Relatório 1º projecto ASA 2020/2021

Grupo: al090

Aluno(s): Sara Marques (93342)

Descrição do Problema: Este projeto tem dois objetivo: determinar qual o número mínimo de intervenções que se tem de fazer para deitar abaixo todos os dominós numa cadeia de dominós e determinar o tamanho da maior sequência de dominós a cair numa intervenção.

Descrição da Solução: A solução proposta foi desenvolvida em C++. A sequência de dominós é representada por um grafo dirigido acíclico onde os dominós são os vértices e a relação entre dois dominós os arcos. No código desenvolvido este grafo é representado por uma lista de adjacências.

Para resolver o problema usou-se o algoritmo de procura em profundidade (DFS) por ordem numérica para ordenar os vértices topologicamente, ou seja, por ordem decrescente do tempo em que são terminados (todas as suas adjacências estão terminadas). Após termos a lista ordenada dos vértices conseguimos obter os dois valores pedidos, uma vez que temos uma sequência de dominós baseada nas dependências de cada um. Para o número mínimo de intervenções usa-se novamente o algoritmo da DFS mas desta vez pela ordem que acabámos de obter. Assim, vai-se realizar uma DFS apenas para os vértices que necessariamente se vão ter de deitar abaixo para que todos os dominós caiam. Somando o número de DFSs realizadas temos o número mínimo de intervenções necessárias. Para o tamanho da maior sequência de dominós percorre-se a lista ordenada dos vértices por ordem inversa, ou seja, por ordem crescente de tempo de fim. Para cada vértice percorremos as suas adjacências verificando o tamanho máximo da sequência que cada origina e guardando-o. A iteração inversa na lista ordenada permite que isto funcione uma vez que a ordenação topológica nos garante que se tivermos um vértice *u*, então as adjacências de *u* vão ser obrigatoriamente exploradas primeiro que o próprio *u*. Assim, quando se explora o vértice *u*, já temos os valores das sequências máximas geradas por cada uma das suas adjacências calculado, por isso basta guardar o maior e somar 1, que é o próprio *u*. Após percorrer toda a lista temos o tamanho da maior sequência de dominós a cair.

Análise Teórica

Segue abaixo o pseudocódigo dos algoritmos implementados para o cálculo dos dois valores pedidos com a análise das suas complexidades.

Algorithm 1 Calcula número mínimo de intervenções a fazer para que todos os dominós caiam.

```
procedure GETNUMBEROFINTERVENTIONS(G, topologicalSort)   ▷ Complexidade: O(V + E)
    let interventions be a new integer
    for each vertex u ∈ G.V do                            ▷ Inicialização: O(V)
        u.color = white
    for each vertex u ∈ topologicalSort do                ▷ Percorrer os vértices: O(V)
        if u.color == white then
            DFS-VISIT(G, u)                               ▷ Chamada ao DFS-VISIT: O(E)
            interventions++
    return interventions
```

Relatório 1º projecto ASA 2020/2021

Grupo: al090

Aluno(s): Sara Marques (93342)

Algorithm 2 Calcula tamanho da maior sequência de dominós a cair.

```
procedure GETLONGESTSEQUENCE(G, topologicalSort)        ▷ Complexidade: O(V + E)
    let sequence be a new integer
    let max be a new array
    for each vertex u ∈ topologicalSort.reverse do      ▷ Percorrer os vértices: O(V)
        for each vertex v ∈ Adj[u] do                   ▷ Percorrer lista de adjacências: O(E)
            if aux < max[v] then
                aux = max[v]
        max[u] = aux + 1
        if sequence < max[u] then
            sequence = max[u]
    return sequence
```

Seja V o número de vértices e E o número de arcos num grafo, temos as seguintes complexidades:

- Leitura dos dados de entrada e criação do grafo (depende linearmente do número de arcos): O(E);
- Aplicação do algoritmo DFS para obter a ordenação topológica: O(V + E);
- Cálculo do número mínimo de intervenções necessárias: O(V + E);
- Cálculo do tamanho da maior sequência possível: O(V + E).

Complexidade global da solução: O(V + E).

Avaliação Experimental dos Resultados

Para confirmar a complexidade da solução obtida teoricamente realizaram-se vários testes a grafos com número de vértices mais arcos variável medindo-se o tempo de execução para cada um. Os testes foram realizados num computador com processador Intel® Core™ i7 1.10GHz e com 16GB de memória RAM. Os dados obtidos encontram-se no gráfico abaixo.

Como se pode observar pelo gráfico o tempo de execução cresce linearmente com o tamanho do gráfico o que comprova a análise teórica prevista.

Referências

- https://en.wikipedia.org/wiki/Topological_sorting