# Relatório 2º projecto ASA 2020/2021

**Grupo:** al030

**Aluno(s):** Mara Alves (95625) e Margarida Rodrigues (95627)

---

**Descrição do Problema e da Solução**

Este problema consiste em encontrar a atribuição de processos de um programa a processadores tal que o custo total de execução do programa seja mínimo, sendo que para além do custo de executar um certo processo num dos processadores, temos também de ter em conta o custo de comunicação entre processos executados em processadores diferentes.

Interpretámos este problema como um problema de fluxo máximo, e representámos o grafo como uma matriz de adjacências. Definimos como *source* da rede de fluxo um vértice X, que representa o processador X; e como *sink* o vértice Y, ou seja, o processador Y. Os restantes vértices são os *n* processos. Na rede de fluxo há arestas de X para cada processo, sendo a capacidade o custo de executar esse processo no processador X; e do processo para Y, com capacidade correspondente ao custo de executar o processo em Y. Por fim, existem também arestas entre processos, com capacidade igual ao custo de comunicação entre os dois processos. Se há aresta do processo i para o processo j com custo c, então também há uma aresta de j para i com o mesmo custo na nossa representação. Atualizamos as capacidades das arestas após calcularmos o valor do fluxo de cada caminho de aumento, subtraíndo ao valor que está na matriz o valor do fluxo e somando esse mesmo valor à capacidade da aresta em sentido inverso.

Fontes:

- https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/
- https://www.sanfoundry.com/cpp-program-implement-edmonds-karp-algorithm/

**Análise Teórica**

- Leitura dos dados de entrada, e simultânea construção da matriz representativa do grafo:
  Valor de *n* e *k*: leitura de apenas uma linha.
  Capacidades de X para processos e de processos para Y: leitura de *n* linhas.
  Capacidades entre processos: leitura de *k* linhas.
  Logo, O(n+k)
- Aplicação do algoritmo de Edmonds-Karp para encontrar valor do fluxo máximo:
  Ford-Fulkerson, com o caminho de aumento dado pela BFS.
  A complexidade da BFS é O($V^2$), e como V = n+2, O($n^2$), porque o ciclo while executa no máximo V vezes e o ciclo for interior executa também V vezes em cada iteração, por termos representado o grafo como uma matriz.
  O algoritmo de Edmonds Karp pode encontrar no máximo EV caminhos de aumento.
  Logo, O($EV^3$), o que é O($n^4$) pois E = 2(k+n)
- Apresentação do valor do fluxo máximo. Logo, O(1)

Complexidade global da solução: O($n^4$)

# Relatório 2º projecto ASA 2020/2021

**Grupo:** al030

**Aluno(s):** Mara Alves (95625) e Margarida Rodrigues (95627)

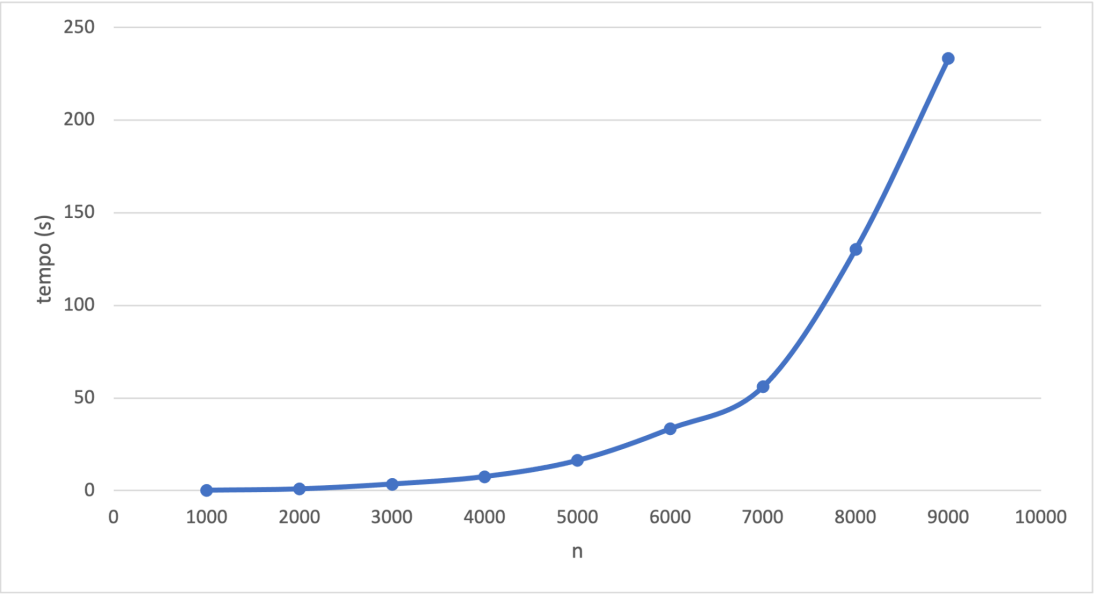
---

**Avaliação Experimental dos Resultados**

Corremos o nosso programa com o comando *time*, e como input utilizámos grafos gerados pelo gerador de instâncias fornecido. Testámo-lo com vários *n* diferentes e obtivemos os seguintes resultados:

| *n* | Tempo (s) |
|---|---|
| 1000 | 0,114 |
| 2000 | 0,830 |
| 3000 | 3,468 |
| 4000 | 7,499 |
| 5000 | 16,243 |
| 6000 | 33,347 |
| 7000 | 56,051 |
| 8000 | 130,345 |
| 9000 | 233,265 |

Sabendo que a complexidade desta solução é $O(n^4)$, o comportamento assintótico predominante deste algoritmo em função ao tamanho do conjunto de dados a ser processado é polinomial de grau 4. Sendo assim, o gráfico previsto para esta função, nos eixos positivos, é crescente. Analisando o gráfico gerado com os nossos valores obtidos...




Observamos que o gráfico está de acordo com a nossa previsão, pois comporta-se como uma função polinomial de grau 4.