**Descrição do Problema e da Solução**

O problema proposto passava por, dado um grafo G=(V,E), calcular o valor máximo de trocas comerciais minimizando os custos de infraestrutura, isto é, calcular o peso máximo do caminho que passa por todos os vértices com o menor número de arestas. Após a leitura do enunciado, a nossa ideia foi imediatamente utilizar o algoritmo de Kruskal para encontrar a Maximum Spanning Tree do grafo G e a partir desta calcular o peso.

Podemos concluir que a nossa abordagem inicial funcionaria, porém tínhamos de chegar a uma solução eficiente. Para tal, decidimos fazer o algoritmo utilizando também conjuntos disjuntos com “union by rank” (sempre que unimos dois conjuntos, ligamos a árvore mais pequena à raiz da árvore maior) e “path compression” (todos os elementos do set podem ser filhos da raiz para que sempre que usamos a operação “find()” não haja chamadas recursivas desperdiçadas e uma maior eficiência). Para tal, fizemos uma estrutura “Forest” com dois vetores com tamanho V, um para guardar os vértices pai (inicialmente é o próprio vértice) e outro para guardar o rank (inicialmente é -1). Por fim, utilizamos também um vetor de tamanho E para guardar “Edges”, estrutura que continha os 2 vértices e o peso de uma aresta no grafo G.

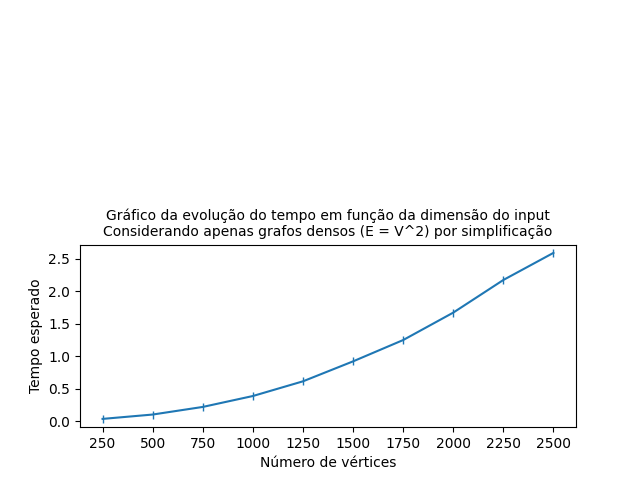
**Análise Teórica**

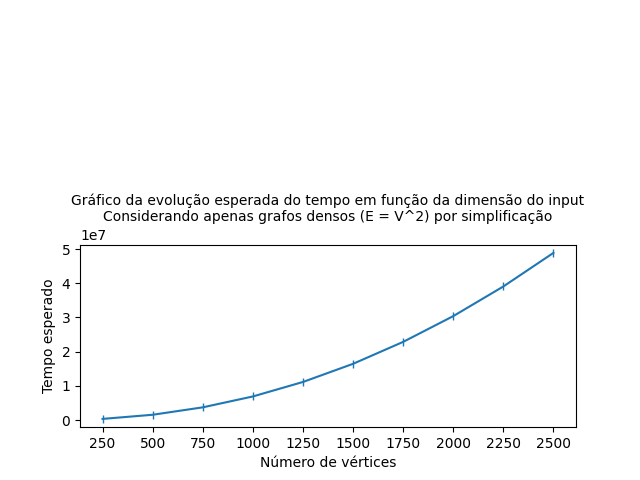
* Leitura dos dados do grafo: V e E  - O(1)
  + Pedido de input do número de vértices e arestas.
* Criação da estrutura “Forest” para representação dos conjuntos disjuntos - O(V)
  + Ciclo a depender linearmente de V para a inicialização dos vetores, representantes dos vértices pai e do rank.
* Leitura dos dados das arestas e inicialização do vetor de arestas. - O(E)
  + Pedido de input dos 2 vértices e peso de cada aresta e armazenamento dos mesmos num vetor.
* Aplicação do algoritmo de Kruskal para encontrar a Maximum Spanning Tree - (E log E)
  + Aplicação do algoritmo de sort nativo da biblioteca do C++ para ordenar as arestas de forma decrescente de peso. O(E log E)
  + Ciclo a depender de E e a depender das operações find() e unionset() dos conjuntos disjuntos. O(V log V).
    - Operação find() para cada um dos vértices. O(V log V)
    - No caso dos pais de cada vértice serem diferentes, unir os sets a que cada um pertence através da operação unionset(). O(V log V)
* Apresentação dos dados. O(1)
  + Imprime o valor máximo de trocas comerciais.

Complexidade global da solução: O(E log E). (para grafos densos)

**Avaliação Experimental dos Resultados**

Para testar o nosso programa e a nossa previsão teórica do tempo de execução geramos os dois gráficos que se seguem.





Após a análise dos gráficos, podemos concluir que a previsão teórica do tempo de execução do programa para grafos densos estava correta uma vez que a linha do crescimento esperado e a linha do crescimento real são muito semelhantes.