**Relatório**

Algoritmos e Estruturas de Dados I

**Universidade de São Paulo**

Campus São Carlos

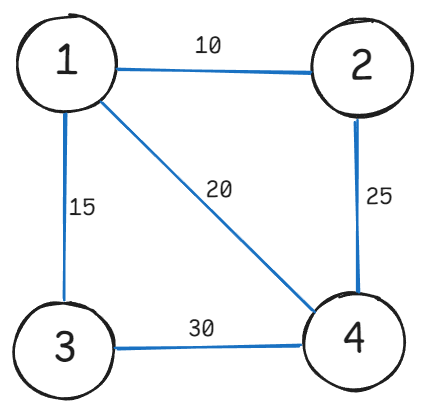
Alunos: Renan Trofino Silva, Guilherme Pereira de Sá, Augusto Fernandes Ildefonso

Professor: Rudinei Goularte

// Sumário.

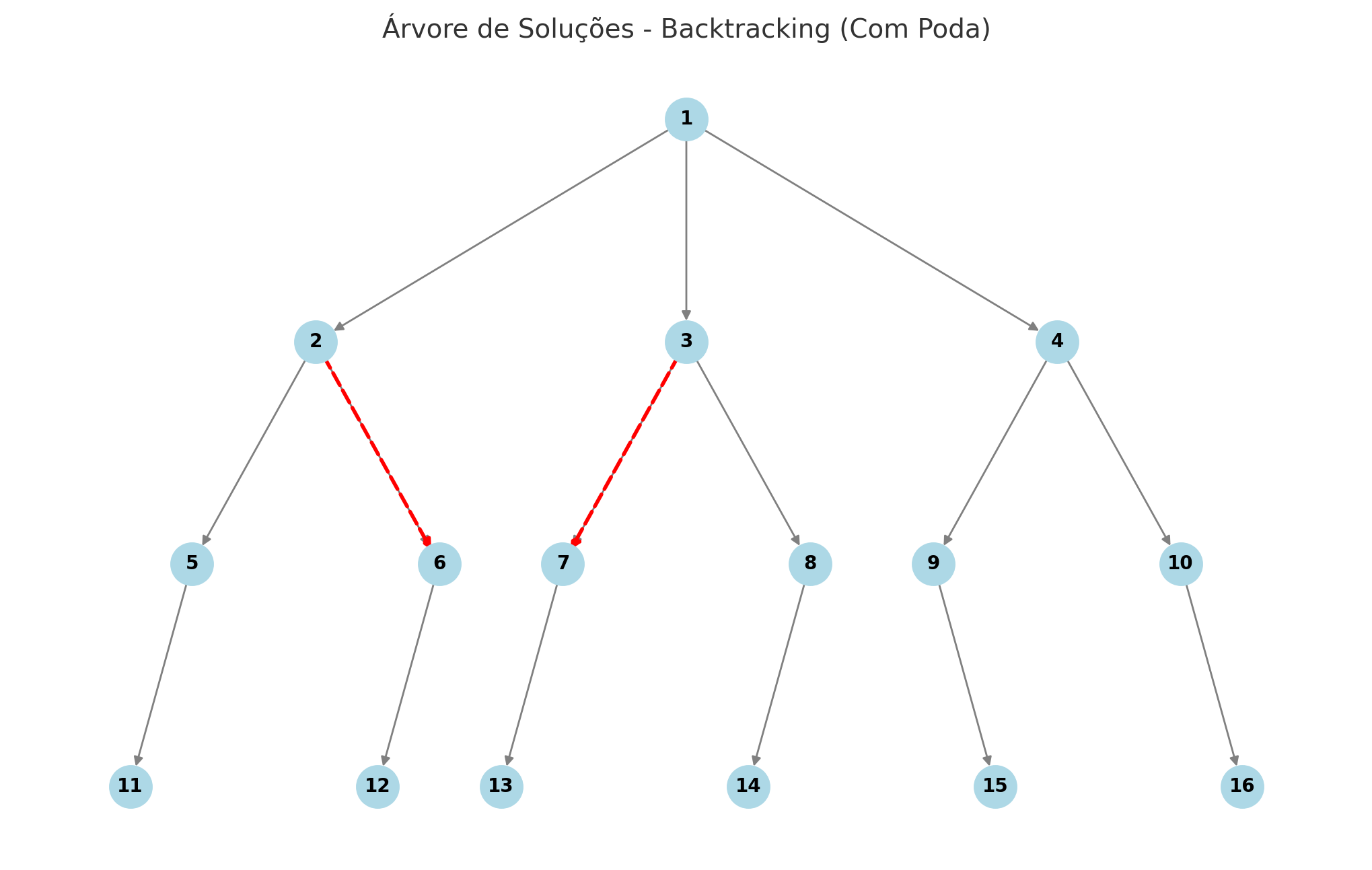
Introdução ao projeto.c

Na pasta-pai entregada para a resolução deste problema, *O Problema do Caixeiro Viajante*, onde precisamos optar pelo menor custo de distâncias que o viajante tem de fazer ao visitar cada uma das cidades, saindo, obviamente, da cidade de origem. Leremos de um arquivo externo as informações número de cidades, cidade de origem, número de arestas que o grafo possuirá, logo, o número de possibilidades que teremos, por exemplo:

Este grafo ao lado não possui aresta de 2 para 3.

Portanto o número de arestas é 5.

Backtracking, um Brute Force mais eficiente

A ideia aqui é de "podar", já que tratamos de uma árvore de "processos", escolhas, a serem tomadas por cada cidade, e o algoritmo ao analisar um "ramo" da árvore, percebendo que é pior que o anterior, ou seja, fixamos, tomamos o anterior, ou que aparecer, no nosso caso, ter o menor custo, como a melhor opção e apenas comparamos com o atual caminho seguido. E, sem algum caso, ultrapassar esse menor custo, já podemos retirar esse "ramo", essa possibilidade de caminho, como a que garantiria o menor custo. Então seguimos com a ideia do Brute Force, de analisar todas as combinações possíveis, porém, agora, retirando aquelas rotas que já começam com um custo acima do requerido.

Ao lado vemos como é representado o backtracking. E, como já falado anteriormente o backtracking remove podas que já não são uma solução local ótima. Então aqui teríamos um exemplo onde o custo de 1 – 2 – 6 e 1 – 3 – 7 já são maiores que o menor custo, ultrapassam algum limite pré-definido.

// colocar graph de eficiencia.py

O algoritmo começa na cidade de origem e marca-a como visitada. Ele recursivamente tenta visitar todas as cidades não visitadas, empilhando cada cidade visitada. Se uma rota completa (visitando todas as cidades) é formada, o algoritmo calcula o custo da rota incluindo o retorno à cidade de origem. Se o custo da rota atual for menor que o menor custo encontrado, essa rota se torna a nova melhor solução. O algoritmo desfaz o último movimento (backtrack), desempilhando a cidade e marcando-a como não visitada, para explorar outras rotas.

Cálculo de Complexidade

O algoritmo continua tendo o pior caso como o (n-1)!, porém, na prática, ele diminui significativamente os casos cedo, reduzindo número de caminhos que devem ser explorados. A eficiência do Backtracking se torna mais relevante quanto mais aumentarmos o número de entradas. Para provarmos nosso tempo, precisamos fazer uma análise empírica do backtracking comparando-a com o Brute Force.

// Possibilidade de explorar uma implementação para a análise empírica do backtracking

// time-back-perm.c

// colocar data de acesso.

Referências

* **Wikipedia.** Backtracking. Disponível em: <https://pt.wikipedia.org/wiki/Backtracking>. Acesso em: [data de acesso].
* **Wikipedia.** Problema do caixeiro-viajante. Disponível em: <https://pt.wikipedia.org/wiki/Problema_do_caixeiro-viajante>. Acesso em: [data de acesso].
* **Livro.** Algorithm Design Manual. Disponível em: <https://www.google.com/search?q=Algorithm+Design+Manual>. Acesso em: [data de acesso].
* **Livro.** Introduction to Algorithms. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, e Clifford Stein. Disponível em: [https://www.google.com/search?q=%22Introduction+to+Algorithms%22+-+Thomas+H.+Cormen%2C+Charles+E.+Leiserson%2C+Ronald+L.+Rivest%2C+e+Clifford+Stein](https://www.google.com/search?q="Introduction+to+Algorithms"+-+Thomas+H.+Cormen%2C+Charles+E.+Leiserson%2C+Ronald+L.+Rivest%2C+e+Clifford+Stein). Acesso em: [data de acesso].
* **GeeksforGeeks.** Backtracking Algorithms. Disponível em: <https://www.geeksforgeeks.org/backtracking-algorithms/?ref=header_outind>. Acesso em: [data de acesso].
* **GeeksforGeeks.** Travelling Salesman Problem Implementation Using Backtracking. Disponível em: <https://www.geeksforgeeks.org/travelling-salesman-problem-implementation-using-backtracking/?ref=header_outind>. Acesso em: [data de acesso].