

# Implementação de um Gerador de Subgrafos de um Grafo Completo em C++

Lucas Gualtieri F. E.<sup>1</sup>, Gabriel Quaresma<sup>1</sup>

<sup>1</sup> Pontifical Catholic University of Minas Gerais  
Coração Eucarístico – 30535-901 – Belo Horizonte – MG – Brazil

{lgualtieri, gabrielquaresma}@sga.pucminas.br

**Resumo.** Este documento descreve a implementação de um gerador de subgrafos de um grafo completo com  $n$  vértices, onde  $n$  é informado pelo usuário. A solução foi implementada em C++ utilizando uma lista linear personalizada. O programa gera todos os subgrafos possíveis e informa o número total de subgrafos gerados.

## 1. Introdução

Neste projeto, foi desenvolvido um gerador de subgrafos de um grafo completo com  $n$  vértices. A tarefa foi implementada em C++, e o programa permite ao usuário informar o valor de  $n$ , em que o grafo gerado terá  $n$  vértices e todas as arestas possíveis. O objetivo é gerar todos os subgrafos possíveis e calcular o número total de subgrafos gerados.

## 2. Metodologia

A implementação foi realizada utilizando C++ e fez uso de uma lista linear personalizada para armazenar os conjuntos de vértices e arestas. A função principal do programa é a geração do conjunto potência (*PowerSet*) dos vértices e arestas, a partir dos quais são gerados os subgrafos.

### 2.1. Estruturas de Dados

O código faz uso de duas estruturas de dados principais:

- **Vértices:** Uma lista linear de inteiros representando os vértices de um subgrafo.
- **Arestas:** Uma lista linear de listas lineares de inteiros, representando as arestas entre os vértices do subgrafo.

### 2.2. Funções

- **factorial(int n):** Calcula o fatorial de  $n$ .
- **subsetCombinations(int i, int n):** Calcula o número de combinações possíveis de  $i$  elementos a partir de um conjunto de  $n$  elementos.
- **PowerSet(LinearList<T> set, size\_t min, size\_t max):** Gera o conjunto potência (subconjuntos) de um conjunto dado. Permite especificar o tamanho mínimo e máximo dos subconjuntos gerados.
- **subgraphs(int N):** Calcula o número total de subgrafos de um grafo completo com  $N$  vértices.

### 3. Implementação

A função PowerSet implementada neste trabalho é uma solução original, desenvolvida para atender às necessidades do projeto. Nossa implementação foi projetada para lidar com a definição de tamanhos mínimos e máximos para os subconjuntos gerados. Essa personalização proporciona maior flexibilidade na geração dos subconjuntos, isso permitiu que o processo ficasse mais de perto das ideias de implementação do projeto. A seguir, apresentamos o código completo implementado para o projeto:

```
1 typedef LinearList<int> Vertices;
2 typedef LinearList<LinearList<int>> Arestas;
3
4 size_t subsetCombinations(int i, int n) {
5     return factorial(n) / (factorial(i) * factorial(n - i));
6 }
7
8 template <typename T>
9 LinearList<LinearList<T>> PowerSet(LinearList<T> set, size_t min = 0,
10 ↪ size_t max = 0x7fffffff) {
11     if (max == 0x7fffffff) max = set.size();
12
13     int N = set.size();
14
15     LinearList<LinearList<T>> powerSet(pow(2, N));
16
17     for (int i = min; i <= max; i++) {
18
19         LinearList<T> subset(i);
20
21         int currentIndex = 0;
22
23         size_t combinations = subsetCombinations(i, N);
24
25         for (int j = 0; j < combinations; j++) {
26
27             while (subset.size() < i) {
28
29                 if (currentIndex >= N) {
30                     if (subset.back() == set.back()) subset.pop_back();
31                     currentIndex = set.indexOf(subset.pop_back()) + 1;
32                 }
33
34                 subset.push_back(set[currentIndex++]);
35             }
36
37             powerSet.push_back(subset);
38
39             if (!subset.empty()) subset.pop_back();
40         }
41     }
42
43     return powerSet;
44 }
```

```

45 size_t subgraphs(int N) {
46
47     size_t number = 0;
48
49     for (int i = 1; i <= N; i++) {
50         number += pow(2, (i * i - i) / 2) * subsetCombinations(i, N);
51     }
52
53     return number;
54 }
55
56 int main() {
57
58     int N;
59
60     cout << "Digite o |V| do grafo completo: ";
61     cin >> N;
62
63     LinearList<int> set(1, N);
64
65     int i = 1;
66
67     cout << endl;
68
69     for (Vertices V : PowerSet(set, 1)) {
70
71         Arestas edgesSet = PowerSet(V, 2, 2);
72
73         for (Arestas E : PowerSet(edgesSet)) {
74
75             cout << "Subgrafo " << i++ << ": " << endl;
76             cout << "V = " << V << endl;
77             cout << "E = " << E << endl << endl;
78         }
79     }
80
81     cout << "O número de subgrafos gerados foi: " << subgraphs(N) <<
82         ↵ endl;
83 }

```

## 4. Resultados

O programa gera todos os subgrafos possíveis de um grafo completo com  $n$  vértices, onde  $n$  é informado pelo usuário. Para cada subgrafo, é exibido o conjunto de vértices e arestas que o compõem.

## 5. Conclusão

A implementação do gerador de subgrafos foi concluída com sucesso. O programa cumpre o objetivo de gerar e listar todos os subgrafos de um grafo completo com  $n$  vértices, além de calcular o número total de subgrafos gerados. A estrutura de dados utilizada facilitou a manipulação dos conjuntos de vértices e arestas, e o uso do conjunto potência foi essencial para a geração dos subgrafos.