D > Dm -> Dm is D, Ci is H),
                                        compararDistanciaMediaAux([],Ci,Dm).
compararDistanciaMediaAux([H|T],Ci,Dm) :- calcDist(H,0,D),
                                        (D > Dm -> Dm is D, Ci is H),
                                        compararDistanciaMediaAux(T,Ci,Dm).
```

Quantidade de resíduos recolhida

A mesma lógica que a anterior foi utilizada para comparar circuitos no que toca à quantidade de resíduos recolhida, recorrendo a outra informação fornecida no *dataset* dado.

```
compararQtRecolhida(L,R) :- compararQtRecolhidaAux(L,'0',0).

compararQtRecolhidaAux([],Sum,R).
compararQtRecolhidaAux([H],Sum,R) :- edgePoints(H,E),
                        somarQuantidadesRecolhidas(H,0,S),
                        Sum2 = Sum + S,
                        compararQtRecolhidaAux([],Sum2,R).
compararQtRecolhidaAux([H|T],Sum,R) :- edgePoints(H,E),
                        somarQuantidadesRecolhidas(H,0,S),
                        Sum2 = Sum + S,
                        compararQtRecolhidaAux(T,Sum2,R).

somarQuantidadesRecolhidas([],Sum,R).
somarQuantidadesRecolhidas([A],Sum,R) :- ponto(Lat,Lon,0,X,T,Ct,Ccap,Cqt,Ctotal) = A,
                        Sum2 = Sum + Ctotal,
                        somarQuantidadesRecolhidas([],Sum2,R).
somarQuantidadesRecolhidas([H|T],Sum,R) :- ponto(Lat,Lon,0,X,T,Ct,Ccap,Cqt,Ctotal) = H,
                        Sum2 = Sum + Ctotal,
                        somarQuantidadesRecolhidas(T,Sum2,R).
```

3

Conclusões

Com a realização deste projeto foi-me possível aplicar diversas funcionalidades e conhecimentos adquiridos nas aulas práticas da Unidade Curricular, especificamente do tema de Métodos de resolução de problemas e procura, desenvolvendo também as minhas capacidades na linguagem *Prolog*.

Foram no entanto enfrentadas diversas dificuldades, especialmente no que tocou à interpretação inicial do *dataset* e do contexto do problema, havendo uma inicial má perceção e decisão da forma como tratar os dados, o que dificultou o resto do desenvolvimento, no que visa à implementação de alguns dos métodos de procura e funcionalidades.

Bibliografia

[1] Cesar Analide, Paulo Novais, José Neves. *Sugestões para a Redacção de Relatórios Técnicos*. 2011.

[2] PROLOG,
`https://pt.wikibooks.org/wiki/Prolog`

[3] SWI-Prolog,
`https://www.swi-prolog.org`

A

Código fonte

A.1 baseConhecimento.pl

Base de Conhecmento gerada.

```
%-----------------------------------------------
% SRCR INDIVIDUAL | Joana Afonso Gomes | a84912
%-----------------------------------------------
:- set_prolog_flag( discontiguous_warnings,off ).
:- set_prolog_flag( single_var_warnings,off ).
:- set_prolog_flag( unknown,fail ).
%-----------------------------------------------
:- op( 900,xfy,'::' ).
:- dynamic '-'/1.
:- dynamic(ponto/9).
%-----------------------------------------------

% Extencao do predicado ponto: Latitude, longitude, ObjectID, Ponto Recolha Local, Contentor Residuos, Contentor Tipo, Contentor
↪  Capacidade, Contentor Quantidade, Contentor Total Litros -> {V,F,D}

% Base de Conhecimento
ponto(-9.14330880914791, 38.7080787857024,1,'15804','Garagem','-',0,0,0).
ponto(-9.14330880914792, 38.7080787857025,355,'15805','Lixos','CV0090',90,1,90).
ponto(-9.14330880914792, 38.7080787857025,356,'15805','Lixos','CV0240',240,7,1680).
ponto(-9.14330880914792, 38.7080787857025,357,'15805','Lixos','CV0090',90,1,90).
ponto(-9.14330880914792, 38.7080787857025,358,'15805','Papel e Cartao','CV0240',240,6,1440).
ponto(-9.14330880914792, 38.7080787857025,359,'15805','Papel e Cartao','CV0090',90,1,90).
ponto(-9.14337777820218, 38.7080781891571,364,'15806','Lixos','CV0240',240,1,240).
ponto(-9.14337777820218, 38.7080781891571,365,'15806','Lixos','CV0140',140,6,840).
ponto(-9.14337777820218, 38.7080781891571,366,'15806','Lixos','CV0120',120,1,120).
ponto(-9.14337777820218, 38.7080781891571,367,'15806','Lixos','CV0090',90,2,180).
ponto(-9.14337777820218, 38.7080781891571,368,'15806','Lixos','CV0240',240,1,240).
ponto(-9.14337777820218, 38.7080781891571,369,'15806','Lixos','CV0140',140,6,840).
ponto(-9.14337777820218, 38.7080781891571,370,'15806','Lixos','CV0120',120,1,120).
ponto(-9.14337777820218, 38.7080781891571,360,'15806','Lixos','CV0090',90,2,180).
ponto(-9.14337777820218, 38.7080781891571,361,'15806','Papel e Cartao','CV0140',140,2,280).
ponto(-9.14337777820218, 38.7080781891571,362,'15806','Papel e Cartao','CV0090',90,1,90).
```

```
ponto(-9.14337777820218, 38.7080781891571,363,'15806','Papel e Cartao','CA0090',90,1,90).
ponto(-9.14348180670535, 38.7073026157039,371,'15807','Lixos','CV0240',240,1,240).
ponto(-9.14348180670535, 38.7073026157039,372,'15807','Lixos','CV0140',140,3,420).
ponto(-9.14348180670535, 38.7073026157039,373,'15807','Lixos','CV0240',240,1,240).
ponto(-9.14348180670535, 38.7073026157039,374,'15807','Lixos','CV0140',140,3,420).
ponto(-9.14348180670535, 38.7073026157039,375,'15807','Papel e Cartao','CV0240',240,1,240).
ponto(-9.14348180670535, 38.7073026157039,376,'15807','Papel e Cartao','CV0140',140,1,140).
ponto(-9.14255098678099, 38.7073286838222,377,'15808','Lixos','CV0140',140,2,280).
ponto(-9.14255098678099, 38.7073286838222,378,'15808','Lixos','CV0240',240,4,960).
ponto(-9.14255098678099, 38.7073286838222,379,'15808','Lixos','CV0140',140,2,280).
ponto(-9.14255098678099, 38.7073286838222,380,'15808','Lixos','CV0240',240,4,960).
ponto(-9.14276596081499, 38.7070836135523,381,'15809','Lixos','CV0140',140,2,280).
ponto(-9.14276596081499, 38.7070836135523,382,'15809','Lixos','CV0120',120,1,120).
ponto(-9.1426225690344, 38.7066975168166,383,'15810','Lixos','CV0240',240,4,960).
ponto(-9.1426225690344, 38.7066975168166,384,'15810','Lixos','CV0240',240,4,960).
ponto(-9.1426225690344, 38.7066975168166,385,'15810','Papel e Cartao','CV0240',240,1,240).
ponto(-9.14240835587344, 38.7069966274245,386,'15811','Lixos','CV0140',140,3,420).
ponto(-9.14240835587344, 38.7069966274245,387,'15811','Lixos','CV0240',240,4,960).
ponto(-9.14240835587344, 38.7069966274245,388,'15811','Lixos','CV0140',140,3,420).
ponto(-9.14240835587344, 38.7069966274245,389,'15811','Lixos','CV0240',240,4,960).
ponto(-9.14240835587344, 38.7069966274245,390,'15811','Papel e Cartao','CV0240',240,1,240).
ponto(-9.14305015543156, 38.7068559589223,391,'15812','Lixos','CV0140',140,5,700).
ponto(-9.14305015543156, 38.7068559589223,392,'15812','Lixos','CV0240',240,3,720).
ponto(-9.14305015543156, 38.7068559589223,393,'15812','Lixos','CV0120',120,3,360).
ponto(-9.14305015543156, 38.7068559589223,394,'15812','Lixos','CV0140',140,5,700).
ponto(-9.14305015543156, 38.7068559589223,395,'15812','Lixos','CV0240',240,3,720).
ponto(-9.14305015543156, 38.7068559589223,396,'15812','Lixos','CV0120',120,3,360).
ponto(-9.1512511235953, 38.7087754470349,397,'15813','Lixos','CV0240',240,10,2400).
ponto(-9.1512511235953, 38.7087754470349,398,'15813','Lixos','CV0140',140,7,980).
ponto(-9.1512511235953, 38.7087754470349,399,'15813','Lixos','CV0090',90,1,90).
ponto(-9.1512511235953, 38.7087754470349,400,'15813','Lixos','CV0240',240,10,2400).
ponto(-9.1512511235953, 38.7087754470349,401,'15813','Lixos','CV0140',140,7,980).
ponto(-9.1512511235953, 38.7087754470349,402,'15813','Lixos','CV0090',90,1,90).
ponto(-9.15178946418214, 38.7086356323308,406,'15814','Lixos','CV0120',120,2,240).
ponto(-9.15178946418214, 38.7086356323308,407,'15814','Lixos','CV0140',140,5,700).
ponto(-9.15178946418214, 38.7086356323308,408,'15814','Lixos','CV0090',90,1,90).
ponto(-9.15178946418214, 38.7086356323308,409,'15814','Lixos','CV0240',240,2,480).
ponto(-9.15178946418214, 38.7086356323308,410,'15814','Lixos','CV0120',120,2,240).
ponto(-9.15178946418214, 38.7086356323308,403,'15814','Lixos','CV0140',140,5,700).
ponto(-9.15178946418214, 38.7086356323308,404,'15814','Lixos','CV0090',90,1,90).
ponto(-9.15178946418214, 38.7086356323308,405,'15814','Lixos','CV0240',240,2,480).
ponto(-9.15201897924565, 38.708606605818,411,'15815','Lixos','CV0120',120,1,120).
ponto(-9.15201897924565, 38.708606605818,412,'15815','Lixos','CV0140',140,2,280).
ponto(-9.15201897924565, 38.708606605818,413,'15815','Lixos','CV0240',240,1,240).
ponto(-9.15201897924565, 38.708606605818,414,'15815','Lixos','CV0120',120,1,120).
ponto(-9.15201897924565, 38.708606605818,415,'15815','Lixos','CV0140',140,2,280).
ponto(-9.15201897924565, 38.708606605818,416,'15815','Lixos','CV0240',240,1,240).
ponto(-9.14910552718357, 38.709073383915,417,'15818','Lixos','CV0240',240,1,240).
ponto(-9.14764976145157, 38.708563591768,418,'15819','Lixos','CV0140',140,9,1260).
ponto(-9.14764976145157, 38.708563591768,419,'15819','Lixos','CV0120',120,2,240).
ponto(-9.14764976145157, 38.708563591768,420,'15819','Lixos','CV0240',240,3,720).
ponto(-9.14764976145157, 38.708563591768,421,'15819','Lixos','CV0090',90,2,180).
ponto(-9.14764976145157, 38.708563591768,422,'15819','Lixos','CV1100',1100,1,1100).
ponto(-9.1485717796828, 38.7087267228466,423,'15820','Lixos','CV0140',140,6,840).
ponto(-9.1485717796828, 38.7087267228466,424,'15820','Lixos','CV0240',240,2,480).
ponto(-9.1485717796828, 38.7087267228466,425,'15820','Lixos','CV0090',90,1,90).
ponto(-9.14911344583279, 38.7088210959641,426,'15821','Lixos','CV0140',140,2,280).
ponto(-9.14911344583279, 38.7088210959641,427,'15821','Lixos','CV0240',240,2,480).
ponto(-9.14911344583279, 38.7088210959641,428,'15821','Lixos','CV0090',90,1,90).
ponto(-9.14949354638571, 38.7088718347208,429,'15822','Lixos','CV0090',90,2,180).
ponto(-9.14949354638571, 38.7088718347208,430,'15822','Lixos','CV0120',120,1,120).
ponto(-9.14949354638571, 38.7088718347208,431,'15822','Lixos','CV0140',140,1,140).
ponto(-9.14949354638571, 38.7088718347208,432,'15822','Lixos','CV0240',240,3,720).
ponto(-9.15018388371526, 38.7089108606806,433,'15823','Lixos','CV0140',140,12,1680).
ponto(-9.15018388371526, 38.7089108606806,434,'15823','Lixos','CV0090',90,3,270).
```

```
ponto(-9.15018388371526, 38.7089108606806,435,'15823','Lixos','CV0240',240,13,3120).
ponto(-9.15018388371526, 38.7089108606806,436,'15823','Lixos','CV0120',120,1,120).
ponto(-9.15079311664937, 38.7089055507175,437,'15824','Lixos','CV0140',140,2,280).
ponto(-9.15079311664937, 38.7089055507175,438,'15824','Lixos','CV0120',120,1,120).
ponto(-9.15079311664937, 38.7089055507175,439,'15824','Lixos','CV0090',90,1,90).
ponto(-9.15206034846112, 38.7082819634324,440,'15825','Lixos','CV0140',140,1,140).
ponto(-9.15247211372946, 38.7081342433687,441,'15827','Lixos','CV0240',240,8,1920).
ponto(-9.15247211372946, 38.7081342433687,442,'15827','Lixos','CV0120',120,1,120).
ponto(-9.15178357572543, 38.7082213241484,443,'15828','Lixos','CV0090',90,1,90).
ponto(-9.15178357572543, 38.7082213241484,444,'15828','Lixos','CV0120',120,1,120).
ponto(-9.15178357572543, 38.7082213241484,445,'15828','Lixos','CV0140',140,3,420).
ponto(-9.15178357572543, 38.7082213241484,446,'15828','Lixos','CV0240',240,2,480).
ponto(-9.15154474333579, 38.7084035643017,447,'15830','Lixos','CV0240',240,5,1200).
ponto(-9.1513885281045, 38.7079275125773,448,'15831','Lixos','CV0120',120,2,240).
ponto(-9.1513885281045, 38.7079275125773,449,'15831','Lixos','CV0090',90,1,90).
ponto(-9.1513885281045, 38.7079275125773,450,'15831','Lixos','CV0140',140,2,280).
ponto(-9.1513885281045, 38.7079275125773,451,'15831','Lixos','CV0240',240,3,720).
ponto(-9.14751394875702, 38.7103393147562,452,'15832','Lixos','CV0240',240,1,240).
ponto(-9.14751394875702, 38.7103393147562,453,'15832','Lixos','CV0240',240,1,240).
ponto(-9.14751394875702, 38.7103393147562,454,'15832','Vidro','CV0240',240,1,240).
ponto(-9.14856601878688, 38.7099428303287,455,'15833','Lixos','CV0240',240,3,720).
ponto(-9.14856601878688, 38.7099428303287,456,'15833','Lixos','CV0240',240,3,720).
ponto(-9.15046432876378, 38.7068005850863,457,'15834','Lixos','CV0140',140,1,140).
ponto(-9.15046432876378, 38.7068005850863,458,'15834','Lixos','CV0240',240,8,1920).
ponto(-9.1505486401448, 38.7078807891527,459,'15836','Lixos','CV0240',240,23,5520).
ponto(-9.1505486401448, 38.7078807891527,460,'15836','Lixos','CV0140',140,1,140).
ponto(-9.14851812420916, 38.7098081294277,461,'15838','Lixos','CV0240',240,4,960).
ponto(-9.14851812420916, 38.7098081294277,462,'15838','Lixos','CV0240',240,4,960).
ponto(-9.14704425203433, 38.7080193757895,463,'15841','Lixos','CV0240',240,9,2160).
ponto(-9.14704425203433, 38.7080193757895,464,'15841','Lixos','CV0140',140,5,700).
ponto(-9.14704425203433, 38.7080193757895,465,'15841','Lixos','CV0240',240,9,2160).
ponto(-9.14704425203433, 38.7080193757895,466,'15841','Lixos','CV0140',140,5,700).
ponto(-9.14614421827306, 38.707783978131,467,'15842','Lixos','CV0090',90,1,90).
ponto(-9.14614421827306, 38.707783978131,468,'15842','Lixos','CV0090',90,1,90).
ponto(-9.14563984658037, 38.7078874408968,469,'15843','Lixos','CV0090',90,1,90).
ponto(-9.14563984658037, 38.7078874408968,470,'15843','Lixos','CV0140',140,4,560).
ponto(-9.14563984658037, 38.7078874408968,471,'15843','Lixos','CV0240',240,2,480).
ponto(-9.14563984658037, 38.7078874408968,472,'15843','Papel e Cartao','CV0090',90,1,90).
ponto(-9.14563984658037, 38.7078874408968,473,'15843','Papel e Cartao','CV0140',140,2,280).
ponto(-9.14598863635406, 38.7081636570213,474,'15844','Lixos','CV0140',140,1,140).
ponto(-9.14598863635406, 38.7081636570213,475,'15844','Lixos','CV0140',140,1,140).
ponto(-9.15199352638039, 38.7076249751805,476,'15845','Lixos','CV0120',120,1,120).
ponto(-9.15199352638039, 38.7076249751805,477,'15845','Lixos','CV0240',240,20,4800).
ponto(-9.15093562538604, 38.7076071810499,478,'15846','Lixos','CV0240',240,5,1200).
ponto(-9.14993558128589, 38.7076158965573,479,'15847','Lixos','CV0240',240,18,4320).
ponto(-9.14993558128589, 38.7076158965573,480,'15847','Lixos','CV0120',120,1,120).
ponto(-9.14829055489657, 38.7075401361692,481,'15848','Lixos','CV0240',240,13,3120).
ponto(-9.14829055489657, 38.7075401361692,482,'15848','Lixos','CV1100',1100,4,4400).
ponto(-9.14829055489657, 38.7075401361692,483,'15848','Lixos','CV0120',120,1,120).
ponto(-9.14829055489657, 38.7075401361692,484,'15848','Lixos','CV0090',90,1,90).
ponto(-9.14714620183592, 38.707099690356,485,'15849','Lixos','CV0240',240,4,960).
ponto(-9.14835399212529, 38.7103950657776,486,'15850','Lixos','CV0240',240,3,720).
ponto(-9.14835399212529, 38.7103950657776,487,'15850','Lixos','CV0240',240,3,720).
ponto(-9.14835399212529, 38.7103950657776,488,'15850','Embalagens','CV0240',240,1,240).
ponto(-9.14671590072996, 38.7075628281927,489,'15851','Lixos','CV0240',240,1,240).
ponto(-9.14671590072996, 38.7075628281927,490,'15851','Lixos','CV0140',140,2,280).
ponto(-9.14671590072996, 38.7075628281927,491,'15851','Papel e Cartao','CV0140',140,2,280).
ponto(-9.14493001491084, 38.7072810634258,494,'15852','Lixos','CV0140',140,1,140).
ponto(-9.14493001491084, 38.7072810634258,495,'15852','Lixos','CV0120',120,1,120).
ponto(-9.14493001491084, 38.7072810634258,496,'15852','Lixos','CV0090',90,1,90).
ponto(-9.14493001491084, 38.7072810634258,492,'15852','Lixos','CV0240',240,1,240).
ponto(-9.14493001491084, 38.7072810634258,493,'15852','Papel e Cartao','CV0240',240,1,240).
ponto(-9.14585366538093, 38.7075613035211,497,'15853','Lixos','CV0120',120,1,120).
ponto(-9.14585366538093, 38.7075613035211,498,'15853','Lixos','CV0140',140,3,420).
ponto(-9.14585366538093, 38.7075613035211,499,'15853','Lixos','CV0090',90,1,90).
```

```
ponto(-9.14585366538093, 38.7075613035211,500,'15853','Lixos','CV0240',240,2,480).
ponto(-9.14585366538093, 38.7075613035211,501,'15853','Papel e Cartao','CV0340',340,2,680).
ponto(-9.14585366538093, 38.7075613035211,502,'15853','Papel e Cartao','CV0090',90,1,90).
ponto(-9.14585366538093, 38.7075613035211,503,'15853','Papel e Cartao','CV0240',240,3,720).
ponto(-9.14350797013798, 38.7075275840864,504,'15855','Lixos','CV0240',240,3,720).
ponto(-9.14350797013798, 38.7075275840864,505,'15855','Lixos','CV0120',120,1,120).
ponto(-9.14350797013798, 38.7075275840864,506,'15855','Lixos','CV0140',140,5,700).
ponto(-9.14350797013798, 38.7075275840864,507,'15855','Lixos','CV0240',240,3,720).
ponto(-9.14350797013798, 38.7075275840864,508,'15855','Lixos','CV0120',120,1,120).
ponto(-9.14350797013798, 38.7075275840864,509,'15855','Lixos','CV0140',140,5,700).
ponto(-9.14415472749814, 38.7077381729183,510,'15856','Lixos','CV0240',240,4,960).
ponto(-9.14415472749814, 38.7077381729183,511,'15856','Lixos','CV0140',140,2,280).
ponto(-9.14415472749814, 38.7077381729183,512,'15856','Lixos','CV0120',120,1,120).
ponto(-9.14415472749814, 38.7077381729183,513,'15856','Lixos','CV0240',240,4,960).
ponto(-9.14415472749814, 38.7077381729183,514,'15856','Lixos','CV0140',140,2,280).
ponto(-9.14415472749814, 38.7077381729183,515,'15856','Lixos','CV0120',120,1,120).
ponto(-9.14418870357883, 38.707701846559,516,'15857','Lixos','CV0240',240,1,240).
ponto(-9.14418870357883, 38.707701846559,517,'15857','Lixos','CV0140',140,1,140).
ponto(-9.14418870357883, 38.707701846559,518,'15857','Lixos','CV0240',240,1,240).
ponto(-9.14418870357883, 38.707701846559,519,'15857','Lixos','CV0140',140,1,140).
ponto(-9.14418870357883, 38.707701846559,520,'15857','Papel e Cartao','CV0140',140,1,140).
ponto(-9.14351882987146, 38.7074824516687,521,'15858','Lixos','CV0090',90,1,90).
ponto(-9.14351882987146, 38.7074824516687,522,'15858','Lixos','CV0140',140,3,420).
ponto(-9.14351882987146, 38.7074824516687,523,'15858','Lixos','CV0240',240,1,240).
ponto(-9.14351882987146, 38.7074824516687,524,'15858','Lixos','CV0090',90,1,90).
ponto(-9.14351882987146, 38.7074824516687,525,'15858','Lixos','CV0140',140,3,420).
ponto(-9.14351882987146, 38.7074824516687,526,'15858','Lixos','CV0240',240,1,240).
ponto(-9.14475474461576, 38.7078951146912,527,'15859','Lixos','CV0090',90,3,270).
ponto(-9.14475474461576, 38.7078951146912,528,'15859','Lixos','CV0140',140,5,700).
ponto(-9.14475474461576, 38.7078951146912,529,'15859','Lixos','CV0240',240,2,480).
ponto(-9.14475474461576, 38.7078951146912,530,'15859','Lixos','CV0090',90,3,270).
ponto(-9.14475474461576, 38.7078951146912,531,'15859','Lixos','CV0140',140,5,700).
ponto(-9.14475474461576, 38.7078951146912,532,'15859','Lixos','CV0240',240,2,480).
ponto(-9.14475474461576, 38.7078951146912,533,'15859','Papel e Cartao','CV0140',140,1,140).
ponto(-9.14552858843579, 38.7081496329634,534,'15860','Lixos','CV0090',90,3,270).
ponto(-9.14552858843579, 38.7081496329634,535,'15860','Lixos','CV0240',240,3,720).
ponto(-9.14552858843579, 38.7081496329634,536,'15860','Lixos','CV0140',140,11,1540).
ponto(-9.14552858843579, 38.7081496329634,537,'15860','Lixos','CV0120',120,2,240).
ponto(-9.14552858843579, 38.7081496329634,538,'15860','Lixos','CV0090',90,3,270).
ponto(-9.14552858843579, 38.7081496329634,539,'15860','Lixos','CV0240',240,3,720).
ponto(-9.14552858843579, 38.7081496329634,540,'15860','Lixos','CV0140',140,11,1540).
ponto(-9.14552858843579, 38.7081496329634,541,'15860','Lixos','CV0120',120,2,240).
ponto(-9.14552858843579, 38.7081496329634,542,'15860','Papel e Cartao','CV0240',240,2,480).
ponto(-9.14606015123382, 38.7083431936003,543,'15861','Lixos','CV0140',140,5,700).
ponto(-9.14606015123382, 38.7083431936003,544,'15861','Lixos','CV0090',90,2,180).
ponto(-9.14606015123382, 38.7083431936003,545,'15861','Lixos','CV0120',120,1,120).
ponto(-9.14606015123382, 38.7083431936003,546,'15861','Lixos','CV0140',140,5,700).
ponto(-9.14606015123382, 38.7083431936003,547,'15861','Lixos','CV0090',90,2,180).
ponto(-9.14606015123382, 38.7083431936003,548,'15861','Lixos','CV0120',120,1,120).
ponto(-9.14634879700261, 38.7084307669213,549,'15862','Lixos','CV0090',90,1,90).
ponto(-9.14634879700261, 38.7084307669213,550,'15862','Lixos','CV0140',140,2,280).
ponto(-9.14634879700261, 38.7084307669213,551,'15862','Lixos','CV0090',90,1,90).
ponto(-9.14634879700261, 38.7084307669213,552,'15862','Lixos','CV0140',140,2,280).
ponto(-9.14664843006564, 38.7084822125522,553,'15863','Lixos','CV0240',240,1,240).
ponto(-9.14664843006564, 38.7084822125522,554,'15863','Lixos','CV0140',140,1,140).
ponto(-9.14664843006564, 38.7084822125522,555,'15863','Lixos','CV0240',240,1,240).
ponto(-9.14664843006564, 38.7084822125522,556,'15863','Lixos','CV0140',140,1,140).
ponto(-9.14695866414731, 38.7084705102342,563,'15864','Lixos','CV0240',240,4,960).
ponto(-9.14695866414731, 38.7084705102342,564,'15864','Lixos','CV0140',140,4,560).
ponto(-9.14695866414731, 38.7084705102342,557,'15864','Lixos','CV0090',90,1,90).
ponto(-9.14695866414731, 38.7084705102342,558,'15864','Lixos','CV0120',120,1,120).
ponto(-9.14695866414731, 38.7084705102342,559,'15864','Lixos','CV0240',240,4,960).
ponto(-9.14695866414731, 38.7084705102342,560,'15864','Lixos','CV0140',140,4,560).
ponto(-9.14695866414731, 38.7084705102342,561,'15864','Lixos','CV0090',90,1,90).
ponto(-9.14695866414731, 38.7084705102342,562,'15864','Organicos','CV0120',120,1,120).
```

```
ponto(-9.14428821714939, 38.7074217423076,565,'15865','Lixos','CV0120',120,1,120).
ponto(-9.14428821714939, 38.7074217423076,566,'15865','Lixos','CV0140',140,2,280).
ponto(-9.14428821714939, 38.7074217423076,567,'15865','Lixos','CV0240',240,4,960).
ponto(-9.14428821714939, 38.7074217423076,568,'15865','Lixos','CV0120',120,1,120).
ponto(-9.14428821714939, 38.7074217423076,569,'15865','Lixos','CV0140',140,2,280).
ponto(-9.14428821714939, 38.7074217423076,570,'15865','Lixos','CV0240',240,4,960).
ponto(-9.14428821714939, 38.7074217423076,571,'15865','Papel e Cartao','CV0140',140,1,140).
ponto(-9.14428821714939, 38.7074217423076,572,'15865','Papel e Cartao','CV0240',240,1,240).
ponto(-9.14381451843206, 38.7072546959625,573,'15866','Lixos','CV0140',140,1,140).
ponto(-9.14381451843206, 38.7072546959625,574,'15866','Lixos','CV0240',240,11,2640).
ponto(-9.14381451843206, 38.7072546959625,575,'15866','Lixos','CV0140',140,1,140).
ponto(-9.14381451843206, 38.7072546959625,576,'15866','Lixos','CV0240',240,11,2640).
ponto(-9.14381451843206, 38.7072546959625,577,'15866','Papel e Cartao','CV0240',240,2,480).
ponto(-9.14319266057494, 38.7071790073949,578,'15867','Lixos','CV0090',90,2,180).
ponto(-9.14319266057494, 38.7071790073949,579,'15867','Lixos','CV0240',240,3,720).
ponto(-9.14319266057494, 38.7071790073949,580,'15867','Lixos','CV0140',140,3,420).
ponto(-9.14319266057494, 38.7071790073949,581,'15867','Lixos','CV0090',90,2,180).
ponto(-9.14319266057494, 38.7071790073949,582,'15867','Lixos','CV0240',240,3,720).
ponto(-9.14319266057494, 38.7071790073949,583,'15867','Lixos','CV0140',140,3,420).
ponto(-9.14402783597271, 38.7068925345681,584,'15868','Lixos','CV0090',90,1,90).
ponto(-9.14402783597271, 38.7068925345681,585,'15868','Lixos','CV0240',240,5,1200).
ponto(-9.14402783597271, 38.7068925345681,586,'15868','Lixos','CV0140',140,5,700).
ponto(-9.14402783597271, 38.7068925345681,587,'15868','Lixos','CV0090',90,1,90).
ponto(-9.14402783597271, 38.7068925345681,588,'15868','Lixos','CV0240',240,5,1200).
ponto(-9.14402783597271, 38.7068925345681,589,'15868','Lixos','CV0140',140,5,700).
ponto(-9.14402783597271, 38.7068925345681,590,'15868','Papel e Cartao','CV0140',140,2,280).
ponto(-9.14447892499263, 38.7070868003123,591,'15869','Lixos','CV0140',140,4,560).
ponto(-9.14447892499263, 38.7070868003123,592,'15869','Lixos','CV0120',120,2,240).
ponto(-9.14447892499263, 38.7070868003123,593,'15869','Lixos','CV0140',140,4,560).
ponto(-9.14447892499263, 38.7070868003123,594,'15869','Lixos','CV0120',120,2,240).
ponto(-9.14441649477531, 38.7067360350326,602,'15871','Lixos','CV0240',240,1,240).
ponto(-9.14441649477531, 38.7067360350326,603,'15871','Lixos','CV0240',240,1,240).
ponto(-9.14416983805955, 38.7071795562796,611,'15873','Lixos','CV0140',140,2,280).
ponto(-9.14416983805955, 38.7071795562796,612,'15873','Lixos','CV0240',240,1,240).
ponto(-9.14416983805955, 38.7071795562796,613,'15873','Lixos','CV0120',120,1,120).
ponto(-9.14416983805955, 38.7071795562796,614,'15873','Lixos','CV0140',140,2,280).
ponto(-9.14416983805955, 38.7071795562796,615,'15873','Lixos','CV0240',240,1,240).
ponto(-9.14416983805955, 38.7071795562796,616,'15873','Lixos','CV0120',120,1,120).
ponto(-9.14401335962188, 38.7074961856438,617,'15875','Lixos','CV0090',90,1,90).
ponto(-9.14401335962188, 38.7074961856438,618,'15875','Lixos','CV0240',240,1,240).
ponto(-9.14401335962188, 38.7074961856438,619,'15875','Lixos','CV0140',140,1,140).
ponto(-9.14401335962188, 38.7074961856438,620,'15875','Lixos','CV0090',90,1,90).
ponto(-9.14401335962188, 38.7074961856438,621,'15875','Lixos','CV0240',240,1,240).
ponto(-9.14401335962188, 38.7074961856438,622,'15875','Lixos','CV0140',140,1,140).
ponto(-9.14404732851606, 38.7058294413797,623,'15876','Lixos','CV0240',240,7,1680).
ponto(-9.14324233291858, 38.7058093867279,624,'15877','Lixos','CV0240',240,1,240).
ponto(-9.14281417070135, 38.7064256244632,625,'15878','Lixos','CV0240',240,1,240).
ponto(-9.14281417070135, 38.7064256244632,626,'15878','Lixos','CV0140',140,1,140).
ponto(-9.14281417070135, 38.7064256244632,627,'15878','Lixos','CV0240',240,1,240).
ponto(-9.14281417070135, 38.7064256244632,628,'15878','Lixos','CV0140',140,1,140).
ponto(-9.14222984894838, 38.7065657946941,629,'15879','Lixos','CV0140',140,2,280).
ponto(-9.14222984894838, 38.7065657946941,630,'15879','Lixos','CV0140',140,2,280).
ponto(-9.14222984894838, 38.7065657946941,631,'15879','Papel e Cartao','CV0140',140,1,140).
ponto(-9.14622006213417, 38.7082697430488,632,'15880','Lixos','CV0090',90,1,90).
ponto(-9.14622006213417, 38.7082697430488,633,'15880','Lixos','CV0090',90,1,90).
ponto(-9.14956443154647, 38.7090063348014,634,'15881','Lixos','CV0240',240,1,240).
ponto(-9.15059949287472, 38.709033347926,635,'15882','Lixos','CV0140',140,1,140).
ponto(-9.15059949287472, 38.709033347926,636,'15882','Lixos','CV0090',90,1,90).
ponto(-9.1446205470921, 38.7073467997814,637,'15883','Lixos','CV0090',90,1,90).
ponto(-9.1446205470921, 38.7073467997814,638,'15883','Lixos','CV0090',90,1,90).
ponto(-9.14476400355141, 38.7069221884751,639,'15884','Lixos','CV0240',240,1,240).
ponto(-9.14476400355141, 38.7069221884751,640,'15884','Lixos','CV0240',240,1,240).
ponto(-9.14476400355141, 38.7069221884751,641,'15884','Papel e Cartao','CA1100',1100,1,1100).
ponto(-9.14508822170713, 38.7079012322757,642,'15885','Lixos','CV0240',240,1,240).
ponto(-9.14508822170713, 38.7079012322757,643,'15885','Lixos','CV0240',240,1,240).
```

```
ponto(-9.14488886847759, 38.7076237175904,644,'15886','Lixos','CV0140',140,4,560).
ponto(-9.14488886847759, 38.7076237175904,645,'15886','Lixos','CV0240',240,1,240).
ponto(-9.14488886847759, 38.7076237175904,646,'15886','Papel e Cartao','CV0140',140,1,140).
ponto(-9.14447518350945, 38.7076363096132,647,'15887','Lixos','CV0240',240,1,240).
ponto(-9.14447518350945, 38.7076363096132,648,'15887','Lixos','CV0240',240,1,240).
ponto(-9.14329884772974, 38.7065565483955,649,'15888','Lixos','CV0240',240,2,480).
ponto(-9.14329884772974, 38.7065565483955,650,'15888','Papel e Cartao','CV0240',240,1,240).
ponto(-9.14340471238935, 38.7067267810448,651,'15889','Lixos','CV0140',140,2,280).
ponto(-9.14340471238935, 38.7067267810448,652,'15889','Lixos','CV0090',90,1,90).
ponto(-9.14340471238935, 38.7067267810448,653,'15889','Lixos','CV0120',120,1,120).
ponto(-9.14340471238935, 38.7067267810448,654,'15889','Lixos','CV0140',140,2,280).
ponto(-9.14340471238935, 38.7067267810448,655,'15889','Lixos','CV0090',90,1,90).
ponto(-9.14340471238935, 38.7067267810448,656,'15889','Lixos','CV0120',120,1,120).
ponto(-9.14340471238935, 38.7067267810448,657,'15889','Papel e Cartao','CV0140',140,2,280).
ponto(-9.14361174262131, 38.7067339965055,658,'15890','Lixos','CV0090',90,1,90).
ponto(-9.14361174262131, 38.7067339965055,659,'15890','Lixos','CV0140',140,4,560).
ponto(-9.14361174262131, 38.7067339965055,660,'15890','Lixos','CV0090',90,1,90).
ponto(-9.14361174262131, 38.7067339965055,661,'15890','Lixos','CV0140',140,4,560).
ponto(-9.1437590142256, 38.7065795884311,662,'15891','Lixos','CV0140',140,2,280).
ponto(-9.1437590142256, 38.7065795884311,663,'15891','Lixos','CV0140',140,2,280).
ponto(-9.1437590142256, 38.7065795884311,664,'15891','Papel e Cartao','CV0240',240,1,240).
ponto(-9.14200502758224, 38.7069280516782,333,'15892','Lixos','CV0140',140,1,140).
ponto(-9.14201010036865, 38.7072883208907,334,'15893','Lixos','CV0090',90,1,90).
ponto(-9.14201010036865, 38.7072883208907,335,'15893','Lixos','CV0140',140,4,560).
ponto(-9.14201010036865, 38.7072883208907,336,'15893','Lixos','CV0240',240,2,480).
ponto(-9.14201010036865, 38.7072883208907,337,'15893','Lixos','CV0090',90,1,90).
ponto(-9.14201010036865, 38.7072883208907,338,'15893','Lixos','CV0140',140,4,560).
ponto(-9.14201010036865, 38.7072883208907,339,'15893','Lixos','CV0240',240,2,480).
ponto(-9.1420813502454, 38.7074498453842,342,'15894','Lixos','CV0240',240,1,240).
ponto(-9.1420813502454, 38.7074498453842,343,'15894','Lixos','CV0140',140,2,280).
ponto(-9.1420813502454, 38.7074498453842,344,'15894','Lixos','CV0240',240,1,240).
ponto(-9.1420813502454, 38.7074498453842,340,'15894','Lixos','CV0140',140,2,280).
ponto(-9.1420813502454, 38.7074498453842,341,'15894','Papel e Cartao','CV0240',240,1,240).
ponto(-9.14986368552042, 38.7090307521821,345,'15896','Lixos','CV0140',140,1,140).
ponto(-9.14534815013291, 38.7075837039225,346,'15897','Lixos','CV0140',140,2,280).
ponto(-9.14534815013291, 38.7075837039225,347,'15897','Lixos','CV0240',240,1,240).
ponto(-9.14534815013291, 38.7075837039225,348,'15897','Lixos','CV0140',140,2,280).
ponto(-9.14534815013291, 38.7075837039225,349,'15897','Lixos','CV0240',240,1,240).
ponto(-9.14534815013291, 38.7075837039225,350,'15897','Papel e Cartao','CV0140',140,1,140).
ponto(-9.14437680001953, 38.7063670578938,351,'15898','Lixos','CV0090',90,1,90).
ponto(-9.14437680001953, 38.7063670578938,352,'15898','Lixos','CV0240',240,1,240).
ponto(-9.14437680001953, 38.7063670578938,353,'15898','Lixos','CV0140',140,1,140).
ponto(-9.14473414894124, 38.7064360246404,354,'15899','Lixos','CV0140',140,1,140).
ponto(-9.14751690960422, 38.7146089969724,43,'19365','Papel e Cartao','CV0240',240,2,480).
ponto(-9.14751690960422, 38.7146089969724,42,'19365','Vidro','CV0240',240,1,240).
ponto(-9.14751690960422, 38.7146089969724,41,'19365','Embalagens','CV0240',240,2,480).
ponto(-9.14751690960422, 38.7146089969724,40,'19365','Lixos','CV0240',240,4,960).
ponto(-9.14751690960422, 38.7146089969724,39,'19365','Lixos','CV0240',240,4,960).
ponto(-9.15186340541146, 38.7154539083141,133,'19216','Papel e Cartao','CV0240',240,1,240).
ponto(-9.15186340541146, 38.7154539083141,134,'19216','Papel e Cartao','CV0240',240,1,240).
ponto(-9.15186340541146, 38.7154539083141,135,'19216','Embalagens','CV0240',240,1,240).
ponto(-9.15186340541146, 38.7154539083141,136,'19216','Lixos','CV0240',240,1,240).
ponto(-9.15186340541146, 38.7154539083141,137,'19216','Lixos','CV0240',240,1,240).
ponto(-9.15186340541146, 38.7154539083141,138,'19216','Vidro','CV0240',240,1,240).
ponto(-9.15186340541146, 38.7154539083141,139,'19216','Embalagens','CV0240',240,1,240).
ponto(-9.14754956156816, 38.7112307798315,210,'19280','Papel e Cartao','CV0240',240,3,720).
ponto(-9.14754956156816, 38.7112307798315,209,'19280','Lixos','CV0240',240,9,2160).
ponto(-9.14754956156816, 38.7112307798315,208,'19280','Embalagens','CV0240',240,2,480).
ponto(-9.14754956156816, 38.7112307798315,211,'19280','Lixos','CV0240',240,9,2160).
ponto(-9.14754956156816, 38.7112307798315,207,'19280','Vidro','CV0240',240,1,240).
ponto(-9.14422400736949, 38.7069448834987,233,'19301','Vidro','VSP',1500,1,1500).
ponto(-9.14422400736949, 38.7069448834987,234,'19301','Vidro','VSP',1500,1,1500).
ponto(-9.14440731848469, 38.7101590906286,265,'19315','Vidro','CV0140',140,1,140).
ponto(-9.14490054388133, 38.7108934583781,231,'19300','Lixos','CV0240',240,2,480).
ponto(-9.14490054388133, 38.7108934583781,232,'19300','Lixos','CV0240',240,2,480).
```

```
ponto(-9.14388063353866, 38.7094970732753,66,'19407','Papel e Cartao','CV0240',240,1,240).
ponto(-9.14388063353866, 38.7094970732753,67,'19407','Embalagens','CV0240',240,1,240).
ponto(-9.1513885505624, 38.7087382169296,214,'19282','Papel e Cartao','CV0140',140,1,140).
ponto(-9.1513885505624, 38.7087382169296,213,'19282','Embalagens','CV0140',140,1,140).
ponto(-9.1531085611251, 38.713272150325,180,'19257','Vidro','CV0140',140,2,280).
ponto(-9.1531085611251, 38.713272150325,181,'19257','Vidro','CV0140',140,2,280).
ponto(-9.1531085611251, 38.713272150325,182,'19257','Vidro','CV0140',140,2,280).
ponto(-9.15311830964257, 38.7163797615419,225,'19293','Vidro','CV0140',140,1,140).
ponto(-9.14226074740088, 38.7079437265862,245,'19310','Vidro','CV0240',240,1,240).
ponto(-9.14481349084208, 38.7079846835909,267,'19319','Embalagens','CV0140',140,1,140).
ponto(-9.15360843944475, 38.7144658214995,163,'19236','Vidro','CV0140',140,1,140).
ponto(-9.15360843944475, 38.7144658214995,164,'19236','Organicos','CV0140',140,1,140).
ponto(-9.14460348898318, 38.7142829757734,227,'19295','Vidro','CV0140',140,1,140).
ponto(-9.14464751990681, 38.7108866428877,48,'19371','Lixos','CV0240',240,3,720).
ponto(-9.14464751990681, 38.7108866428877,47,'19371','Lixos','CV0240',240,3,720).
ponto(-9.15242517945081, 38.7096839983153,121,'19390','Lixos','CV0240',240,3,720).
ponto(-9.15242517945081, 38.7096839983153,122,'19390','Lixos','CV0240',240,3,720).
ponto(-9.15305561262562, 38.7119754868767,205,'19278','Vidro','CV0240',240,1,240).
ponto(-9.15305561262562, 38.7119754868767,206,'19278','Organicos','CV0240',240,1,240).
ponto(-9.14301166531656, 38.7073877536231,85,'19415','Embalagens','CV0140',140,1,140).
ponto(-9.14385688118425, 38.7094432317241,290,'19337','Vidro','CV0140',140,1,140).
ponto(-9.14385688118425, 38.7094432317241,291,'19337','Papel e Cartao','CV0140',140,1,140).
ponto(-9.14385688118425, 38.7094432317241,292,'19337','Embalagens','CV0140',140,1,140).
ponto(-9.15264387969663, 38.7105108085557,692,'21843','Organicos','CV0090',90,1,90).
ponto(-9.15264387969663, 38.7105108085557,693,'21843','Vidro','CV0090',90,1,90).
ponto(-9.15130039259538, 38.7122430297428,694,'21844','Papel e Cartao','CV0240',240,1,240).
ponto(-9.15130039259538, 38.7122430297428,695,'21844','Papel e Cartao','CV0240',240,1,240).
ponto(-9.15130039259538, 38.7122430297428,696,'21844','Embalagens','CV0240',240,1,240).
ponto(-9.15130039259538, 38.7122430297428,697,'21844','Lixos','CV0240',240,1,240).
ponto(-9.15130039259538, 38.7122430297428,698,'21844','Vidro','CV0240',240,1,240).
ponto(-9.15130039259538, 38.7122430297428,699,'21844','Embalagens','CV0240',240,1,240).
ponto(-9.15130039259538, 38.7122430297428,700,'21844','Lixos','CV0240',240,1,240).
ponto(-9.14752862812709, 38.7056911463271,938,'21957','Embalagens','CV0240',240,2,480).
ponto(-9.14752862812709, 38.7056911463271,939,'21957','Lixos','CV0240',240,1,240).
ponto(-9.14752862812709, 38.7056911463271,940,'21957','Vidro','CV0240',240,1,240).
ponto(-9.1507756102811, 38.7084823352254,944,'21959','Lixos','CV0240',240,1,240).
ponto(-9.1507756102811, 38.7084823352254,945,'21959','Embalagens','CV0240',240,2,480).
ponto(-9.1507756102811, 38.7084823352254,946,'21959','Vidro','CV0140',140,4,560).
ponto(-9.1507756102811, 38.7084823352254,947,'21959','Organicos','CV0240',240,3,720).
ponto(-9.14605416845402, 38.707919877341,843,'21911','Papel e Cartao','ECIS3000',3000,2,6000).
ponto(-9.14605416845402, 38.707919877341,844,'21911','Embalagens','ECIS3000',3000,2,6000).
ponto(-9.14605416845402, 38.707919877341,845,'21911','Vidro','ECIS3000',3000,1,3000).
ponto(-9.14605416845402, 38.707919877341,846,'21911','Lixos','ECIS3000',3000,2,6000).
ponto(-9.14906797670637, 38.7088575233976,921,'21952','Papel e Cartao','CV0090',90,1,90).
ponto(-9.14906797670637, 38.7088575233976,922,'21952','Organicos','CV0090',90,1,90).
ponto(-9.14906797670637, 38.7088575233976,923,'21952','Vidro','CV0140',140,1,140).
ponto(-9.14906797670637, 38.7088575233976,924,'21952','Embalagens','CV0090',90,1,90).
ponto(-9.15158073972755, 38.7077006401431,965,'21969','Papel e Cartao','CV0240',240,1,240).
ponto(-9.15158073972755, 38.7077006401431,966,'21969','Embalagens','CV0090',90,1,90).
ponto(-9.14444160448415, 38.7093300738379,715,'21854','Papel e Cartao','CV0140',140,1,140).
ponto(-9.14444160448415, 38.7093300738379,716,'21854','Vidro','CV0240',240,1,240).
ponto(-9.14444160448415, 38.7093300738379,717,'21854','Embalagens','CV0140',140,1,140).
ponto(-9.14408342911238, 38.7116481877359,897,'21938','Organicos','CV0240',240,1,240).
ponto(-9.14329248945415, 38.7077366293235,951,'21961','Embalagens','CV0090',90,1,90).
ponto(-9.14731529890835, 38.7117372535528,736,'21866','Papel e Cartao','CV0090',90,1,90).
ponto(-9.14731529890835, 38.7117372535528,737,'21866','Organicos','CV0090',90,1,90).
ponto(-9.14731529890835, 38.7117372535528,738,'21866','Vidro','CV0090',90,1,90).
ponto(-9.14731529890835, 38.7117372535528,739,'21866','Embalagens','CV0090',90,1,90).
ponto(-9.14731529890835, 38.7117372535528,740,'21866','Lixos','CV0090',90,1,90).
ponto(-9.14731529890835, 38.7117372535528,741,'21866','Lixos','CV0090',90,1,90).
ponto(-9.14642470554241, 38.7105469477308,959,'21966','Lixos','CV0140',140,3,420).
ponto(-9.14642470554241, 38.7105469477308,960,'21966','Lixos','CV0140',140,3,420).
ponto(-9.14642470554241, 38.7105469477308,961,'21966','Organicos','CV0140',140,1,140).
ponto(-9.14642470554241, 38.7105469477308,962,'21966','Vidro','CV0140',140,1,140).
ponto(-9.14497783133511, 38.7106675928332,963,'21967','Organicos','CV0140',140,1,140).
```

```
ponto(-9.14323374592279, 38.7076470601131,906,'21944','Vidro','CV0240',240,1,240).
ponto(-9.14323374592279, 38.7076470601131,907,'21944','Organicos','CV0240',240,1,240).
ponto(-9.14841674214284, 38.7075210227377,880,'21932','Embalagens','CV0240',240,1,240).
ponto(-9.14841674214284, 38.7075210227377,881,'21932','Papel e Cartao','CV0240',240,1,240).
ponto(-9.14620232759384, 38.7078285138109,789,'21889','Vidro','CV0090',90,1,90).
ponto(-9.14620232759384, 38.7078285138109,790,'21889','Vidro','CV0090',90,1,90).
ponto(-9.14259782763484, 38.7090217515467,869,'21925','Papel e Cartao','CV0090',90,1,90).
ponto(-9.14259782763484, 38.7090217515467,870,'21925','Lixos','CV0140',140,1,140).
ponto(-9.14259782763484, 38.7090217515467,871,'21925','Lixos','CV0140',140,1,140).
ponto(-9.14584574313843, 38.7102547130905,874,'21927','Lixos','CV0240',240,5,1200).
ponto(-9.14584574313843, 38.7102547130905,875,'21927','Lixos','CV0240',240,5,1200).
ponto(-9.14408457176942, 38.7117292486178,898,'21939','Organicos','CV0240',240,1,240).
ponto(-9.15225663484226, 38.7140002169301,912,'21949','Organicos','CV0140',140,1,140).
ponto(-9.15225663484226, 38.7140002169301,913,'21949','Vidro','CV0140',140,1,140).
ponto(-9.15225663484227, 38.7140002169302,950,'21950','Deposicao','-',0,0,0).
```

A.2 grafo.pl

(O ficheiro *grafo.pl* completo, com o grafo criado, não deixava o documento compilar corretamente, sendo porém apresentado parcialmente em 2.1.3 e no código anexado no envio.)

A.3 indiv.pl

Este ficheiro contém o código base do programa.

```
%-----------------------------------------------
% SRCR INDIVIDUAL | Joana Afonso Gomes | a84912
%-----------------------------------------------
:- set_prolog_flag( discontiguous_warnings,off ).
:- set_prolog_flag( single_var_warnings,off ).
:- set_prolog_flag( unknown,fail ).
:- set_prolog_flag(toplevel_print_options,[quoted(true), portrayed(true),
 ↪  max_depth(0)]).
:- set_prolog_flag(stack_limit, 4_294_967_296).
%-----------------------------------------------
:- op( 900,xfy,'::' ).
:- dynamic '-'/1.
:- dynamic(ponto/9).
:- op(  500,  fx, [ +, - ]).
```

```
:- op(  300, xfx, [ mod ]).
:- op(  200, xfy, [ ^ ]).
%--------------------------------------------
:- include('baseConhecimento.pl').
:- include('grafo.pl').
:- include('auxs.pl').
:- use_module(library(lists)).
%--------------------------------------------

% Gerar os circuitos de recolha

% -- Caminho entre nodos --
caminho(G,X,Y,C):-
    caminhoAux(G,X,[Y],C).

caminhoAux(_,X,[X|T],[X|T]).
caminhoAux(G,X,[Y|T],P) :-
    adjacente(Prox_nodo,Y,G), nao(membro(Prox_nodo,[Y|T])),
     ↪  caminhoAux(G,X,[Prox_nodo,Y|T],P).

% -- Caminho com a distancia percorrida --
caminhoK(G,A,B,P,D) :-
  caminhoAuxK(G,A,[B],P,D).

caminhoAuxK(G,A,[A|P1],[A|P],0,[]).

caminhoAuxK(G,A,[Y|P1],P,D1) :-
  adjacente(X,Y,Di,G),
  nao(membro(X,[Y|P1])),
  distanciaEntreNodos(X,Y,Di),
  caminhoAuxK(G,A,[X,Y|P1],P,K),
  D1 is D + Di.

% Todos os circuitos entre dois nodos

allCircuitos(G,X,Y,R):- solucoes(C,dFirst(G,X,Y,C),R).

% -- Gerar circuitos com Depth First --
```

```
dFirst(G,X,Y,R) :- dFirst(G,X,Y,[X],R).

dFirst(G,X,Y,V,[aresta(X,Y)]) :- adjacente(X,Y,G).

dFirst(G,X,Y,V,[aresta(X,Z)|R]) :-
 adjacente(X,Z,G),
 \+ membrochk(aresta(X,Z),V),
 \+ membro(Z,V),
 dFirst(G,Z,Y,[Z|V],R),
 Z \= Y.

% T
test_dFirst(R) :- g(G), getIdGaragem(Gar), getIdDeposicao(D),
                                  dfFirst(G,Gar,D,R).

% -- Gerar circuitos com Breadth First --

bFirst(G,X,Y,Visited) :- bFirstAux(G,Y,Successors,
                                        [],
                                        RevVisited),
                                        sucessor(G,X,Successors),
                                        inverso(RevVisited, Visited).

bFirstAux(G,Y,[], History, []).

bFirstAux(G,Y,[aresta(P,Y)|_], History, [aresta(P,Y)|History]).

bFirstAux(G,Y,[aresta(X,Z)|RestQ], History, RevVisited) :-
 ↪  sucessor(G,Z,Cats),
```

```
% Seletiva

% Através da depthFirst para pontos que tenham um tipo especifico de residuos
 ↪  (circuitos de recolha seletiva)

pontoTipo(aresta(X,Y),T) :- ponto(_,_,_,X,Tp,_,_,_,Ctotal),
 ↪  ponto(_,_,_,Y,Tp,_,_,_,Ctotal).

dFTipo(G,T,X,Y,P) :- dFTipo(G,T,X,Y,[X],P).

dFTipo(G,T,X,Y,V,[aresta(X,Y)]) :- adjacente(X,Y,G),
                        pontoTipo(aresta(X,Y),T).

dFTipo(G,T,X,Y,V,[aresta(X,Z)|P]) :-
 adjacente(X,Z,G),
 pontoTipo(aresta(X,Z),T),
 \+ membrochk(aresta(X,Z),V),
 \+ membro(Z,V),
 dFTipo(G,T,Z,Y,[Z|V],P).


%-----------------------------------------------

% Identificar circuitos com mais pontos de recolha (por tipo de residuo)

% Devolve os pontos de uma aresta se tiverem aquele tipo de residuo
edgePointsTipo(aresta(X,_),Tp,R):-
 ↪  solucoes(ponto(Lat,Lon,O,X,Tp,Ct,Ccap,Cqt,Ctotal),
    ponto(Lat,Lon,O,X,Tp,Ct,Ccap,Cqt,Ctotal),R).

nrPontosRecolhaT([],T,Count,R).
nrPontosRecolhaT([A],T,Count,R) :- edgePointsTipo(A,Tp,S),
                                   length(S,L),
                                   Count2 is Count + L,
                                   nrPontosRecolhaT([],Count2,R).
nrPontosRecolhaT([H|T],Count,T,R) :- edgePointsTipo(H,Tp,S),
                                   length(S,L),
                                   Count2 is Count + L,
```

```
                                                nrPontosRecolhaT(T,Count2,R).

% T
teste_nrPontosRecolha(R) :- edgePointsTipo(aresta('15875','21844'),'Lixos',0,S).

%-----------------------------------------------

% Escolher o circuito mais rápido (criterio distancia)

% DepthFirst
dFDistancia(G,X,Y,R) :- dFDistancia(G,X,Y,[X],R,Km).

dFDistancia(G,X,Y,V,[aresta(X,Y)],0) :- adjacente(X,Y,G).

dFDistancia(G,X,Y,V,[aresta(X,Z)|R],D) :-
 adjacente(X,Z,G),
 Di = distanciaEntreNodos(X,Z),
 \+ membrochk(aresta(X,Z),V),
 \+ membro(Z,V),
 D2 = distanciaEntreNodos(Z,Y),
 dFDistancia(G,Z,Y,[Z|V],R,D2),
 D is D2 + Di.

% Percurso mais rapido (menor distancia)
maisRapido(G,X,Y,P,D):-solucoes((P,C),(dFDistancia(G,X,Y,P,D),!,C=D),L),
                                          minimo(L,(P,C)), D=C.

test_maisRapido(Y,R) :- g(G), getIdGaragem(Gar), getIdDeposicao(Dep),
 ↪  maisRapido(G,Gar,Dep,P,D),
                                addEllement(aresta(Dep,Gar)).

%-----------------------------------------------

% Comparar circuitos de recolha tendo em conta a distância média percorrida entre
 ↪  pontos de recolha

% Calcula a distância total de um path
calcDist([],Di,Di).
```

```
calcDist([A],Di,Di) :- edgePoints(A,A1), [A2|T2]=A1, edgePoints2(A,B), [B1|B2]=B,
                                        distanciaEntrePontos(A2,B1,Dist2),
                                        ZZ is Dist2 + Di,
                                        calcDist([],ZZ,Dist).
calcDist([A|X],Di,Dist):- edgePoints(A,A1), [A2|T2]=A1, edgePoints2(A,B),
↪  [B1|B2]=B,
                    distanciaEntrePontos(A2,B1,Dist2),
                    ZZ is Dist2 + Di,
                    calcDist(X,ZZ,Dist).


compararDistanciaMedia(L,R) :- compararDistanciaMediaAux(L,'0',0).

compararDistanciaMediaAux([],Ci,Dm).
compararDistanciaMediaAux([H],Ci,Dm) :- calcDist(H,0,D),
                        (D > Dm -> Dm is D, Ci is H),
                        compararDistanciaMediaAux([],Ci,Dm).
compararDistanciaMediaAux([H|T],Ci,Dm) :- calcDist(H,0,D),
                        (D > Dm -> Dm is D, Ci is H),
                        compararDistanciaMediaAux(T,Ci,Dm).

test_compararDistanciaMedia(G,R) :- getIdGaragem(Gar), getIdDeposicao(D), g(G),
                        allCircuitos(G,Gar,D,L),
                        compararDistanciaMedia(L,R).

%-----------------------------------------------

% Comparar circuitos de recolha tendo em conta a quantidade recolhida

compararQtRecolhida(L,R) :- compararQtRecolhidaAux(L,'0',0).

compararQtRecolhidaAux([],Sum,R).
compararQtRecolhidaAux([H],Sum,R) :- edgePoints(H,E),
                        somarQuantidadesRecolhidas(H,0,S),
                        Sum2 = Sum + S,
                        compararQtRecolhidaAux([],Sum2,R).
compararQtRecolhidaAux([H|T],Sum,R) :- edgePoints(H,E),
                        somarQuantidadesRecolhidas(H,0,S),
```

```
                        Sum2 = Sum + S,
                        compararQtRecolhidaAux(T,Sum2,R).

somarQuantidadesRecolhidas([],Sum,R).
somarQuantidadesRecolhidas([A],Sum,R) :- ponto(Lat,Lon,O,X,T,Ct,Ccap,Cqt,Ctotal) =
 ↪  A,
                        Sum2 = Sum + Ctotal,
                        somarQuantidadesRecolhidas([],Sum2,R).
somarQuantidadesRecolhidas([H|T],Sum,R) :- ponto(Lat,Lon,O,X,T,Ct,Ccap,Cqt,Ctotal)
 ↪  = H,
                        Sum2 = Sum + Ctotal,
                        somarQuantidadesRecolhidas(T,Sum2,R).
```

A.4 auxs.pl

Contém predicados auxiliares, tanto para tratamento de informação no geral (como processamento de listas) como auxiliares no contexto do projeto, que são usados no código base.

```
% -- Auxiliares Gerais --

nao( Questao ) :-
    Questao, !, fail.
nao( Questao ).

membro(X, [X|_]).
membro(X, [_|Xs]):-
        membro(X, Xs).

membrochk(X,[X|_]) :- !.
membrochk(X,[_|T]):- membrochk(X,T).

% Concatena listas
concat(List1, List2, Result):-
   append(List1, List2, Result).

% Comprimento de uma lista
compLista([H],R) :- R=1.
compLista([H|T],R) :- compLista(T,O), R is O + 1.
```

```
solucoes( X,Y,Z ) :- findall( X,Y,Z ).

repetidos([],[]).
repetidos([X|L],[X|NL]) :- removerElem(L,X,TL), repetidos(TL,NL).

removerElem([],_,[]).
removerElem([X|L],X,NL) :- removerElem(L,X,NL).
removerElem([X|L],Y,[X|NL]) :- X \== Y, removerElem(L,Y,NL).

list_min([L|Ls], Min) :-
    list_min(Ls, L, Min).

list_min([], Min, Min).
list_min([L|Ls], Min0, Min) :-
    Min1 is min(L, Min0),
    list_min(Ls, Min1, Min).

addElement(X, [], [X]).
addElement(X, [Y | Rest], [X,Y | Rest]) :- X @< Y, !.
addElement(X, [Y | Rest1], [Y | Rest2]) :- addElement(X, Rest1, Rest2).

getHead([], [], []).
getHead([X|_], [], [X]).
getHead([], [X|_], [X]).

inverso(Xs, Ys):-
        inverso(Xs, [], Ys).

inverso([], Xs, Xs).
inverso([X|Xs],Ys, Zs):-
        inverso(Xs, [X|Ys], Zs).

seleciona(E, [E|Xs], Xs).
seleciona(E, [X|Xs], [X|Ys]) :- seleciona(E, Xs, Ys).

minimo([(P,X)],(P,X)).
minimo([(Px,X)|L],(Py,Y)):- minimo(L,(Py,Y)), X>Y.
minimo([(Px,X)|L],(Px,X)):- minimo(L,(Py,Y)), X=<Y.

average( List, Average ):-
    sum( List, Sum ),
    length( List, Length ),
    Length > 0,
    Average is Sum / Length.

count([],0).
count([H|Tail], N) :-
    count(Tail, N1),
```

```
    (  number(H)
    -> N is N1 + 1
    ;  N = N1
    ).

%-----------------------------------------------

% -- Auxiliares do Programa --

% Obter todas os nodos do grafo
todosNodos(R) :- g(G),
                                G = grafo(N,_),
                                R = N.

% Obter todas as arestas do grafo com nodos de um certo tipo de residuo
todasArestasTipo(T,R) :- g(G),
                G = grafo(_,L_arestas),
                A = L_arestas,
                solucoes(aresta(X,Y),(member((aresta(X,Y)),A),
                    ponto(_,_,_,X,T,_,_,_,_),ponto(_,_,_,Y,T,_,_,_,_),X\=Y),S),
                repetidos(S,R).

% Obter todas as arestas do grafo
todasArestas(R) :- g(G),
        G = grafo(_,L_arestas),
        R = L_arestas.

% Obter sucessores de um nodo
sucessor(grafo(_,Es),X,R):- solucoes(aresta(X,Y),membro(aresta(X,Y),Es),R).

% Obter o ID da garagem
getIdGaragem(R) :- solucoes(Id,ponto(Lat,Lon,0,Id,'Garagem',Ct,Ccap,Cqt,Ctotal), S), S = [H|T], R = H.

% Obter o ponto da garagem
getPontoGaragem(R) :-
↪  solucoes(ponto(Lat,Lon,0,Id,'Garagem',Ct,Ccap,Cqt,Ctotal),ponto(Lat,Lon,0,Id,'Garagem',Ct,Ccap,Cqt,Ctotal),
↪  S), S = [H|T], R = H.

% Obter o ID do Local de Deposicao
getIdDeposicao(R) :- solucoes(Id,ponto(Lat,Lon,0,Id,'Deposicao',Ct,Ccap,Cqt,Ctotal), S), S = [H|T], R
↪  = H.

% Obter o ponto do Local de Deposicao
getPontoDeposicao(R) :-
↪  solucoes(ponto(Lat,Lon,0,Id,'Deposicao',Ct,Ccap,Cqt,Ctotal),ponto(Lat,Lon,0,Id,'Deposicao',Ct,Ccap,Cqt,Ctotal),
↪  S), S = [H|T], R = H.

latitudeGaragem(R) :- solucoes(Lat,ponto(Lat,_,_,'15804',_,_,_,_,_),S),
```

```
                                        S = [H|T],
                                        R = H.
longitudeGaragem(R) :- solucoes(Lon,ponto(_,Lon,_,'15804',_,_,_,_,_),S),
                                        S = [H|T],
                                        R = H.

% Verifica se dois nodos sao adjacente
adjacente(X,Y,grafo(_,L_arestas)) :- member(aresta(X,Y),L_arestas).

% Calcula a distancia entre dois nodos
distancia(Lat1,Lon1,Lat2,Lon2,D):- N is sqrt((Lat2-Lat1)^2+(Lon2-Lon1)^2),N=D.

distanciaEntrePontos(ponto(Lat1,Lon1,_,X,_,_,_,_,_),ponto(Lat2,Lon2,_,Y,_,_,_,_,_),R) :-
↪  distancia(Lat1,Lon1,Lat2,Lon2,R).

distanciaEntreNodos(X,Y,D) :- pontosID(X,Ps), Ps = [P|T], P = ponto(Lat1,Lon1,_,X,_,_,_,_,_),
                              pontosID(Y,Ps2), Ps2 = [P2|T2], P2 = ponto(Lat2,Lon2,_,Y,_,_,_,_,_),
                              distancia(Lat1,Lon1,Lat2,Lon2,D).

% Calcula a distancia a garagem do ponto
distanciaGaragemPonto(Pt,R) :- getPontoGaragem(P),
                          distanciaEntrePontos(P,Pt,R).

% Diz se aquele ponto tem o tipo de residuo dado como argumento
tipoResiduo(Tp,ponto(_,_,_,_,Tp,_,_,_,_)).

% Todos os pontos com aquele ID
pontosID(Id,R) :-
↪  solucoes(ponto(Lat,Lon,O,Id,T,Ct,Ccap,Cqt,Ctotal),ponto(Lat,Lon,O,Id,T,Ct,Ccap,Cqt,Ctotal),R).

% Todos os pontos com aquele ID e um determinado tipo
pontosIDTipo(Id,Tp,R) :-
↪  solucoes(ponto(Lat,Lon,O,Id,Tp,Ct,Ccap,Cqt,Ctotal),ponto(Lat,Lon,O,Id,Tp,Ct,Ccap,Cqt,Ctotal),R).

% Encontrar todas as arestas de nodos ligados à garagem
ligadosGaragem(G,R) :- solucoes(aresta('15804',Y),adjacente('15804',Y,G),R).

% Encontrar todas as arestas ligadas à garagem com um determinado tipo
ligadosGaragemTipo(G,T,R) :- solucoes(aresta('15804',Y),(adjacente('15804',Y,G)),R).

% Obter coordenadas de um nodo
latitudeNodo(Id,R) :-
↪  solucoes(ponto(Lat,Lon,O,Id,T,Ct,Ccap,Cqt,Ctotal),ponto(Lat,Lon,O,Id,T,Ct,Ccap,Cqt,Ctotal),S),
                          S = [H|T],
                          H = ponto(Lat1,Lon1,O1,Id,T1,Ct1,Ccap1,Cqt1,Ctotal1),
                          R = Lat1.
```

```
longitudeNodo(Id,R) :-
↪  solucoes(ponto(Lat,Lon,O,Id,T,Ct,Ccap,Cqt,Ctotal),ponto(Lat,Lon,O,Id,T,Ct,Ccap,Cqt,Ctotal),S),
                        S = [H|T],
                        H = ponto(Lat1,Lon1,O1,Id,T1,Ct1,Ccap1,Cqt1,Ctotal1),
                        R = Lon1.

% Encontrar o nodo mais perto da garagem
closerToGaragem(G,R) :- ligadosGaragem(G,L),
                        closerToGaragemAux(L,[]).

closerToGaragemAux([],Lista).
closerToGaragemAux([H|T],Lista) :- closerToGaragemAuxAux(H,S),
                                   A = Lista,
                                   addElement(S,A,Var),
                                   write(Var),
                                   closerToGaragemAux(T,Var).

closerToGaragemAuxAux(H,R) :- H = aresta(X,Y),
                              latitudeNodo(Y,Lat), longitudeNodo(Y,Lon),
                              latitudeGaragem(LatG), longitudeGaragem(LonG),
                              distancia(LatG,LonG,Lat,Lon,R).

nodeMaisProxGaragem(R) :- R = '19310'.

nodeMaisProxGaragem_Lixos(R) :- '15844'.
nodeMaisProxGaragem_PapelECartao(R) :- '21844'.
nodeMaisProx_Embalagens(R) :- '21844'.
nodeMaisProx_Vidros(R) :- '19310'.
nodeMaisProx_Organicos(R) :- '15864'.

% Devolve os pontos do primeiro elemento de uma aresta
edgePoints(aresta(X,_),R):-
↪  solucoes(ponto(Lat,Lon,O,X,T,Ct,Ccap,Cqt,Ctotal),ponto(Lat,Lon,O,X,T,Ct,Ccap,Cqt,Ctotal),R).

% Devolve os pontos do segundo elemento de uma aresta
edgePoints2(aresta(_,Y),R):-
↪  solucoes(ponto(Lat,Lon,O,Y,T,Ct,Ccap,Cqt,Ctotal),ponto(Lat,Lon,O,Y,T,Ct,Ccap,Cqt,Ctotal),R).
```