Algoritmo de Programação Dinâmica (Held-Karp)

### **Descrição do Algoritmo**

O algoritmo de Held-Karp utiliza a técnica de programação dinâmica com bitmask para resolver o Problema do Caixeiro Viajante (PCV) de forma exata. A ideia principal é calcular o menor custo possível para visitar todas as cidades exatamente uma vez e retornar ao ponto de origem.

Para isso, o algoritmo mantém uma tabela memo[mask][last], onde:

* mask representa o conjunto de cidades já visitadas, usando uma máscara de bits (bitmask).
* last é a última cidade visitada.
* memo[mask][last] armazena o menor custo para visitar as cidades em mask terminando na cidade last.

A construção da solução ocorre utilizando **bottom-up**, que preenche a tabela com todas as combinações possíveis de cidades visitadas e suas transições. Ao final, o algoritmo retorna o menor custo total para completar o ciclo, depois disso ele reconstrói o caminho ótimo usando uma tabela auxiliar parent.

### **Complexidade Computacional**

A complexidade temporal do algoritmo Held-Karp é:

* 2^n: número de subconjuntos possíveis de cidades visitadas (máscaras).
* n: número de cidades para cada máscara (última cidade visitada), para cada subconjunto, é preciso considerar cada cidade como destino final.
* Outro n: vem das transições entre cidades ao tentar expandir a rota.

A complexidade espacial também é:

Por isso, esse algoritmo é prático apenas para instâncias com **até cerca de 20 cidades**.

### **Análise Experimental**

Para testar o desempenho do algoritmo, foram utilizadas várias instâncias do problema com tamanhos variados. É importante destacar que como as comparações foram feitas com o algoritmo de força bruta e guloso, tivemos que adaptar as instâncias a título de comparação para no máximo 10 cidades (tamanho suportado pela força bruta).

As instâncias foram geradas com cidades posicionadas aleatoriamente em um plano cartesiano. Para cada instância, foram medidos o tempo de execução e o custo total do percurso.

Esses dados foram usados para gerar um gráfico de comparação entre os diferentes algoritmos implementados no trabalho (força bruta, guloso e programação dinâmica (Held-Karp).

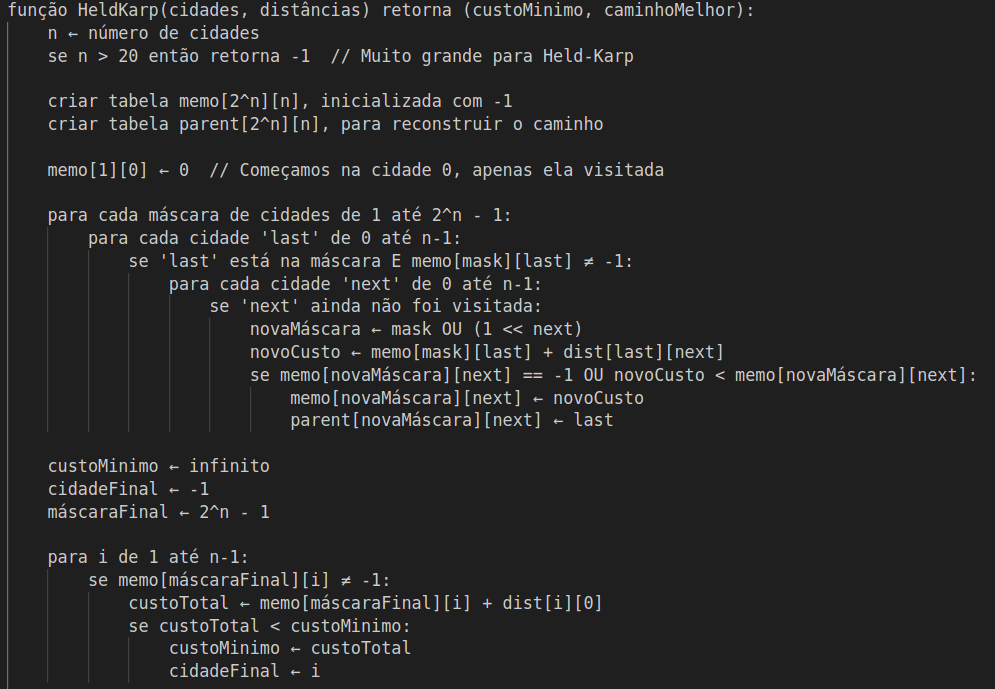
O algoritmo Held-Karp é uma solução exata eficiente para instâncias pequenas e médias do PCV (excelente para instâncias de até 20 cidades). Ele representa um ótimo equilíbrio entre tempo de execução e precisão da resposta, especialmente quando é comparado ao algoritmo de força bruta. No entanto, seu crescimento exponencial o torna inviável para instâncias maiores, onde soluções heurísticas ou aproximações são mais apropriadas.

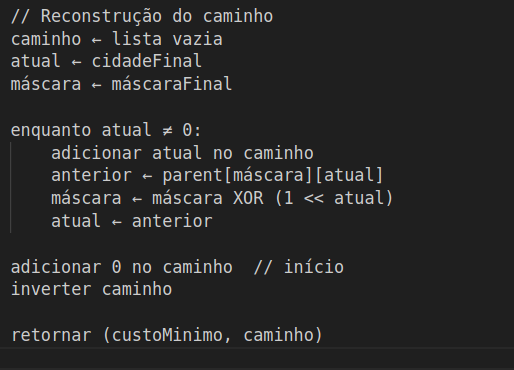
## **Explicação do Código em Portugol**

Para um melhor entendimento, a seguir existe uma adaptação do código em C para pseudocódigo.

1. **Inicialização:**Recebemos as cidades e a matriz de distâncias e criamos duas tabelas:  
   * memo: armazena o custo mínimo para cada conjunto de cidades visitadas e cidade final.
   * parent: armazena o caminho anterior para reconstrução da rota.
2. **Base da Programação Dinâmica:**
   * Começamos com apenas a cidade 0 visitada (mask = 1) e custo 0.
3. **Preenchimento da Tabela:**
   * Iteramos sobre todas as combinações de cidades visitadas (mask).
   * Para cada cidade last no conjunto atual, tentamos visitar novas cidades next.
   * Se a cidade next ainda não foi visitada, calculamos o novo custo e atualizamos a tabela memo se for melhor.
4. **Fechando o Ciclo:**
   * Após visitar todas as cidades (nesse caso a máscara está cheia), tentamos voltar para a cidade 0 a partir de todas as possíveis cidades finais e escolhemos a que tem menor custo.
5. **Reconstrução do Caminho:**
   * Usamos a tabela parent para refazer o caminho desde a última cidade até a cidade 0.
   * Adicionamos as cidades ao vetor caminho e o invertemos ao final.
6. **Último passo:**
   * Retornamos o custo mínimo e o caminho ótimo encontrado.

Algoritmo escrito em pseudocódigo para maior didática:





A implementação do algoritmo Held-Karp com programação dinâmica mostrou uma abordagem eficiente e precisa para resolver o Problema do Caixeiro Viajante em instâncias de tamanho moderado. A utilização de **bitmask** para representar subconjuntos de cidades visitadas, aliada à tabela de memorização, permite que o algoritmo evite recalcular os mesmos subproblemas, reduzindo drasticamente o número de operações em comparação com a força bruta.

Apesar de sua complexidade ser ainda exponencial (O(n² . 2^n)), o algoritmo é significativamente mais rápido que a abordagem exaustiva e permite resolver instâncias com até 20 cidades em um tempo aceitável. Além disso, por ser uma solução exata, o Held-Karp retorna sempre o melhor caminho possível, o que é vantajoso em contextos onde a precisão é essencial.

Durante os testes com diferentes instâncias, o algoritmo demonstrou consistência nos resultados e um crescimento previsível no tempo de execução, o que confirmou a análise teórica da complexidade. Assim, a programação dinâmica se apresenta como uma ótima alternativa entre desempenho e exatidão para o PCV, especialmente em comparação com métodos heurísticos ou a força bruta.