O problema do carteiro chinês

Orientador: Carlos Eduardo Ferreira
Gabriel Fernandes de Oliveira

Sumário

1 Grafos eulerianos

1.1 Soluções

1.1.1 Algoritmo de Hierholzer

O algoritmo de Hierholzer foca em encontrar e unir um conjunto de circuitos $C_1, C_2, \ldots, C_k$, tal que não existe aresta pertencente a dois circuitos diferentes, e que toda aresta pertence ao menos a um circuito. Tal união gera um circuito euleriano.

O algoritmo desenvolvido se baseia em percorrer o grafo euleriano com uma busca em profundidade, apagando todas as arestas percorridas nesta busca. Quando o vértice atual da busca não possui mais arestas para se percorrer, o mesmo é adicionado ao começo da trilha euleriana que está sendo construída.

Segue o pseudo código do algoritmo de hierholzer para encontrar uma trilha euleriana dado um grafo euleriano.

Algorithm 1 Solução de Hierholzer

```
1: função EULER_HIERHOLZER(u)
2:     para cada v em adj[u] faça
3:         apaga a aresta de u a v
4:         EULER_HIERHOLZER(v)
5:     Insere u no início de trilha_euleriana
```

Digamos que a função apresentada seja chamada com um vértice inicial v, pertencente a um grafo euleriano G. A busca segue visitando vértices e apagando arestas até chegar no primeiro vértice em que não existem mais arestas para se percorrer.

Em grafos que possuem um circuito euleriano, tal vértice será o próprio vértice inicial v, e podemos chamar o circuito formado de C_1. Enquanto que, grafos que possuem apenas uma trilha euleriana, tal vértice será aquele cujo grau de entrada é maior que o grau de saída.

Em ambos casos, após encontrar tal vértice sem arestas, a função retorna por todos os vértices percorridos, procurando algum que ainda possua arestas não percorridas, para que ela possa encontrar um novo circuito C_i e juntar o mesmo ao circuito, ou trilha, inicial, unindo ambas estruturas.

Apesar da união dos circuitos ser um procedimento complexo e possivelmente custoso, não é necessário manter o resultado de todas uniões a serem realizados durante o algoritmo, já que apenas estamos interessados no resultado final, a trilha euleriana. Para manter tal trilha, basta adicionar a uma

pilha todo vértice encontrado na busca quando o mesmo não possui mais arestas a serem percorridas.

Segue agora a implementação deste algoritmo em C++:

```
stack<int> trilha;
stack<int> adj[N];

void euler_hierholzer(int u){
        while(!adj[u].empty()){
        int v = adj[u].top();
        adj[u].pop();
        euler_dfs(v);
        }
        trilha.push(u);
}
```

Optou-se nesta implementação o uso de uma pilha para o armazenamento da lista de adjacências *adj*, pois assim é possível apagar facilmente as arestas já percorridas na busca.

Para que tal função retorne corretamente uma trilha euleriana é necessário que a primeira chamada de *euler_hierholzer* tenha como parâmetro um vértice de grau ímpar, no caso dos grafos não direcionados, ou o vértice cujo grau de saída é maior que seu grau de entrada, no caso dos digrafos sem circuito euleriano.

Como em cada iteração deste algoritmo uma aresta é deletada, o número máximo de iterações que ele pode realizar é limitado pelo número de arestas de um grafo, sendo assim o mesmo possui complexidade $\mathcal{O}(|E|)$.