# Relatório 1º projecto ASA 2019/2020

**Grupo:** al106/tp08

**Aluno:** Lourenço Duarte (97023)

---

## Descrição do Problema e da Solução

O problema apresentado trata a nova previsão de notas numa rede de alunos, tendo cada aluno uma previsão inicial. Esta nova previsão corresponderá ao máximo das previsões originais associadas a alunos a quais cada aluno tem acesso transitivamente via relações de amizade.

Desta maneira, dado um grafo dirigido G(V, E) em que os alunos são representados por vértices e as relações de amizade as arestas (as relações podem não ser recíprocas), e onde cada vértice tem um valor inicial retratando a previsão inicial da nota. Pretende-se que este valor seja atualizado para o valor dos seus vértices adjacentes ($\forall u \in V / \{v\} : (v,u) \in E$) caso estes tenham um valor superior. Desta forma, um vértice ficará com o valor final do máximo entre o seu valor inicial, os valores dos seus vértices adjacentes, os valores dos vértices adjacentes dos adjacentes, ect… Ficando, assim, com o maior valor atingível.

A solução proposta passa por descobrir todas as SCCs (Strong Connected Components) e inferir para cada uma delas qual é o seu valor máximo considerando os valores de todos os seus nós pertencentes e os valores das SCCs adjacentes. Foi, assim, utilizada uma variação do algoritmo Tarjan que à medida que vai descobrindo as SCCs vai atualizando o seu valor para o máximo dos seus vértices pertencentes e passando esse valor para as SCCs que as têm como adjacentes.

```
Tarjan_Visit(u, null, value)
    d[u] ← low[u] ← visited
    visited ← visited + 1
    Push(L,u)
    for each v ∈ Adj[u]
        value[u] ← max(value[u], value[v])
        do if (d[v] = ∞ || v ∈ L)
            then if d[v] = ∞
                then Tarjan_Visit(v, u, value)
            low[u] ← min(low[u],low[v])

    Scc ← 0
    if d[u] = low[u]
        then repeat
            v ← Pop(L)
            Push(Scc, v)
            until u = v

        #Update the value in this Scc to
        #the max of it modes's members
        value[Scc] ← -1
        for each s ∈ Scc
            value[Scc] = max(value[Scc], value[s])

        #Update the value of the parent if
        #it is bigger than the curr value
        if parent != null
            value[parent] ← max(value[parent], value[Scc])
```

```
SCC_Tarjan(G, value)
    visited ← 0
    L ← 0
    for each vertex u ∈ V[G]
        do d[u] ← ∞
    for each vertex u ∈ V[G]
        do if d[u] = ∞
            then Tarjan_Visit(u, null, value)
```

- Leitura dos dados de entrada (Não é apresentado no pseudo código):
    - Leitura do valor dos nós: simples leitura do input, com ciclo(s) a depender linearmente de V. **Logo, O(V)**
    - Leitura dos arcos: simples leitura do input, com ciclo(s) a depender linearmente de E. **Logo, O(E)**
- Chamada da função Tarjan_Visit: chamada simples da função que é baseada no algoritmo Tarjan. **Logo, O(V+E)**
- Apresentação dos dados (Não é apresentado no pseudo código): simples escrita para o output, com ciclo(s) a depender linearmente de V. **Logo, O(V)**

Complexidade global da solução: **O(V+E)**

**Avaliação Experimental dos Resultados**

Para verificar a linearidade do tempo de execução foram gerados 10 grafos com o número de vértices entre 1 x $10^6$ e 1 x $10^7$ e o número de arestas entre 1,5 x $10^6$ e 1,5 x $10^7$ tendo sido estes executados pelo algoritmo onde é possível concluir que o gráfico se encontra em concordância com análise teórica feita anteriormente.