Relatório 1º projeto ASA 2020/2021

Alunos: João Vasco (95611), Maria Almeida (95628)

Descrição do Problema e da Solução

O problema apresentado tem por objetivo analisar o número mínimo de intervenções necessárias para garantir que todos os dominós caem e o tamanho da maior sequência de dominós a cair. Para o resolver, utilizámos linguagem C.

Para determinar o número mínimo de intervenções necessárias para garantir que todos os dominós caem, somámos todos os vértices que são sources (sem arestas apontar para eles).

Para determinar o tamanho da maior sequência de dominós a cair, realizámos uma ordenação topológica dos vértices, onde também fomos atribuindo pesos aos vértices, conforme fosse o tamanho máximo percorrido até eles. Seguidamente, fomos procurar o maior peso na lista de pesos, e, assim, obtemos o resultado desejado.

O algoritmo onde nos baseámos, é o seguinte:

https://www.geeksforgeeks.org/topological-sorting-indegree-based-solution/

Análise Teórica

- Leitura dos dados de entrada: scanf(pai,filho) para todas as edges
  O(E)- percorre todas as edges, dadas como input

- Processamento do grafo para fazer alguma coisa:

  Foi tudo guardado numa estrutura de vértices, onde cada vértice guardava, o seu índice, numero de filhos, uma lista de filhos, bool (se tinha pais), bool(se estava visitado).

  Para percorrer a lista dos filhos de cada vértice a complexidade será O(V+E), pois é necessário encontrar o vértice O(V) e posteriormente o seu filho adjacente O(E).

- Aplicação do algoritmo X para fazer algo: Order[] = TopologicalSort(){}

  O(V+E)- Algoritmo Topological Sort com complexidade V+E, dado que percorre todos os vértices uma vez, e tem em conta todas as edges adjacentes a cada um.

- Transformação dos dados com uma dada finalidade:

  For vertice in Order

  For filho in vertice.filhos{

  if(distancias[vertice]>= ditancias[filho])

  distancias[filho]=distancias[vertice]+1

  O(V+E)- Percorrer filhos de vértices

  For peso in distancias{

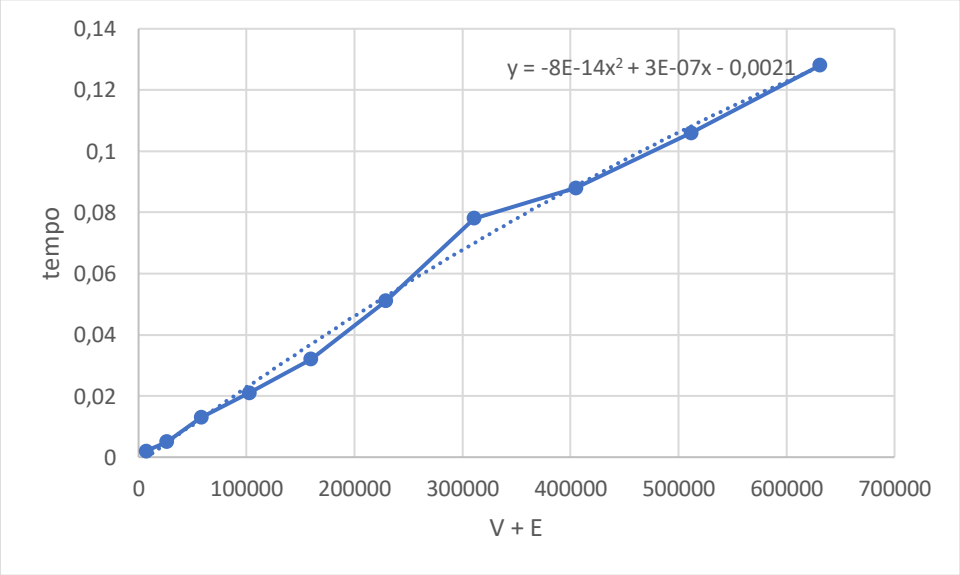
  if (peso > melhor_dist) { melhor_dist = peso }

  O(V)- Pois são percorridos todos os vértices observando qual é o que obtém o maior caminho

- Apresentação dos dados: print(melhor_dist)
  O(1)
- **Complexidade total = max(complexidades) = O(V+E)**

Avaliação Experimental dos Resultados

Para podermos testar a nossa análise teórica, selecionámos 10 grafos e determinámos o tempo necessário para percorrer cada um. Estes grafos foram corridos num processador I7-7700 3.6GHZ com (2x8)Gb Ram 2400MHz.



Como o algoritmo que foi usado em maior parte do processamento foi a ordenação topológica, que tem complexidade linear O(V+E), o gráfico obtido também aparenta ter complexidade linear (ordem 10^{-14}), apesar do pequeno desvio no meio. **Concluímos, assim, que o gráfico está em concordância com a análise teórica anteriormente prevista.**