Métodos Exatos

Pedro Hokama

Fontes

- Curso Discrete Optimization no Coursera - Prof. Dr. Pascal Van Hentenryck
- [*clrs*] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest e Clifford Stein.
- [*timr*] Algorithms Illuminated Series, Tim Roughgarden
- Pesquisa Operacional, Arenales, M. N., Armentano, V. A., Morabito Neto, R., e Yanasse, H. H. (2015)
- Algoritmos, Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
- Stanford Algorithms
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V`
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZl7A5HEK6`
- Introdução à Otimização Combinatória, Flávio K. Miyazawa e Cid C. de Souza.
- Otimização Combinatória, da Carla Negri Lintzmayer do CMCC, UFABC

Qualquer erro é de minha responsabilidade.

Classes de Complexidade

- Formalmente as classes de complexidade P, NP, NP-completo e outras, são melhor definidas sobre os problemas de decisão, para os quais a resposta é simplesmente **SIM** ou **NÃO**. Mas os problemas tem uma forte relação entre si.
  - Caminho mínimo: Dado G, um vértice s e um número k existe um caminho de s para qualquer outro vértice com custo no máximo k?
  - MST: Dado G e um inteiro k, existe uma Árvore geradora mínima de custo no máximo k?
  - Compressão de Texto: Dado um texto T e um número k, existe uma compressão desse texto com no máximo k bits?

Classes de Complexidade

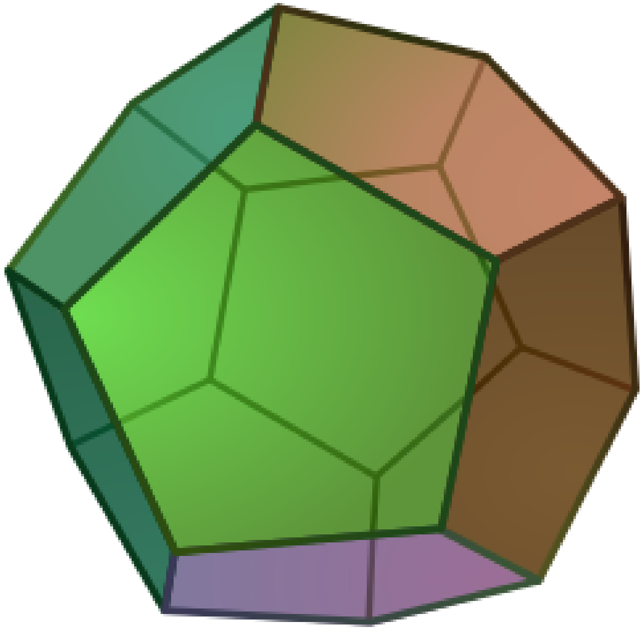
- O problemas que podem ser decididos em tempo polinomial formam a classe de complexidade

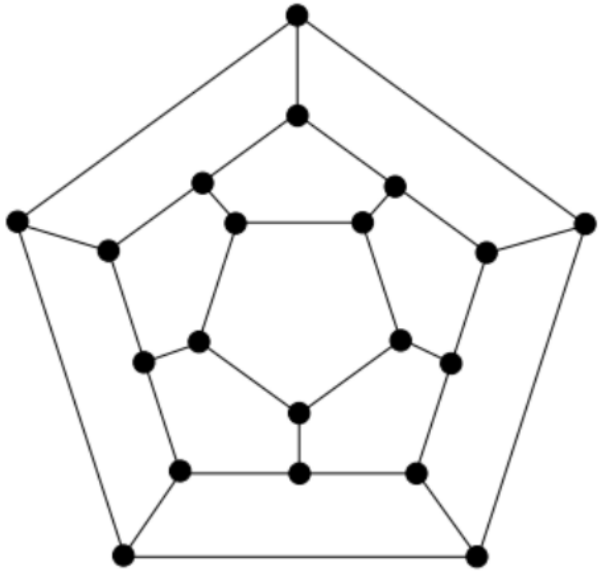
$$P$$

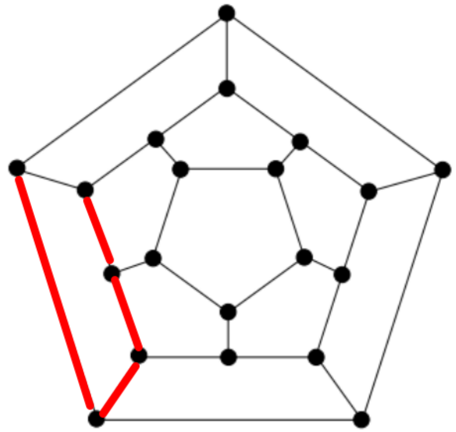
- Vários problemas possuem algoritmos eficientes para resolve-los
  - Ordenação - $O(n \log n)$
  - Multiplicação - $O(n^{1.586})$
  - Multiplicação de Matrizes - $O(n^{2.8})$
  - Busca em Grafos - $O(m + n)$
  - Encontrar Componentes fortemente conexas - $O(m + n)$
  - Caminhos Mínimos - $O(m \log n)$
  - Escalonamento Ponderado - $O(n \log n)$
  - Compressão de Texto - $O(n \log n)$
  - MST - $O(m \log n)$

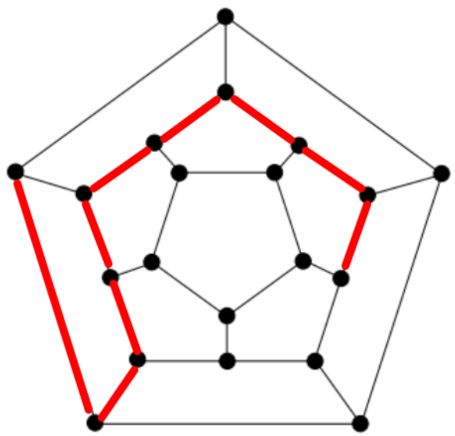
- Mas será que todos os problemas podem ser resolvidos em tempo polinomial?

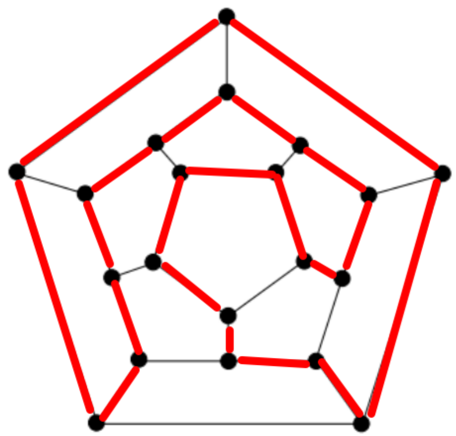
Sir William Rowan Hamilton (1805-1865)

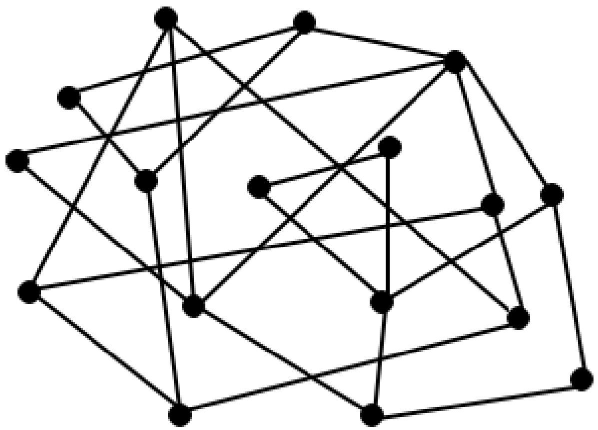


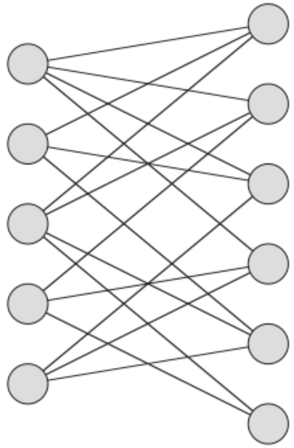












- Um **ciclo hamiltoniano** de um grafo G é um ciclo que passa por todos os vértices exatamente uma vez.

Problema do Ciclo Hamiltoniano

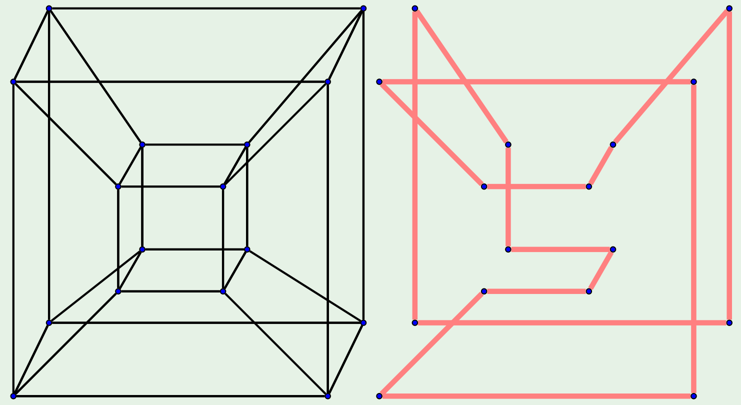
Dado um grafo não orientado $G = (V, E)$. Decidir se G possui um ciclo hamiltoniano.

- Não se conhece nenhum algoritmo de tempo polinomial para resolve-lo!

A classe *NP*

- **Definição:** Um problema A está em NP se para qualquer instância x de A para o qual a resposta seja "sim", existe um certificado y de tamanho polinomial, que pode ser verificado em tempo polinomial, provando que a resposta de x de fato é "sim".

Ciclo Hamiltoniano



- Dado um grafo G, existe uma árvore geradora de custo $\leq W$
  - Um certificado, é a própria árvore
- Dado um texto T, existe uma compressão que usa $\leq B$ bits
  - Um certificado é o próprio texto comprimido
- Todo problema em P está em NP
- Dado um conjunto de itens I e uma mochila de capacidade W, decidir se cabe na mochila um subconjunto de itens de valor $\geq K$.
  - Um certificado é a própria coleção de itens.

- Todo problema em NP pode ser resolvido por força-bruta em tempo exponencial.
- Gerar um número exponencial de soluções e verificar cada uma em tempo polinomial.
- A maioria dos problemas (computáveis) que encontramos está em NP

- **Definição:** Dada uma classe de problemas C um problema é C-**difícil** se ele é tão difícil quanto qualquer outro problema em C. Ou seja, existe uma redução de tempo polinomial de qualquer problema em C para ele.
- Um problema é C-Completo se é C-difícil e pertence a C

Reduções

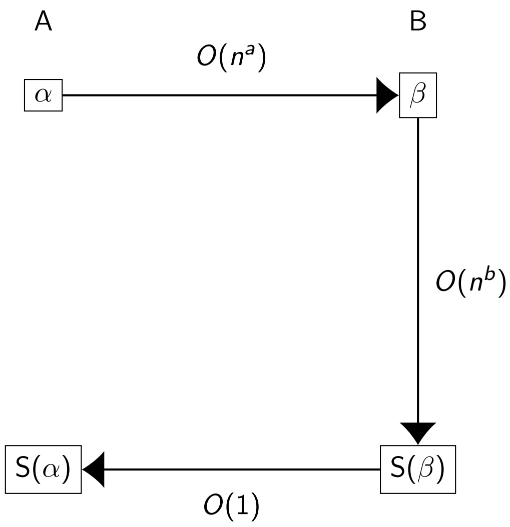
Suponha que temos um problema de decisão A que queremos resolver em tempo polinomial.

Agora suponha que temos outro problema de decisão B que já sabemos que pode ser resolvido em tempo polinomial.

Finalmente, suponha que conhecemos um procedimento que transforma uma instância α do problema A, em uma instância β no problema B. Tal que:

- A transformação leva tempo polinomial
- A resposta de α para A é a mesma da instância β para B.

Reduções

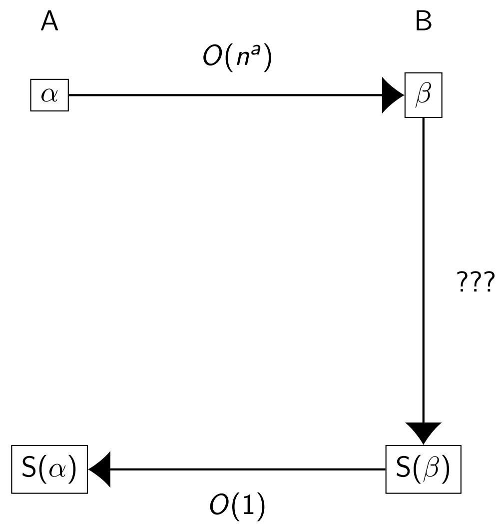


Reduções

- Temos então um algoritmo polinomial para resolver o problema A.

- Agora suponha que temos um problema A que sabemos que é difícil de resolver.
- e suponha que conseguimos uma redução de A para B que seja muito rápida.
- O que podemos afirmar sobre B?

Problema muito difícil



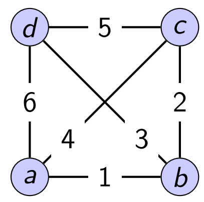
A classe NP-Completo

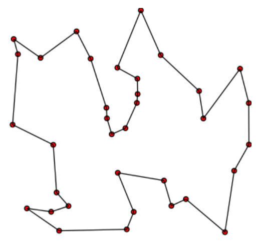
- Será que existem problemas NP-Completos?
- E se existirem será que existe um algoritmo polinomial para resolve-los?
- A maioria dos Ciências da Computação acredita que tal algoritmo não existe.

O Problema do Caixeiro Viajante

Travelling Salesman Problem - TSP

- Dado um grafo com custos nas arestas, encontrar um ciclo que passa por todos os vértices exatamente uma vez de custo mínimo.

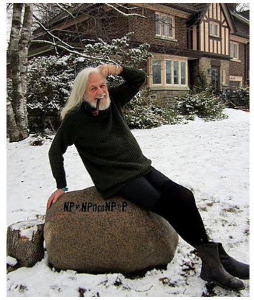




- Versão de decisão: Dado um grafo com custos nas arestas e um valor W, decidir se existe um ciclo que passa por todos os vértices exatamente uma vez de custo $\leq W$.

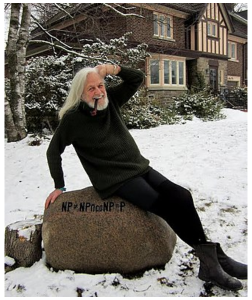
História

- Jack Edmonds é um Matemático e Cientista da Computação Estadunidense
- Graduado em 1957 na George Washington University e Pós-graduado na University of Maryland em 1959.



História

- Trabalhou no National Institute of Standards and Technology de 1959 até 1969
- Em 1965 em um artigo chamado "Paths, Trees and Flowers" conjecturou que não existe um algoritmo em tempo polinomial para o TSP.



História

- Stephen Arthur Cook é um cientista da computação e matemático Estadunidense-Canadense
  - Obteve em 1962 e 1966 seu mestrado e doutorado pela Universidade de Harvard
  - Em 1970 mostrou que existem Problemas NP-completos

História

- Leonid Levin é um matemático e cientista da computação Soviético
  - Obteve seu mestrado e doutorado em 1970 e 1972 na universidade de Moscou
  - Em 1972 mostrou a existência dos problemas *NP*-completos.
- O teorema de Cook-Levin afirma que o problema da satisfabilidade boleana é NP-Completo.

História

- Richard Karp é um cientista da computação Estadunidense.
- Obteve seu mestrado e doutorado em matemática aplicada em 1956 e 1959 em Harvard.

História

- Trabalhou na IBM e foi professor de ciência da computação e Pesquisa Operacional na Universidade da Califórnia, Berkeley.
- Mostrou 21 problemas que eram NP-completos. E a partir desses milhares foram provados.‘

Problemas *NP*-Completos

- Suponha então que você se deparou com um problema π, e tentou todas as técnicas que você aprendeu, mas não obteve um algoritmo polinomial.
- Mais uma ferramenta essencial para a nossa caixa: talvez seu problema seja *NP*-Completo. Receita para provar:
  1. Provar que ele pertence a *NP*
  2. Provar que ele é *NP*-difícil,
     - Escolha um problema π' que sabidamente é *NP*-completo
     - Faça uma redução (polinomial) de π' para π.

Problemas *NP*-Completos

- Qual problema π' escolher?
- 21 problemas de Karp
- Cap. 34 do CLRS
- Garey e Johnson, Computers and Intractability, 1979