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

Lucas Gualtieri F. E.[1], **Gabriel Quaresma** [1], **Pedro Rodrigues Alves** [1]

[1] Pontifical Catholic University of Minas Gerais
Coração Eucarístico – 30535-901 – Belo Horizonte – MG – Brazil

{lgualtieri, gabrielquaresma, pedro.alves.1446100}@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:

- **Vertices**: 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:

```
typedef LinearList<int> Vertices;
typedef LinearList<LinearList<int>> Arestas;

size_t subsetCombinations(int i, int n) {
  return factorial(n) / (factorial(i) * factorial(n - i));
}

template <typename T>
LinearList<LinearList<T>> PowerSet(LinearList<T> set, size_t min = 0,
 ↪  size_t max = 0x7fffffff) {

  if (max == 0x7fffffff) max = set.size();

  int N = set.size();

  LinearList<LinearList<T>> powerSet(pow(2, N));

  for (int i = min; i <= max; i++) {

    LinearList<T> subset(i);

    int currentIndex = 0;

    size_t combinations = subsetCombinations(i, N);

    for (int j = 0; j < combinations; j++) {

      while (subset.size() < i) {

        if (currentIndex >= N) {
          if (subset.back() == set.back()) subset.pop_back();
          currentIndex = set.indexOf(subset.pop_back()) + 1;
        }

        subset.push_back(set[currentIndex++]);
      }

      powerSet.push_back(subset);

      if (!subset.empty()) subset.pop_back();
    }
  }

  return powerSet;
}
```

```cpp
size_t subgraphs(int N) {

    size_t number = 0;

    for (int i = 1; i <= N; i++) {
        number += pow(2, (i * i - i) / 2) * subsetCombinations(i, N);
    }

    return number;
}

int main() {

    int N;

    cout << "Digite o |V| do grafo completo: ";
    cin >> N;

    LinearList<int> set(1, N);

    int i = 1;

    cout << endl;

    for (Vertices V : PowerSet(set, 1)) {

        Arestas edgesSet = PowerSet(V, 2, 2);

        for (Arestas E : PowerSet(edgesSet)) {

            cout << "Subgrafo " << i++ << ": " << endl;
            cout << "V = " << V << endl;
            cout << "E = " << E << endl << endl;
        }
    }

    cout << "O número de subgrafos gerados foi: " << subgraphs(N) <<
    ↪  endl;
}
```

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.