Problema do Cavalo no Xadrez

Trabalho de Algoritimos e Estrutura de Dados 2

1st Caetano Chinarelli Souza *Registro Acadêmico 2344955*
Universidade Tecnológica Federal do Paraná
Pato Branco, Paraná
2nd Guilherme Iago Marcante Della Libera *Registro Acadêmico 2199572*
Universidade Tecnológica Federal do Paraná
Pato Branco, Paraná
3rd Kelvyn Augusto Waltrick Nonato *Registro Acadêmico 2345048*
Universidade Tecnológica Federal do Paraná
Pato Branco, Paraná
4th Luiz Eduardo Rufatto *Registro Acadêmico 2079933*
Universidade Tecnológica Federal do Paraná
Pato Branco, Paraná
5th Vinicius Soares do Rosario *Registro Acadêmico 2247305*
Universidade Tecnológica Federal do Paraná
Pato Branco, Paraná

I. Introdução

Com o tempo, a programação foi se desenvolvendo cada vez mais rápido ao redor do mundo, e com isso foi se criando das mais diversas técnicas para a resolução de muitos problemas existente, e com isso conseguindo cada vez mais a otimização de códigos para seu melhor funcionamento e execução.

No problema que foi escolhido para resolvermos, o deslocamento de um cavalo em um tabuleiro de xadrez, em que é necessário achar a melhor combinação de passos possíveis para chegar até a posição final definida pelo jogador. Com isso, se encaixa a otimização de resultados, fazendo com que o processamento seja cada vez menor, buscando o melhor caso entre todos os possíveis.

Ao fazer a escolha do melhor modelo para resolver o problema selecionado, foi analisado qual seria a ideia para melhor aproveitamento de memória e processamento, onde o backtraking acabou se destacando por ser um modelo que descarta soluções mediana ou péssimas sem mesmo terminar sua checagem, e escolhendo sempre o melhor caso para a resolução do problema.

Após o problema resolvido e a concretização da melhor ação a ser tomada, a possibilidade da resolução de outros tipos de problema desse meio utilizando essa mesma técnica poderá ser feita com muito mais rapidez e economizando memória e processamento da máquina que está sendo utilizada.

II. Problemática

A. Descrição do problema

O problema tratado foi o do deslocamento do cavalo no tabuleiro de xadrez. Esta proposta, segue a seguinte descrição (conforme consta nas especificações do trabalho): dado um tabuleiro de ordem N (NxN) e uma posição inicial (X0, Y0), o algoritmo deve encontrar uma solução ótima (caso exista) com a menor quantidade de movimentos para chegar a posição final. O padrão de movimento do cavalo em um jogo de xadrez normal deve ser seguido, isto é, “em L”. Além disso, deve ser impresso a quantidade de passos e uma matriz onde cada elemento indica o número de passos para chegar à posição em questão.

A escolha deste problema resumiu-se a um acordo comum entre todos os integrantes do grupo através de uma votação, levando em consideração o consenso entre os membros que este foi de fácil compreensão, o que permitiu um planejamento mais claro e eficiente quanto a como abordá-lo.

B. Motivação para a escolha do problema

Para o desafio que nos propusemos a desenvolver, após alguns testes realizados, decidimos pela utilização do uso de backtracking, pois com ele sempre pegando o melhor caso sempre conseguir uma eficiência máxima.

E muitas vezes quando atrelado a programação dinâmica, é possível conseguir resultados melhor que o esperado, pois ele sempre evita vários cálculos desnecessários em diversos códigos, assim economizando ainda mais memória sempre que possível.

C. Estrategias para a solução

Do mesmo modo que são tratados diversos problemas envolvendo algoritmos, é possível encontrar múltiplas soluções através de diferentes estratégias aplicadas. O critério de desempate, neste caso e em geral, reduz-se a escolha do algoritmo com melhor eficiência em comparação aos demais, ou seja, que possui o menor custo computacional. Ainda para a escolha, consideram-se a facilidade de manipular a estratégia para

a aplicação desejada e a legibilidade do código. A partir disso, foi decidido a escolha de duas estratégias de algoritmos comprovadamente eficientes que nos auxiliam com a resolução do cenário em questão: backtracking aliada a busca em largura.

Backtracking é um tipo de algoritmo que deriva da busca por força bruta, sendo um refinamento desta. A estratégia em questão é utilizada para encontrar todas ou algumas soluções para problemas computacionais, especialmente problemas de satisfação de restrições. Consiste em construir candidatos para as soluções, "abandonando" um candidato (origina o termo backtrack, traduzindo, "retroceder") ao determinar que não pode ser completado para uma solução válida. Isto implica que o algoritmo só é válido para problemas que admitem "soluções parciais" como candidatos para soluções, então possibilitando um teste de validez relativamente simples para que a solução possa ser completada.

Conceitualmente, backtracking é frequentemente representado por uma estrutura de árvore, seguindo o padrão de busca em profundidade. Os candidatos a solução são os nós da árvore e cada nó de solução parcial é nó pai dos nós candidatos a solução, portanto também são nós intermediários. Se um nó candidato parcial é determinado como inválido, toda a subárvore desse nó também é desconsiderada, então a busca retrocede para o nó anterior e repete o processo de verificação para os demais nós da estrutura. A árvore é percorrida recursivamente, garantindo que toda a árvore válida é percorrida, mas não toda a árvore potencial (que contém todos os possíveis candidatos mas nem todos válidos).

O termo backtrack é creditado a Derrick H. Lehmer na década de 1950, porém diversos autores trabalharam acerca do tema ao longo dos anos.

Já a busca em largura é um algoritmo de busca que expande e examina todos os vértices de um grafo ou todos os nós de uma árvore, dependendo da representação escolhida. Na prática, o algoritmo geralmente utiliza uma estrutura de fila para marcar os nós visitados e os nós ainda não visitados. Dessa maneira, percorre todas as alternativas em um determinado nível até encontrar a solução ótima. Portanto, a busca em largura provou-se um algoritmo eficiente e apropriado para a resolução do problema do menor caminho para o cavalo no tabuleiro de tamanho N, dado que, por sua natureza, garante que o menor caminho seja escolhido.

A busca em largura e sua aplicação em encontrar componentes conectados de grafos foi inventado pelo engenheiro e cientista da computação alemão Konrad Zuse em 1945, porém seu estudo foi publicado apenas em 1972. Foi reinventado em 1959 pelo também cientista da computação e professor de matemática americano Edward F. Moore, que a utilizou para encontrar o menor caminho em um labirinto.

Para a resolução do problema, utilizamos a busca em largura para encontrar o menor caminho que o cavalo deve percorrer até chegar ao destino e ainda uma estrutura auxiliar (matriz "pai") para registrar e posteriormente reconstruir o caminho percorrido. A reconstrução é feita através de backtracking, utilizando a estrutura citada que mantém o pai de cada célula de posição, por meio de uma função recursiva que preenche uma outra matriz com os passos do cavalo. Ela percorre o caminho do destino de volta à origem, utilizando uma abordagem recursiva para reconstruir o percurso. Essa técnica permite explorar todas as possíveis opções para construir o caminho correto, mas neste caso específico, a função apenas segue as informações previamente calculadas pela busca em largura, o que simplifica o processo de backtracking.

III. Descrição das Soluções do Problema

Tendo que o problema que tivemos em mãos foi o cavalo que se move em padrões pré-definidos por um tabuleiro de xadrez, do local de saída escolhido pelo usuario, até um local de chegada escolhido pelo usuario, sendo o tabuleiro alocado dentro de uma matriz NxN, procuramos por um meio de encontrar o menor caminho que leva ao local de chegada, um meio de marcá-lo, e de mostrá-lo em uma matriz.

A solução encontrada para a resolução do problema foi uma combinação entre BFS e backtracking. A busca em profundidade do BFS encontra todos os caminhos possíveis para a área escolhida, fazendo diversos caminhos e os marcando para calcular cada caminho possível, salvando sempre a distância de cada local encontrado até a origem em seus próprios locais de armazenamento, isto é, se o caminho for encontrado, caso contrário, e retornado o valor –1. A realização do backtracking e enfim usada nos resultados encontrados pela aplicação anterior, depois de conferido o menor caminho, para então seguir da célula final a célula inicial, fazendo o caminho reverso ao qual foi dado do menor caminho, marcando assim na matriz o caminho que deve ser impresso no tabuleiro, e finalizando o processo temos uma função para a impressão da matriz no formato informado pelo usuario para ser o tabuleiro de xadrez.

IV. Análise de Complexidade Tempo e de Espaço

Texto aqui

```c
#include <stdio.h>
int main() {
    printf("Ola, Mundo!\n");
    return 0;
}
```

TABLE I
TABLE TYPE STYLES

Table Head	Table Column Head		
	Table column subhead	*Subhead*	*Subhead*
copy	More table copy[a]		

[a]Sample of a Table footnote.

Fig. 1. Example of a figure caption.

V. Conclusão

Texto aqui

VI. Declaração de Autoria

O integrante Caetano C. Souza foi responsável pelo desenvolvimento do código, o que inclui implementação e correção das funções de busca em largura para o menor caminho do cavalo, backtracking para reconstrução e representação do caminho, impressão da matriz e correções na main. O integrante ainda foi responsável pela documentação do código, a descrição do problema, motivações para a escolha das estratégias e sua descrição no artigo.

O integrante Vinícius S. Rosário foi responsável pela criação da base do código, bem como pela implementação dos algoritmos de backtracking, BFS e o arquivo de entrada de dados, contribuiu para a construção do relatório, fornecendo informações relevantes e detalhes sobre a implementação do código, organizou o repositório Git, criou o README e garanti que o código-fonte estivesse adequadamente versionado e assumiu a responsabilidade de organizar o grupo, distribuindo tarefas entre os membros e garantindo uma colaboração eficiente.

Referências

[1] Zuse, Konrad (1972), Der Plankalkül (in German), Konrad Zuse Internet Archive. See pp. 96–105 of the linked pdf file (internal numbering 2.47–2.56).

[2] Moore, Edward F. (1959). "The shortest path through a maze". Proceedings of the International Symposium on the Theory of Switching. Harvard University Press. pp. 285–292. As cited by Cormen, Leiserson, Rivest, and Stein.

[3] CORMEN, T. et al. Algoritmos: Teoria e Prática. Editora Campus, 2012.

[4] "Breadth First Search or BFS for a Graph - GeeksforGeeks". GeeksforGeeks. Consult. 2024-06-23. [Em linha]. Disponível: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/

[5] "Shortest path in an unweighted graph - GeeksforGeeks". GeeksforGeeks. Consult. 2024-06-23. [Em linha]. Disponível: https://www.geeksforgeeks.org/shortest-path-unweighted-graph/

[6] "Minimum Steps to Reach the Target - GeeksforGeeks". GeeksforGeeks. Consult. 2024-06-23. [Em linha]. Disponível: https://www.geeksforgeeks.org/minimum-steps-to-reach-the-target/

[7] J. R. Bitner e E. M. Reingold, "Backtrack programming techniques", Commun. ACM, vol. 18, n.º 11, pp. 651–656, novembro de 1975. Consult. 2024-06-23. [Em linha]. Disponível: https://doi.org/10.1145/361219.361224

[8] T. H. Cormen, Algoritmos. Elsevier, 2012.

[9] F. R. (Editor), P. v. B. (Editor) e T. W. (Editor), Eds., Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Sci., 2006.

[10] F. L. Bauer e H. Wössner, "The "Plankalkül" of Konrad Zuse", Commun. ACM, vol. 15, n.º 7, pp. 678–685, julho de 1972. Consult. 2024-06-23. [Em linha]. Disponível: https://doi.org/10.1145/361454.361515

texto aqui