Explorando Soluções Eficientes com Backtracking

Brenda de Oliveira dos Santos

Bacharelado em Ciência da Computação – Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA) – Camaçari, Bahia, Brasil

Abstract. *Backtracking is an algorithmic search technique that consists of looking for candidates for a valid solution through trial and error, in other words, if the candidate found is not valid, the algorithm will return from the previous point to that candidate and search again until it finds the valid solution for that problem. It is commonly used in the exploration of multiple possibilities or in solutions where it is necessary to reach all possible solutions for that application. In this article, we will explore the fundamentals of backtracking, its characteristics and how it can be applied to find efficient solutions to a variety of computational problems by analyzing practical examples that demonstrate the application of backtracking in real-world situations.*

Keywords. *Backtracking; Solution; Algorithm; Possibilities; Application.*

Resumo. Backtracking é uma técnica algorítmica de busca que consiste na procura por candidatos à uma solução válida através da tentativa e erro, ou seja, caso o candidato encontrado não seja válido, o algoritmo irá retornar do ponto anterior a esse candidato e buscar novamente até achar a solução válida para aquele problema. Ela é comumente utilizada na exploração de múltiplas possibilidades ou em soluções que seja necessário alcançar todas as soluções possíveis para aquela aplicação. Neste artigo, será explorado os fundamentos do backtracking, suas características e como ele pode ser aplicado para encontrar soluções eficientes para uma variedade de problemas computacionais a partir de análises de exemplos práticos que demonstram a aplicação do backtracking em situações do mundo real.

Palavras-Chave. Backtracking; Solução; Algoritmo; Possibilidades; Aplicação.

1. Introdução

A resolução de problemas complexos, muitas vezes, envolve a exploração de múltiplas soluções em busca da melhor solução possível e, no contexto dos problemas computacionais, isso não é diferente. As técnicas de busca algorítmica desempenham um papel fundamental em aplicações que requerem esse tipo de exploração, e o backtracking surge como uma abordagem que pode

ser simples e ao mesmo tempo complexa e sistemática para explorar soluções eficientes em uma variedade de domínios computacionais.

O fundamento do backtracking se caracteriza pela habilidade de construir soluções incrementais, testar sua viabilidade e retroceder quando necessário, o que permite a exploração eficiente das possibilidades e soluções. Entretanto, compreender quando e como aplicar o backtracking é essencial, uma vez que sua eficácia pode ser afetada pela natureza do problema levando ou à melhor solução ou a uma busca exaustiva que pode tornar o problema mais complexo.

O backtracking segue uma abordagem que facilita a compreensão e implementação nas aplicações e seu retrocesso é realizado tipicamente através de uma recursão. Apesar de sua adaptabilidade para diversas aplicações, em outros casos não tão viáveis, o algoritmo pode demandar de uma grande quantidade de memória, pois a quantidade de variável por cada chamada recursiva é diretamente proporcional ao tamanho do problema. Isso implica que a complexidade de tempo desse algoritmo pode crescer de modo exponencial. Embora seja eficaz para muitos problemas, o backtracking não garante eficiência em todos os casos, o que exige cuidadosa análise do problema específico e, em alguns casos, a busca por abordagens alternativas que melhor se aplicam para a necessidade da solução.

2. Soluções com Backtracking

Para desenvolver uma solução utilizando o backtracking, é importante analisar e entender o problema como o tamanho da entrada dos dados, o intervalo de valores e as características de sua saída. Ao entender todos esses parâmetros, é possível alcançar a otimização da solução e evitar processamento computacional desnecessário.

A escolha da estrutura de dados apropriada para o problema também tem grande impacto na performance do algoritmo considerando suas características e limitações, como exemplo prático, a utilização da fila de prioridade para armazenar os próximos movimentos possíveis que reduz a complexidade de tempo de sua solução. Além disso, a técnica de dividir para conquistar é extremamente útil para o desenvolvimento de soluções mais práticas para problemas que são mais complexos de serem solucionados como um todo.

Na vida real é possível exemplificar a utilização eficiente do backtracking em diferentes áreas de pesquisa e desenvolvimento. No processamento de imagens, o backtracking desempenha um papel significativo na resolução de problemas relacionados à reconhecimento de padrões e análise de

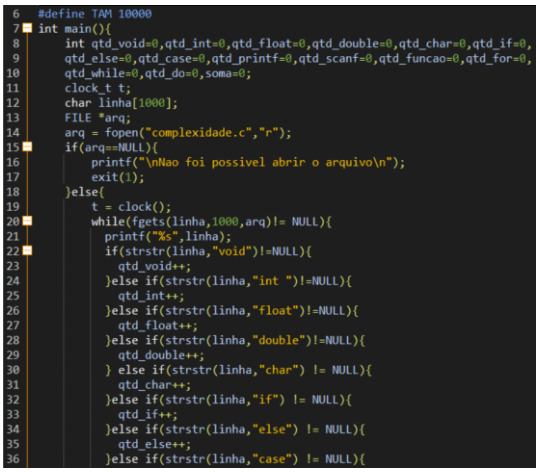
características visuais complexas como, por exemplo, na segmentação de imagens para reconhecimento de objetos, onde o backtracking pode ser utilizado para delimitar regiões de interesse, ajustando parâmetros de segmentação de maneira iterativa. Ainda, na reconstrução de imagens tridimensionais a partir de múltiplas perspectivas, o backtracking pode ser dedicado para otimizar a correspondência de características entre diferentes vistas, contribuindo para a precisão na reconstrução espacial.

Analisando em outro contexto, no desenvolvimento de jogos, o backtracking emerge como uma técnica valiosa para resolver desde a busca por trajetórias em labirintos até a otimização de estratégias de inteligência artificial (IA). Ao criar ambientes interativos e dinâmicos, o backtracking implementa sistemas de navegação eficientes para personagens, garantindo que estes possam encontrar caminhos otimizados através de cenários complexos, respondendo a mudanças nas condições do jogo. Além disso, em jogos de estratégia, o backtracking é empregado na tomada de decisões de IA, onde a análise retroativa de possíveis jogadas pode resultar em estratégias mais inteligentes e desafiadoras para os jogadores.

Por fim, analisando o último exemplo, no âmbito da otimização de redes de comunicação, ao lidar com a sua complexidade, o backtracking dispõe de suas características para encontrar configurações ideais, determinando caminhos eficientes para a transmissão de dados e otimizando a alocação de recursos, como largura de banda e capacidade de roteamento. Este método permite a exploração de diferentes topologias de rede, levando em consideração restrições específicas, como requisitos de largura de banda, latência e confiabilidade.

3. Analisador

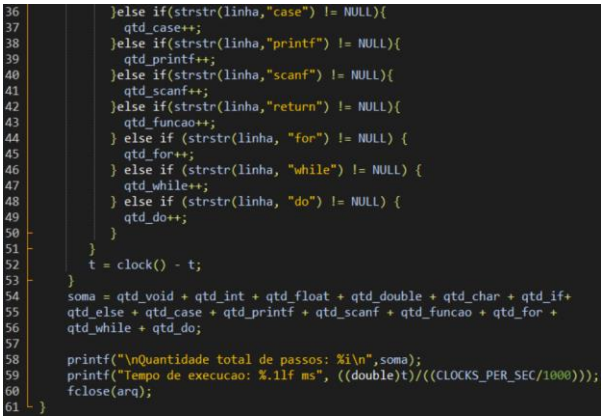
O analisador desenvolvido tem como objetivo analisar a quantidade de passos e calcular o tempo de execução do algoritmo selecionado no programa. Para sua implementação, foi utilizado a leitura de arquivo para a análise e a biblioteca *time.h* para o cálculo do tempo de execução.



```c
#define TAM 10000
int main(){
    int qtd_void=0,qtd_int=0,qtd_float=0,qtd_double=0,qtd_char=0,qtd_if=0,
    qtd_else=0,qtd_case=0,qtd_printf=0,qtd_scanf=0,qtd_funcao=0,qtd_for=0,
    qtd_while=0,qtd_do=0,soma=0;
    clock_t t;
    char linha[1000];
    FILE *arq;
    arq = fopen("complexidade.c","r");
    if(arq==NULL){
        printf("\nNao foi possivel abrir o arquivo\n");
        exit(1);
    }else{
        t = clock();
        while(fgets(linha,1000,arq)!= NULL){
          printf("%s",linha);
          if(strstr(linha,"void")!=NULL){
            qtd_void++;
          }else if(strstr(linha,"int ")!=NULL){
            qtd_int++;
          }else if(strstr(linha,"float")!=NULL){
            qtd_float++;
          }else if(strstr(linha,"double")!=NULL){
            qtd_double++;
          } else if(strstr(linha,"char") != NULL){
            qtd_char++;
          }else if(strstr(linha,"if") != NULL){
            qtd_if++;
          }else if(strstr(linha,"else") != NULL){
            qtd_else++;
          }else if(strstr(linha,"case") != NULL){
```

Figura 1. Analisador

Na figura 1, o algoritmo inicia com a declaração das variáveis para em seguida abrir o arquivo que é o programa em C que será analisado. Após a abertura do arquivo, há uma verificação para saber se o arquivo foi, de fato, aberto, caso não puder ser aberto, uma mensagem é exibida para o usuário. A variável “t” armazena o tempo inicial antes da análise da quantidade de passos do programa.



```c
          }else if(strstr(linha,"case") != NULL){
            qtd_case++;
          }else if(strstr(linha,"printf") != NULL){
            qtd_printf++;
          }else if(strstr(linha,"scanf") != NULL){
            qtd_scanf++;
          }else if(strstr(linha,"return") != NULL){
            qtd_funcao++;
          } else if (strstr(linha, "for") != NULL) {
            qtd_for++;
          } else if (strstr(linha, "while") != NULL) {
            qtd_while++;
          } else if (strstr(linha, "do") != NULL) {
            qtd_do++;
          }
        }
        t = clock() - t;
    }
    soma = qtd_void + qtd_int + qtd_float + qtd_double + qtd_char + qtd_if+
    qtd_else + qtd_case + qtd_printf + qtd_scanf + qtd_funcao + qtd_for +
    qtd_while + qtd_do;

    printf("\nQuantidade total de passos: %i\n",soma);
    printf("Tempo de execucao: %.11f ms", ((double)t)/((CLOCKS_PER_SEC/1000)));
    fclose(arq);
}
```

Figura 2. Segunda parte do Analisador

Na figura 2, vemos a continuação da contagem dos passos do algoritmo e, logo após, o cálculo do tempo final de execução. Em seguida, o analisador soma a quantidade desses passos contados e exibe para o usuário a quantidade total e o tempo de execução do programa. Por fim, o arquivo analisado é fechado.

4. Conclusão

A exploração de soluções eficientes com backtracking revela-se como uma abordagem robusta e versátil para a resolução de uma ampla variedade de problemas computacionais. A capacidade do backtracking de construir soluções

incrementais, testar sua viabilidade e retroceder quando necessário oferece uma estrutura bastante eficaz para abordar problemas complexos e, embora suas vantagens sejam evidentes na busca por soluções ótimas e na resolução eficiente de muitos cenários, é importante reconhecer suas limitações, especialmente em casos de complexidade exponencial. Portanto, ao explorar soluções com backtracking, é crucial equilibrar a compreensão profunda do problema em questão com a aplicação de otimizações específicas para alcançar soluções práticas e elegantes.

5. Referências

O que é um algoritmo Backtracking?. Stack Overflow, 2015. Disponível em: https://pt.stackoverflow.com/questions/103184/o-que-%C3%A9-um-algoritmo-backtracking. Acesso em fevereiro, 2024.

Backtracking: Exploring Backtracking Techniques through IOI Challenges. Faster Capital, 2023. Disponível em: https://fastercapital.com/content/Backtracking--Exploring-Backtracking-Techniques-through-IOI-Challenges.html. Acesso em fevereiro, 2024.

Força Bruta / Backtracking. Apresentação em PDF. Disponível em: https://edisciplinas.usp.br/pluginfile.php/7527342/course/section/6450765/ForcaBruta_Backtracking.pdf. Acesso em fevereiro, 2024.