

**Relatório ALGAV**

Miguel Gonçalves 1190903

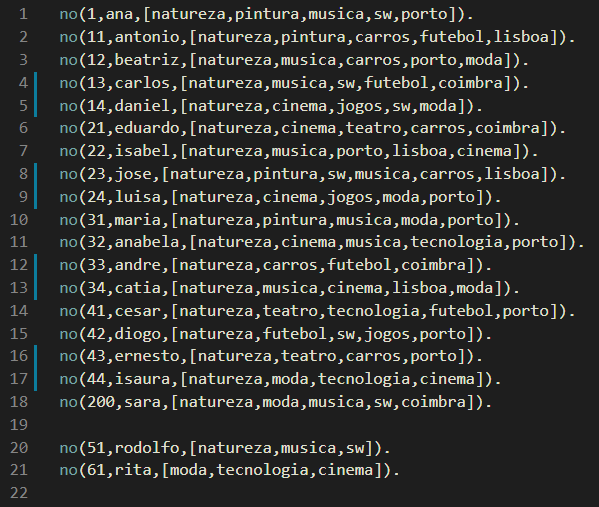
Ruben Rodrigues 1191018

Rui Pinto 1191042

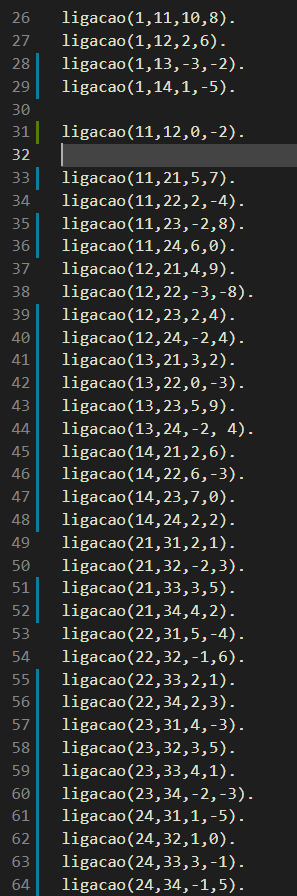
Tomás Limbado 1191106

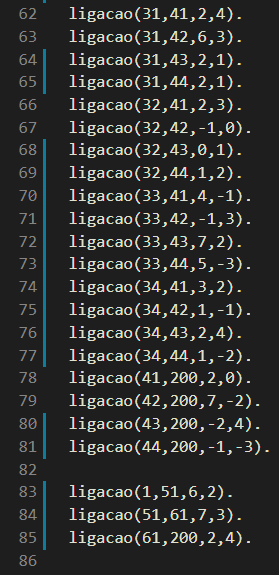
3DJ @ 2021-2022

* **Representação do conhecimento do domínio**

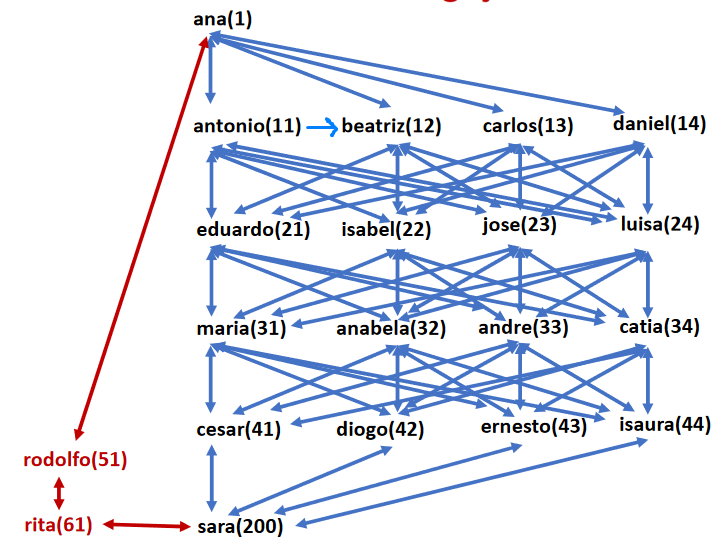


Nodes / Utilizadores (no/3 {id,nome,tags})



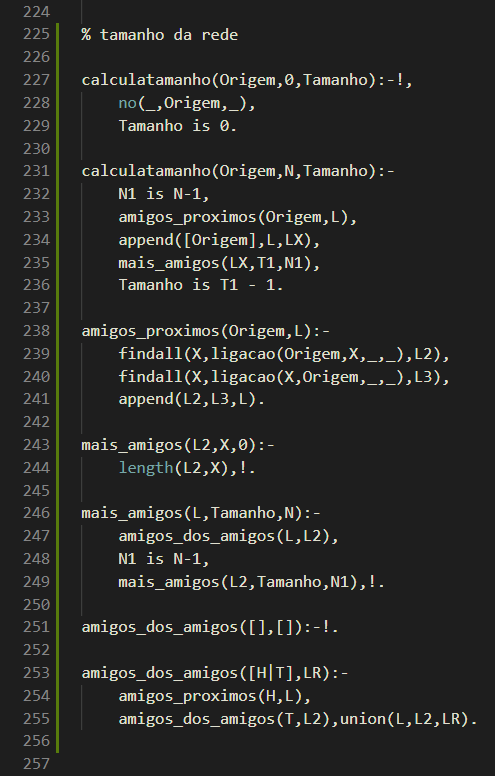


Ligações entre os nodes / utilizadores (ligação/4 {id1,id2,força1,força2})



Representação da rede social

* **Determinação do tamanho da rede de um utilizador até um determinado nível**



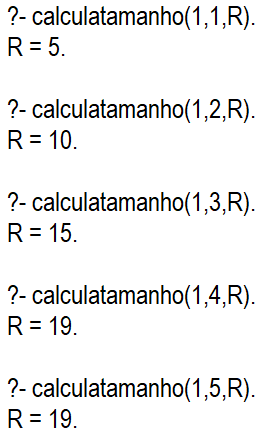
O predicado calculacaminho/3 recebe o id do utilizador e analisar a rede, o N máximo e a variável que vai indicar o tamanho da rede. Invoca o predicado amigos\_proximos/2 de modo a obter os jogadores com ligação direta ao utilizador. De seguida inclui-se o utilizador origem na lista gerada e invoca-se o predicado mais\_amigos/3 para pesquisar pelo resto da rede.

O predicado amigos\_proximos/2, recebendo o id do utilizador a analisar e a lista de amigos próximos a retornar, pesquisa todas ligações que envolvam o utilizador origem e adiciona-os à lista a retornar.

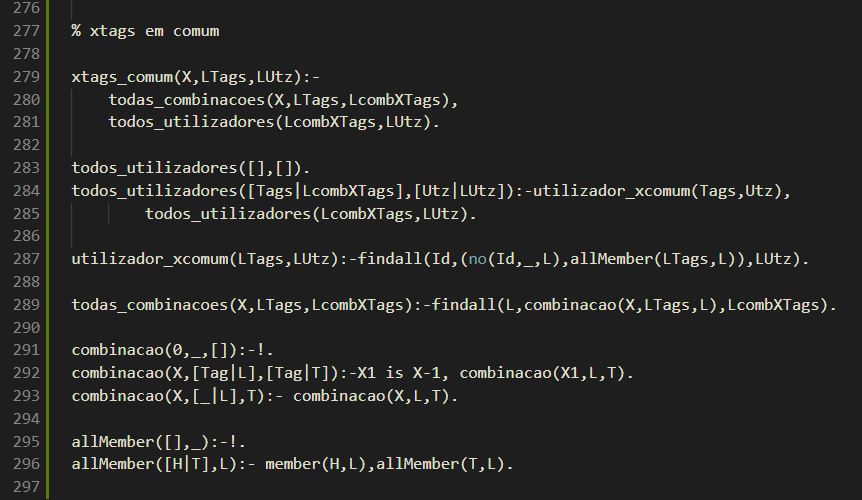
O predicado mais\_amigos/3, recebendo a lista de nível 1, a variável de tamanho final da rede e os níveis a analisar, invoca o predicado amigos\_dos\_amigos/2 para encontrar os amigos próximos dos amigos dos diferentes níveis, invocando-se a si próprio de seguida até os níveis acabarem.

Por fim, o predicado amigos\_dos\_amigos/2, que recebe a lista de amigos já definidos e a lista de retorno, invoca o predicado amigos\_proximos/2 de modo a encontrar os amigos diretos dos utilizadores na lista recebida e une as listas com a lista de retorno.

Exemplo:



* **Obtenção dos utilizadores que tenham em comum X tags**

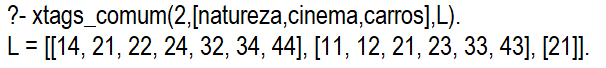


O predicado xtags\_comum/3, que recebe o número de tags (X), a lista de tags a ser analisadas e a lista de retorno de users, dá inicio ao algoritmo. Invoca o predicado todas\_combinacoes/3 de forma a obter todas as combinações de x tags dentro da lista recebida (predicado indicado na documentação das aulas Teórico-Práticas). Após isso é invocado o predicado todos\_utilizadores/2 que retorna todos os jogadores que possuam uma combinação das tags.

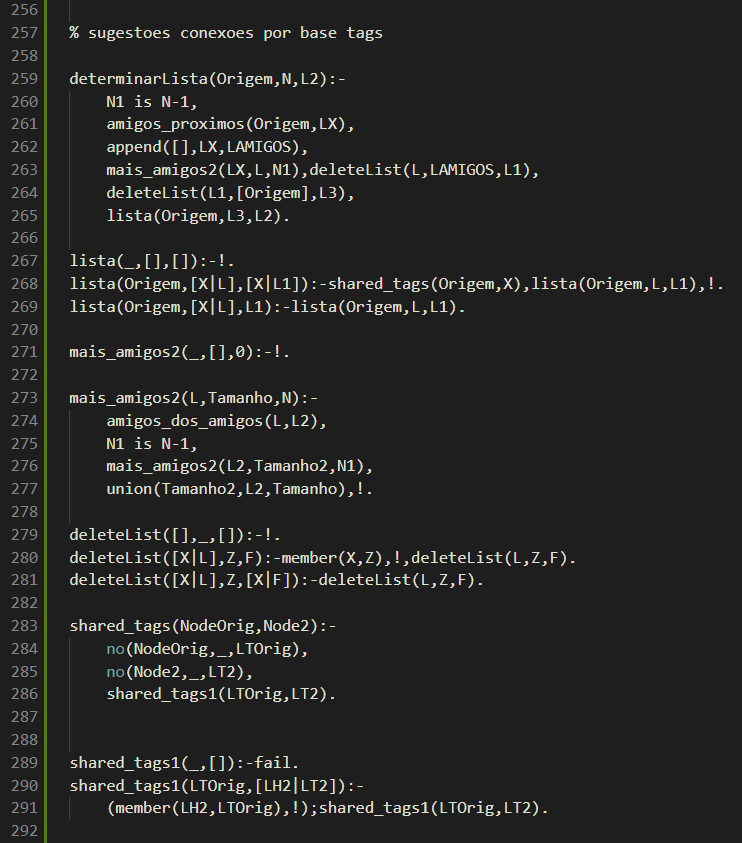
O predicado todos\_utilizadores/2, que recebe a lista das combinações e a lista de retorno de users, o método invoca o predicado utilizador\_xcomum/2 de modo a encontrar os utilizadores que possuam as tags das combinação, e invoca-se a si próprio recursivamente até já ter analisado tudo.

Por fim, o predicado utilizador\_xcomum/2, que recebe a combinação a analisar e a lista de retorno de users, pesquisa todos os utilizadores cujas tags se encontram na combinação e adiciona-os à lista de retorno.

Exemplo:



* **Sugestão das conexões com outros utilizadores tendo por base as tags e conexões partilhadas até determinado nível**



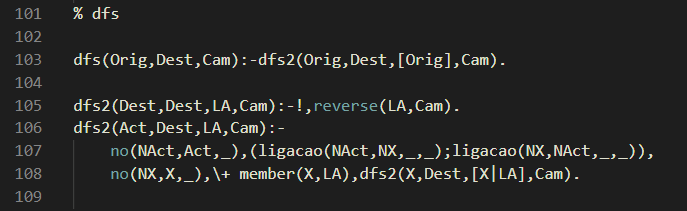
O método determinarLista/3 inicia a sugestão. O algoritmo recebe o id do utilizador origem, o nível até onde se quer efetuar a procura e a lista de retorno. Invoca o predicado amigos\_proximos/2 (explicado anteriormente) e de seguida invoca o mais\_amigos2/3, invocando de seguida o predicado deleteList/3 de modo a apagar os amigos sugeridos repetidos. Por fim, invoca o predicado lista/3 que determina quais desses amigos têm tags em comum com ele.

O predicado mais\_amigos2/3 funciona de forma semelhante ao mais\_amigos/3, que foi explicado anteriormente, porém aqui une as listas geradas.

O predicado deleteList/3 recebe duas listas de ids e uma lista de retorno, sendo que a lista de retorno será preenchida com os membros da primeira lista de ids que não estão contidos na segunda lista de ids.

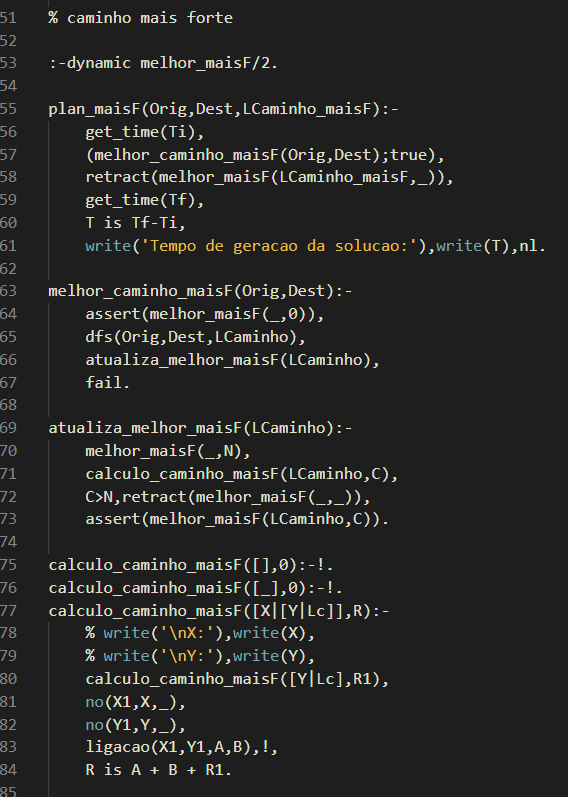
O predicado list/3 filtra a lista de sugestões com base nas tags dos utilizadores origem e sugeridos (através do predicado shared\_tags/2).

* **Determinação de caminhos**
  + **Métodos gerais (DFS)**



Método de busca em profundidade (Depth First Search – DFS) de modo a obter todos os caminhos entre dois nodes de um grafo

* + **Caminho mais forte**



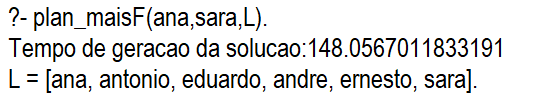
O predicado plan\_maisF/3, que recebe o nome do jogador inicial, o nome do jogador destino e a lista onde vai ser introduzida a solução, é o predicado chamado de modo a iniciar a solução. Este predicado utiliza o predicado melhor\_caminho\_maisF/2 que calcula qual é o caminho mais forte entre os dois jogadores. De seguida seleciona o caminho mais forte e apresenta-o.

O predicado melhor\_caminho\_maisF/2, que recebe o nome do jogador inicial e o nome do jogador destino, inicia a variável dinâmica correspondente ao caminho mais forte e a respetiva força (melhor\_maisF/2) com uma força de 0. De seguida utiliza o dfs/3 para calcular todos os caminhos possíveis entre os dois jogadores introduzidos, enviando então essa lista para o predicado atualiza\_melhor\_maisF/1 de modo a determinar o melhor caminho. O *fail* está posto para obrigar a voltar atrás e gerar um novo caminho.

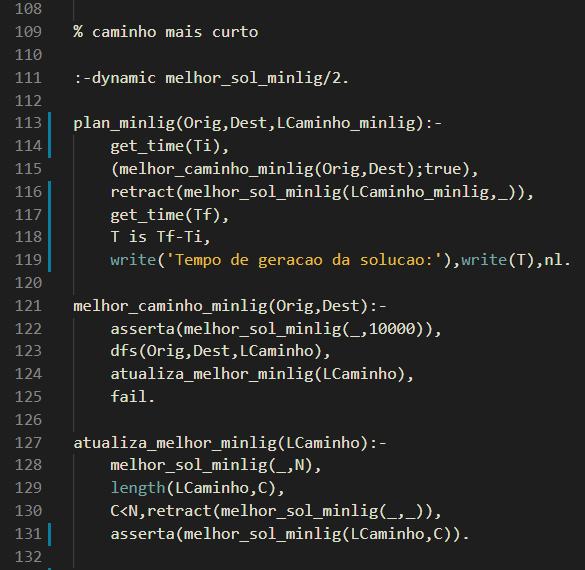
O predicado atualiza\_melhor\_maisF/1, que recebe um caminho, começa por ir ao melhor caminho atual buscar a sua força, de seguida convoca o predicado calculo\_caminho\_maisF/2 que calcula a força do caminho introduzido, de seguida compara a força calculada com o força do melhor caminho atual e se a calculada for maior o melhor caminho atual é substituído pelo novo caminho.

Por fim, o predicado calculo\_caminho\_maisF/2 recebe um caminho e calcula o somatório das forças de ligação no sentido da travessia, através da ligação entre dois utilizadores seguidos da lista.

Exemplo:



* + **Caminho mais curto**

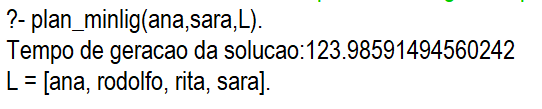


O predicado plan\_minlig/3, que recebe o nome do jogador inicial, o nome do jogador destino e a lista onde vai ser introduzida a solução, é o predicado chamado de modo a iniciar a solução. Este predicado utiliza o predicado melhor\_caminho\_minlig /2 que calcula qual é o caminho mais curto entre os dois jogadores. De seguida seleciona o caminho mais curto e apresenta-o.

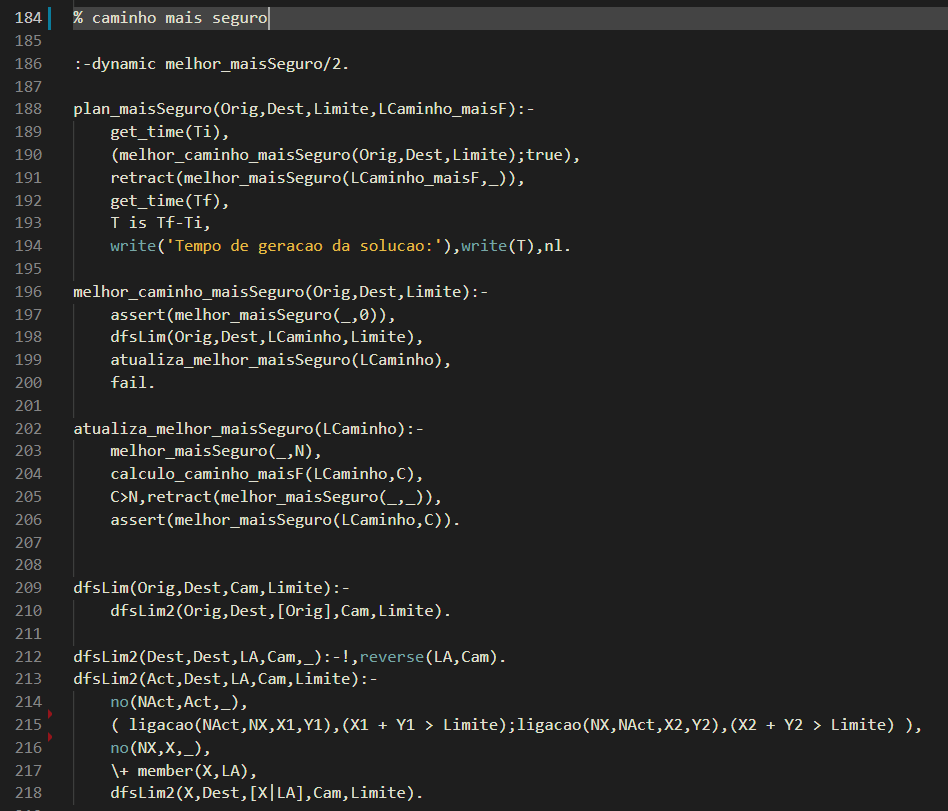
O predicado melhor\_caminho\_minlig /2, que recebe o nome do jogador inicial e o nome do jogador destino, inicia a variável dinâmica correspondente ao caminho mais curto e o respetivo tamanho (melhor\_maisF/2) com um tamanho de 0. De seguida utiliza o dfs/3 para calcular todos os caminhos possíveis entre os dois jogadores introduzidos, enviando então essa lista para o predicado atualiza\_melhor\_minlig/1 de modo a determinar o melhor caminho. O *fail* está posto para obrigar a voltar atrás e gerar um novo caminho.

O predicado atualiza\_melhor\_maisF/1, que recebe um caminho, começa por ir ao melhor caminho atual buscar o seu tamanho, de seguida convoca o predicado length/2 que calcula o tamanho do caminho introduzido, de seguida compara o tamanho do novo caminho com o tamanho do melhor caminho atual e se novo tamanho for menor o melhor caminho atual é substituído pelo novo caminho.

Exemplo:



* + **Caminho mais seguro**

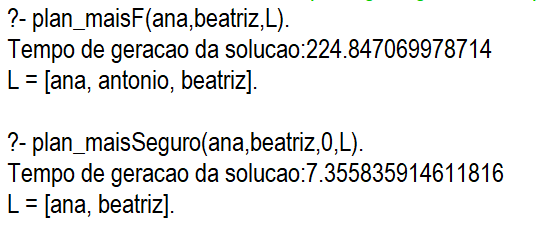


Para o cálculo do caminho mais seguro abordámos de forma idêntica ao cálculo do caminho mais forte, alterando somente o dfs.

Os predicados plan\_maisSeguro/4, melhor\_caminho\_maisSeguro/3, dfsLim/3 e dfsLim2/4 recebem agora o parâmetro Limite, cujas soma das forças da ligação tem que ser superior a esse limite.

Este novo DFS (dfsLim) apenas tem em consideração ligações cuja soma das forças é superior ao Limite introduzido.

Exemplo:



Neste exemplo verificamos que apesar do caminho mais forte ser Ana -> Antonio -> Beatriz, o caminho mais seguro é apenas Ana -> Beatriz, pois a ligação Antonio -> Beatriz tem uma força (-2) inferior ao limite imposto (0).

* **Estudo da complexidade do problema da determinação de caminhos**

|  |  |  |  |
| --- | --- | --- | --- |
| Nº de camadas intermédias | Nº de nós por camada | Nº de soluções | Tempo para gerar todos os caminhos (unidirecional) (s) |
| 1 | 1 | 1 | ~0 |
| 2 | 2 | 16 | ~0 |
| 3 | 3 | 81 | ~0 |
| 4 | 4 | 256 | ~0 |

|  |  |  |  |
| --- | --- | --- | --- |
| Nº de camadas intermédias | Nº de nós por camada | Nº de soluções | Tempo para gerar o caminho mais curto (s) |
| 1 | 1 | 1 | ~0 |
| 2 | 2 | 1 | ~0 |
| 3 | 3 | 1 | ~0 |
| 4 | 4 | 1 | 109 |

|  |  |  |  |
| --- | --- | --- | --- |
| Nº de camadas intermédias | Nº de nós por camada | Nº de soluções | Tempo para gerar o caminho mais forte (s) |
| 1 | 1 | 1 | ~0 |
| 2 | 2 | 1 | ~0 |
| 3 | 3 | 1 | ~0 |
| 4 | 4 | 1 | 134 |

|  |  |  |  |
| --- | --- | --- | --- |
| Nº de camadas intermédias | Nº de nós por camada | Nº de soluções | Tempo para gerar o caminho mais seguro (s) (limite = 0) |
| 1 | 1 | 1 | ~0 |
| 2 | 2 | 1 | ~0 |
| 3 | 3 | 1 | ~0 |
| 4 | 4 | 1 | 5 |

* **Conclusões**

A complexidade do algoritmo DFS é O(n) (sendo N a soma do número de nodes com o número de ligações). Sendo que cada algoritmo de seleção do melhor caminho necessita de uma pesquisa dentro do caminho para determinar a força de ligação ou o comprimento do caminho, a complexidade de cada algoritmo é O(n^2).

A complexidade revela-se relevante quando o grafo ultrapassa as 3 camadas intermédias com 3 nós por camadas.