# Relatório 2º projecto ASA 2019/2020

**Grupo:** al010
**Alunos:** Mónica Jin (92532) e Margarida Moreira (93881)

**Descrição do Problema:**

Neste relatório pretendemos analisar a solução proposta para o 2º projeto da cadeira Análise e Síntese de Algoritmos.

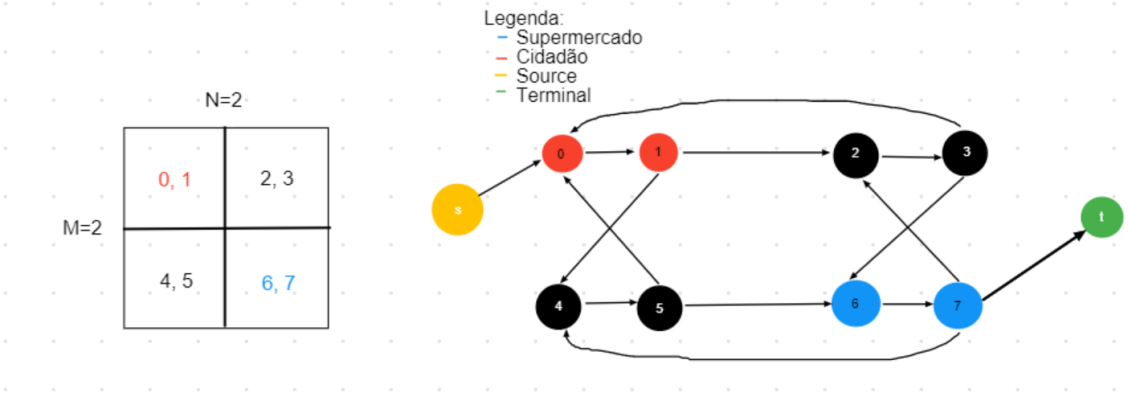
O problema apresentado tem como objetivo calcular o número máximo de cidadãos que pode deslocar-se a um supermercado sem se encontrar com outro cidadão ao longo do seu percurso. Os cidadãos e os supermercados estão localizados nas esquinas de uma grelha que representa a cidade de Manhattan, composta por ruas e avenidas. As ruas podem ser percorridas na direção Este-Oeste e as avenidas na direção Norte-Sul.

A informação inicial fornecida através do 'standard input' inclui o número de avenidas(M) e ruas(N), o número de supermercados(S) e cidadãos(C), e as coordenadas de cada supermercado e cidadão. A solução é apresentada no 'standard output' e indica o número de cidadãos que conseguem chegar a um supermercado.

**Descrição da Solução:**

A implementação do programa foi elaborada em linguagem C++.

Para resolver o problema apresentado, foi elaborado um mapeamento da cidade que a converteu num grafo dirigido. Este grafo foi representado através de listas de adjacências para cada vértice, de modo a limitar a memória usada.




Cada esquina da cidade passou a ser composta por dois vértices (um de entrada e um de saída). Em cada esquina, o vértice de saída permitia chegar a outras esquinas, enquanto que o vértice de entrada servia para limitar o fluxo de pessoas que entravam na esquina para apenas uma. A ligação entre cada esquina foi representada como uma aresta entre o vértice de saída da esquina inicial e o vértice de entrada da esquina final.

# Relatório 2º projecto ASA 2019/2020

**Grupo:** al010
**Alunos:** Mónica Jin (92532) e Margarida Moreira (93881)

Neste contexto, criámos um vértice *source* que estava ligado a todos os pares de vértices que representam os cidadãos, e os pares de vértices que representam os supermercados estavam ligados ao vértice *sink*. Assim, sintetizamos o problema num problema de fluxo máximo, *i.e.* o fluxo máximo encontrado para o grafo seria o número máximo de cidadãos que conseguem chegar aos supermercados. Para tal, implementámos o algoritmo Edmonds-Karp modificado que encontra o fluxo máximo do grafo.

Uma das principais alterações aplicadas ao algoritmo Edmonds-Karp foi a remoção de arcos pertencentes ao caminho de aumento e a adição dos arcos do caminho residual, enquanto que no algoritmo original as capacidades dos arcos eram diminuidas conforme o fluxo mínimo de aumento. Esta alteração deveu-se ao facto de todos os arcos terem implicitamente capacidade igual a 1, apesar de esta não ser representada no código do programa.

A escolha do algoritmo Edmonds-Karp (algoritmo de aumento de fluxo) deveu-se ao facto deste ser uma especificação do algoritmo de Ford-Fulkerson e portanto também ser limitado pela complexidade temporal O(E|f*|), isto é vantajoso pois o fluxo máximo é majorado pelas constantes de input S e C.

## Análise Teórica:

**Algorithm 1** Edmonds-Karp Modificado

```
procedure EDMONDS-KARP(G, s, t)
    maxFlow ← 0
    parent[V]
    path ← BFS(G, s, t, parent)
    while path exists do
        for v ∈ path do
            u = parent[v]
            delete edge(u,v) from G
            insert edge(v, u) in G
        maxFlow ← maxFlow + 1
    return maxFlow

procedure BFS(G, s, t, parent)
    for v ∈ visited do
        visited[v] ← false
    Q = ∅
    Enqueue(Q, s)
    visited[s] ← true
    parent[s] ← −1
    while Q ≠ ∅ do
        u = Dequeue(Q)
        for v ∈ G[u] do
            if visited[v] = false then
                Enqueue(Q, v)
                parent[v] ← u
                visited[v] ← true
    return (visited[t] = true)
```

# Relatório 2º projecto ASA 2019/2020

**Grupo:** al010
**Alunos:** Mónica Jin (92532) e Margarida Moreira (93881)

**Complexidade:**

- Leitura dos dados de input: O(S + C)
- Tratamento dos dados (conversão da grelha de Manhattan num grafo): O(MN)
- Aplicação do algoritmo do Edmonds-Karp: O(MN*min(S, C))
- Complexidade Total: O(MN*min(S, C))

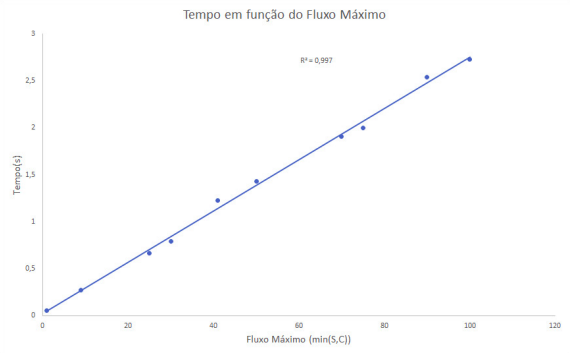
**Legenda:** M - número de avenidas; N - número de ruas; S - número de supermercados; C - número de cidadãos

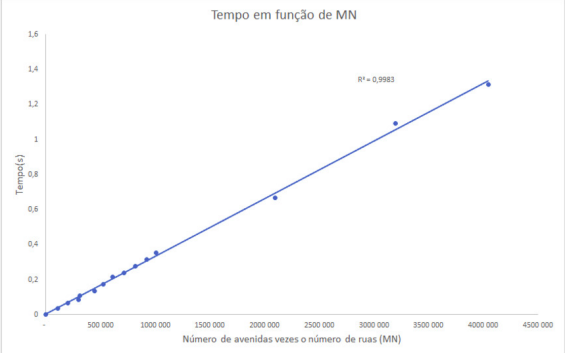
A complexidade do algoritmo usado (Edmonds - Karp) tem como limites assimptóticos $O(VE^2)$ e $O(E|f^*|)$, esta última complexidade deve-se ao facto do Edmonds-Karp ser uma especificação do algoritmo de Ford-Fulkerson. No contexto da solução apresentada, o valor de E é dado por 5MN -2M - 2N + S + C e o valor do fluxo máximo é majorado pelo mínimo entre S e C. Assim, o limite assimptótico de E é O(MN), uma vez que o número de supermercados e cidadãos é majorado pelo número de interseções entre avenidas e ruas, e o limite assimptótico do fluxo máximo é dado por O(min(S, C)). Para provar a complexidade foi escolhido o segundo limite assimptótico (O(E|f*|)) por ser possível majorar o fluxo máximo e pela simplicidade da expressão matemática obtida.

**Análise Experimental:**

De modo a comprovar a complexidade total O(MN*min(S, C)) proposta na análise teórica foram efetuados testes variando MN e variando min(S,C), mantendo sempre um deles constante.

Os gráficos, abaixo representados, variam o tempo de execução do programa em função do fluxo máximo e do MN, respetivamente.



# Relatório 2º projecto ASA 2019/2020

**Grupo:** al010
**Alunos:** Mónica Jin (92532) e Margarida Moreira (93881)

Como está ilustrado no gráfico da esquerda, a dependência entre tempo de execução e o fluxo máximo é linear, tendo $R^2$ um valor aproximadamente igual a 1 (0,997) para a regressão linear. É de notar que experiência foi executada com o MN constante.

Como está ilustrado no gráfico da direita, a dependência entre tempo de execução e o MN é linear, tendo $R^2$ um valor aproximadamente igual a 1 (0,9983) para a regressão linear. É de notar que experiência foi executada com o fluxo máximo constante.

Em conclusão, conseguimos comprovar a complexidade O(MN*min(S, C)) a partir dos resultados obtidos nos gráficos acima, uma vez que ambas as variáveis dependentes(tempo de execução) variam linearmente em função da sua variável independente (min(S, C) para o gráfico 1 e MN para o gráfico 2).