Algoritmos

Pedro Hokama

Fontes

- [*clrs*] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest e Clifford Stein.
- [*timr*] Algorithms Illuminated Series, Tim Roughgarden
- Desmistificando Algoritmos, Thomas H. Cormen.
- Algoritmos, Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
- Stanford Algorithms
  https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V
  https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZl7A5HEK6
- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende
- Conjunto de Slides do Professores Cid C. de Souza para a disciplina MO420

Qualquer erro é de minha responsabilidade.

Grafos

Dois ingredientes:

- Vértices, Nós (Vertex, Vertices, Nodes). Normalmente o conjunto de vértices é denotado por V
- Arestas (Edges) = pares de vértices.
  - Grafo não direcionado (undirected graph): **arestas** são pares não ordenados de vértices. Normalmente o conjunto de arestas é denotado por E.
  - Grafo direcionado (directed graph): **arcos** (arcs) são pares ordenados de vértices, o primeiro vértice é a *ponta inicial* (*tail*) e o segundo é a *ponta final* (*head*). Normalmente o conjunto de arcos é denotado por A.
  - Alguns autores usam arestas ou arcos para ambos os casos.
- Normalmente um grafo é denotado por $G = (V, E)$.

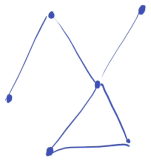
- Grafos aparecem não apenas na Ciência da Computação mas em várias áreas. Ciências sociais, biologia, engenharias etc.
- Exemplo: Mapas de ruas/estradas, Internet (páginas e links), Redes Sociais, Restrições de Precedência (disciplinas e pré-requisitos), etc.

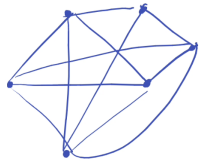
Considere um grafo não direcionado com n vértices, sem arestas paralelas e conexo (ou seja existe pelo menos um caminho entre qualquer par de vértices). Qual é o número mínimo e o número máximo de arestas que um grafo pode ter, respectivamente?

- $n-1$ e $n(n-1)/2$
- $n-1$ e n^2
- n e 2^n
- n e n^n

Grafos Esparsos e Densos

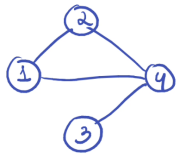
- Na maioria das aplicações $|E|$ é normalmente algo entre $\Omega(|V|)$ e $O(|V|^2)$
- Diremos que um grafo é esparso se $|E|$ estiver próximo de $O(|V|)$.
- Diremos que um grafo é denso se $|E|$ estiver mais próximo de $O(|V|^2)$

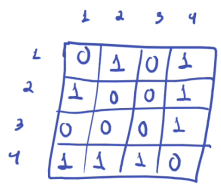




Representação de Grafos - Matriz de Adjacência

- Podemos representar um grafo G por uma matriz A de 0's e 1's de dimensão $|V| \times |V|$. Em que $A_{ij} = 1 \iff \{i,j\} \in E$





Variantes:

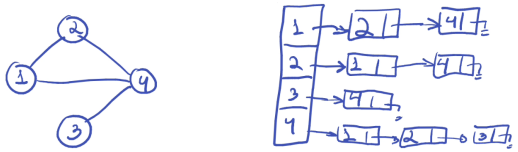
- A_{ij} pode ser o número de arestas se houverem arestas paralelas.
- A_{ij} pode ser o peso da aresta caso sejam diferentes.
- $A_{ij} = \begin{cases} +1 \text{ se a aresta vai de } i \text{ para } j \\ -1 \text{ se a aresta vai de } j \text{ para } i \end{cases}$

Matriz de Adjacência

- Para avaliar uma estrutura de dados, dois bons pontos importantes a considerar são:
  - A quantidade de memória que ele ocupa
  - e as operações que ela permite executar (e em que complexidade)
- Quanto espaço uma matriz de adjacência requer, em função do número de vértices e arestas.
  - $\Theta(|V|)$
  - $\Theta(|E|)$
  - $\Theta(|V| + |E|)$
  - $\Theta(|V|^2)$
- Se o grafo for denso, essa é uma boa representação, note que o espaço em memória não depende do número de arestas.
- Porém se $|E|$ for próximo de $O(|V|)$ essa representação desperdiça muita memória.

Lista de Adjacência

- Podemos representar o Grafo com listas de adjacências, em que armazenamos um vetor (ou lista) de vértices. E cada vértice tem uma lista para os seus vizinhos.



- Quanto espaço uma lista de adjacência requer, em função do número de vértices e arestas.
  - $\Theta(|V|)$
  - $\Theta(|E|)$
  - $\Theta(|V| + |E|)$
  - $\Theta(|V|^2)$

Qual a melhor estrutura?

- Depende da **densidade do grafo** e
- das operações que você ira realizar.
- Por exemplo, se você for executar um algoritmo que pergunta muitas vezes se um vértice i é vizinho de j? é mais fácil em uma matriz.
- Se você precisa percorrer todas as arestas? é mais fácil em uma lista de adjacência.
- Considere um grafo onde cada página é um vértice, numa estimativa bem conservadora temos 10 bilhões de páginas. Numa lista de adjacência temos $c10^{10}$, grande porém ainda pode ser tratado com computadores modernos. Agora 10^{20} já é impraticável.
- Por enquanto vamos considerar então que o nossos grafos serão representados em lista de adjacências.

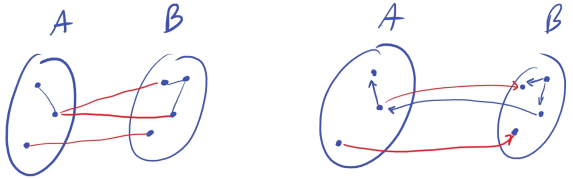
Corte Mínimo em Grafos

- Algoritmo Aleatorizado de Contração

Cortes em Grafos

Definição

Um corte de um Grafo $G = (V, E)$ é uma partição de V em dois conjuntos não vazios A e B.



Uma **aresta de corte** de um corte (A, B) são aquelas com

- um extremo em A e outro em B (em grafos não direcionados)
- com o inicio em A e o final em B (em grafos direcionados)

Cortes em Grafos

Quantos cortes (aproximadamente) um grafo com n vértices tem?

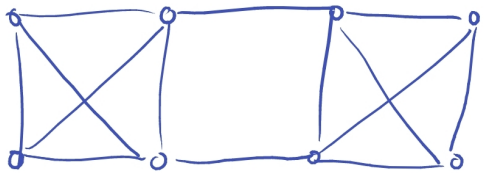
- n
- n^2
- 2^n
- n^n

O Problema do Corte Mínimo

Problema do Corte Mínimo

Dado um Grafo não direcionado $G = (V, E)$. Encontrar um corte com o menor número de arestas.

- O corte com o menor número de arestas em um grafo é o **corte mínimo**
- Arestas paralelas são permitidas.



Quantas arestas de corte tem um corte mínimo no grafo acima?

- 1
- 2
- 3
- 4

Cortes em Grafos

Aplicações:

- Identificar gargalos em redes (de computadores, de estradas, etc)
- Suponha que você queira redundância na sua rede para que a falha em uma (ou duas, ou mais) arestas principais não desconecte sua rede.
- Ou suponha que você quer sabotar um inimigo bombardeando uma estrada desconectando suas cidades.
- Detecção de comunidade em uma rede social. Grupos de usuários conectados entre si, mas com pouca conexão para o restante do grafo.
- Em reconhecimento de imagem, você pode entender cada pixel de uma imagem como um vértice com arestas para seus vizinhos, cada aresta com um peso de acordo com a semelhança dos dois pixels. Um corte minimo pode ser usado para encontrar objetos dentro da imagem.

Algoritmo Aleatorizado de Contração

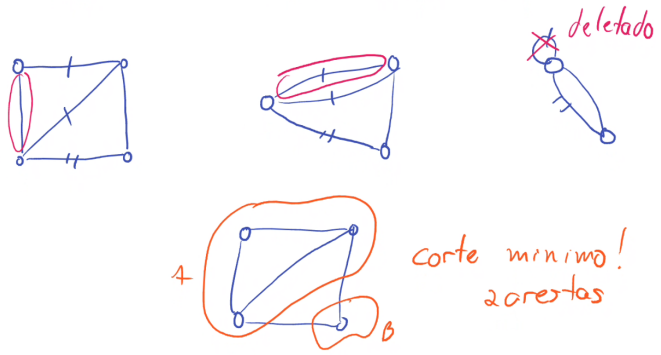
- Publicado por David Ron Karger, em 1993, quando aluno do Doutorado na Universidade de Stanford. (Hoje é professor no MIT)
- Ideia:

Algoritmo 1: Algoritmo Aleatorizado de Contração

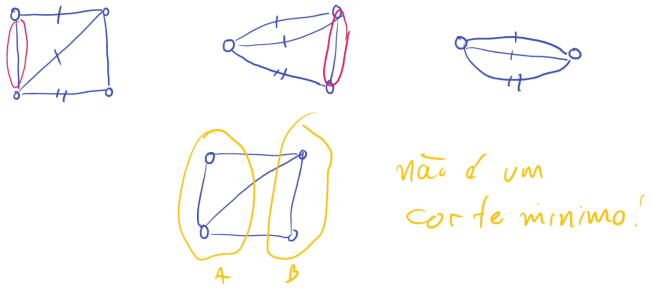
Entrada: Um Grafo G
Saída: Um Corte

1 **enquanto** *houverem mais de 2 vértices* **faça**
2 escolha uma aresta aleatória $\{u, v\}$ com probabilidade uniforme;
3 fundir (contrair) u e v em um único vértice;
4 remover auto-laços ;
5 **devolva** o corte representado pelos 2 vértices finais;

Exemplo - Execução



Exemplo - Execução2



- Algoritmo Aleatorizado de Contração as vezes encontra o corte mínimo, as vezes não.
- Será que um algoritmo assim é útil?
- A probabilidade de encontrar a resposta certa, é maior que zero, mas menos que 1. Mas qual será? 🤔