Relatório 1º projecto ASA 2020/2021

Grupo: al135

Aluno: João Santos Jorge (88079)

Descrição do Problema e da Solução

As sequências de dominós construídas pelo professor João Caracol podem ser vistas como grafos dirigidos acíclicos (DAG) com 'n' sources. Pretende-se saber o número mínimo de intervenções necessárias para garantir que todos os dominós caem e também qual o tamanho da maior sequência de dominós a cair.

Para estruturar cada sequência de dominós construída, foi criada uma lista de adjacências. Cada linha da lista de adjacências indica quais os vértices que são filhos do respetivo índice, que é o pai. Depois da estrutura bem formada, é necessário calcular as sources. Percorre-se então cada linha da lista de adjacências e guardam-se os elementos que não aparecem em nenhuma das linhas. Esses elementos são as sources. E sabendo quais são, também se sabe quantas são.

Para calcular o tamanho da maior sequência de dominós (o caminho mais longo), utilizou-se a procura 'depth first search' (iterativa) e assim obter a ordem topológica. Uma vez obtido o vetor com a ordem topológica (que garante que todos os filhos de dado um elemento estão sempre à sua direita no vetor) processa-se cada elemento desse vetor por ordem. Processar um elemento do vetor consiste em verificar se o custo dos seus filhos é maior ou menor que o custo do elemento a ser processado. Caso seja menorou igual, atualiza-se o custo do filho (o custo do filho atualizado será então o custo do pai +1). Depois de processados todos os elementos da ordem topológica retira-se o maior custo. É importante notar que o vetor que guarda o custo é inicializado a '1'.

Referências

https://en.wikipedia.org/wiki/Longest_path_problem

https://www.geeksforgeeks.org/iterative-depth-first-traversal/

https://www.geeksforgeeks.org/find-longest-path-directed-acyclic-graph/

[também foram utilizados os slides das aulas teóricas]

Análise Teórica

- Leitura dos dados de entrada: simples leitura do input com ciclo a depender linearmente do número de dependências entre 2 elementos do grafo, ou seja, a depender do número de edges, O(E). Para adicionar uma ligação ao grafo, a complexidade é O(1).
- Processar o grafo para descobrir sources: ciclo que percorre a lista de adjacências, V indica o número de vértices e E o número de arestas. Logo, O(V + E). Sources são guardadas num vector auxiliar com o nome 'sources'.
- Aplicação do algoritmo Depth First Search (DFS) iterative para obter a ordem topológica. Logo, a complexidade é O(V + E).
- Processamento dos vértices segundo a ordem topológica para obter o maior caminho. Cada vértice compara o próprio custo com os seus sucessores. Logo O(V + E).

Complexidade global da solução: O(V + E)

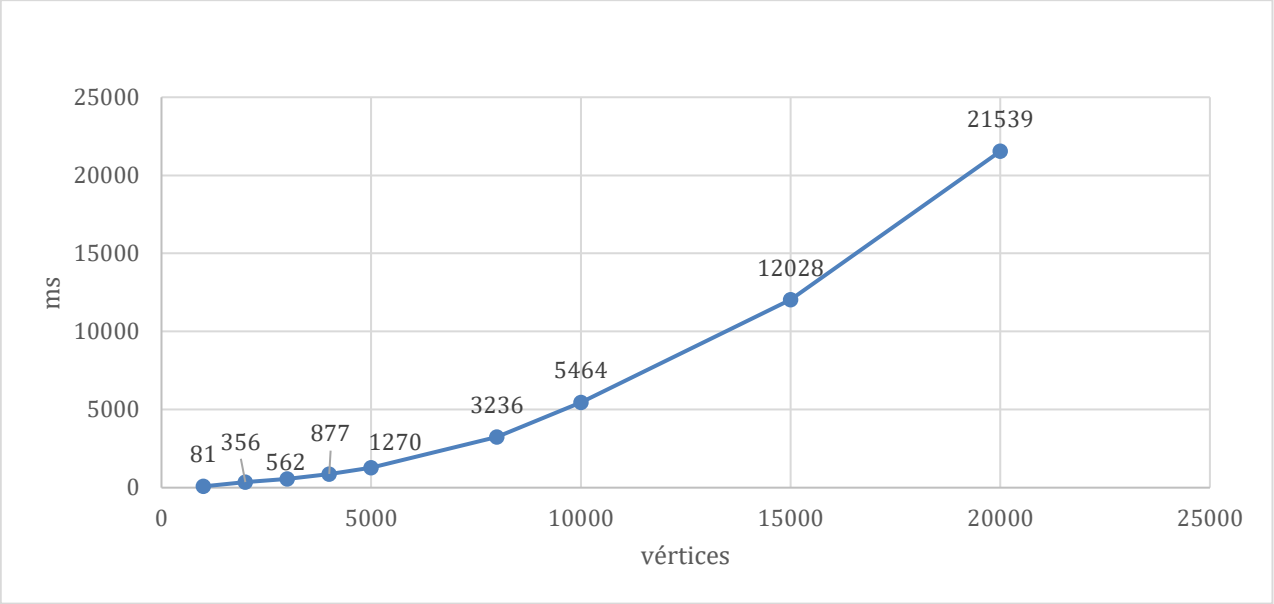
Relatório 1º projecto ASA 2020/2021

Grupo: al135

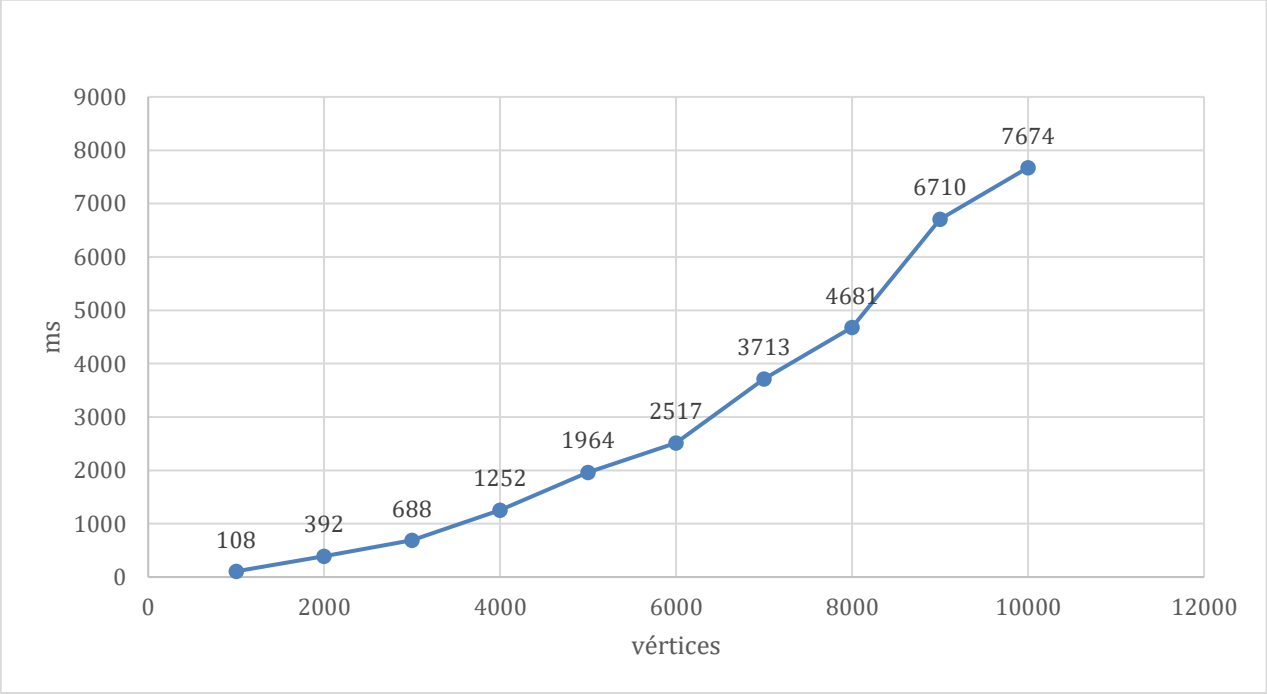
Aluno: João Santos Jorge (88079)

Avaliação Experimental dos Resultados

Foram feitas 2 experiências. Na primeira, foram construídos grafos com o fator = 0.7. Na segunda experiência, o fator = 1. Quanto maior o fator mais denso é o grafo (sendo sempre um DAG), ou seja, tem mais aresta.



Experiência 1



Experiência 2

Relatório 1º projecto ASA 2020/2021

Grupo: al135

Aluno: João Santos Jorge (88079)

Pela análise dos gráficos podemos constatar que a complexidade feita na análise teórica parece correta O(V + E). Uma vez que o computador pode estar a correr outros processos enquanto executa o algoritmo, o tempo verificado pode ter um pequeno desvio em relação ao valor real.