Centro Universitário Senac
Bacharelado em Ciência da Computação
Análise e projeto de algoritmos

Professor: Leonardo Takuno
{leonardo.takuno@gmail.com}

15 de maio de 2020

1. Escreva um algoritmo, utilizando a técnica de programação backtracking, que apresenta todos os números binários de tamanho n, se for informado ao seu algoritmo N=3, teríamos: $000, 001, 010, 011, 100, 101, 110, 111$.

2. Suponha o seguinte problema, gerar todas as sequências possíveis com n números usando backtracking. Por exemplo, considere que temos $n = 3$ e os números são 0, 1 e 2. A solução seria:

   $$000, 001, 002, 010, 011, 012, 020, 021, 022, 100, 101, 102, \cdots 220, 221, 222.$$

   Ou seja, $3^3 = 27$ sequências. Escreva um algoritmo, usando backtracking que gera todas sequências possíveis para N números no vetor A.

3. Considere uma sequência $1 \cdots n$ números, o problema agora é gerar todas as combinações de m elementos desta sequência. A quantidade de combinações possíveis é $\frac{n!}{m! * (n-m)!}$.

   Por exemplo, para $n = 5$ e $m = 3$, as combinações de $1 \cdots 5$ com 3 elementos seria:

   1 2 3
   1 2 4
   1 2 5
   1 3 4
   1 3 5
   1 4 5
   2 3 4
   2 3 5
   2 4 5
   3 4 5

   Escreva um algoritmo, usando backtracking que gera todas combinações possíveis a partir de n números e com m elementos.

4. Considere o conjunto $a = \{a_1, a_2, ..., a_n\}$, o que queremos enumerar, ou listar todos os subconjuntos de a. Por exemplo para $n = 3$ teríamos:

   1
   2
   3
   1 2
   1 3
   2 3
   1 2 3

   Escreva um algoritmo, usando backtracking que gera todos subconjuntos possíveis a partir de n números.

5. (Percurso do cavalo) Dado um tabuleiro com $n \times n$ posições, o cavalo movimenta-se segundo as regras do xadrez. A partir de uma posição inicial (x0, y0), o problema consiste em fazer o cavalo "visitar" todas as casas do tabuleiro, sem repetições.

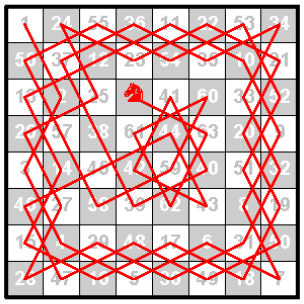

Figura 1: Percurso do cavalo no tabuleiro de xadrez

Escreva um algoritmo que encontre o percurso do cavalo no tabuleiro de xadrez.

6. (Permutação) Dado um conjunto de $n \geq 1$ elementos, imprimir todas as permutações possíveis deste conjunto. Por exemplo, se o conjunto é $\{1, 2, 3\}$, então o conjunto de todas as permutações é

$$\{(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 2, 1), (3, 1, 2)\}.$$

Não é difícil ver que dados n elementos, existem $n!$ permutações diferentes.

Um algoritmo pode ser obtido observando um caso particular, digamos, o caso de um conjunto com quatro elementos $\{1, 2, 3, 4\}$. A resposta pode ser construída da seguinte forma:

(a) 1 seguido de todas as permutações de (2,3,4)
(b) 2 seguido de todas as permutações de (1,3,4)
(c) 3 seguido de todas as permutações de (1,2,4)
(d) 4 seguido de todas as permutações de (1,2,3)

A expressão "seguido de todas as permutações" é a essência da recursão. Ela implica que podemos solucionar o problema para um conjunto com n elementos se temos uma algoritmo que soluciona o mesmo problema para $n - 1$ elementos. Escreva um algoritmo recursivo e um algoritmo iterativo para resolver o problema da permutação.

7. Considere uma partida de futebol entre duas equipes $A \times B$, cujo o placar final é $M \times N$, em que M e N são números de gols marcados por A e B, respectivamente. Escreva um algoritmo utilizando backtracking que imprima todas as possíveis sucessões de gols marcadas. Por exemplo, para um placar 3×1, temos: $AAAB$, $AABA$, $ABAA$, $BAAA$.

8. Escreva um algoritmo utilizando a estratégia backtracking que ordena um vetor de N elementos inteiros. (Um tanto ridículo, mas pode ser um bom exercício.)

9. (Saída do Labirinto) Um labirinto pode ser representado por meio de uma matriz booleana, onde cada posição com valor igual a 0 corresponde a uma passagem livre e uma posição com valor igual a 1 representa uma parede. Uma saída é uma posição livre na borda da matriz que define o labirinto. Escreva o pseudocódigo de um algoritmo backtracking que encontre um caminho que leve uma posição inicial qualquer a uma saída do labirinto, caso exista. Utilize o valor 2 para marcar um caminho válido da entrada até a saída.

10. (SUBSET-SUM) O problema subset-sum pode ser enunciado de forma resumida como: dado um conjunto X com números naturais e um inteiro positivo c, devemos encontrar um subconjunto cuja soma de seus elementos seja igual a c. Escreva o pseudo-código de um algoritmo backtracking que apresente um subconjunto, caso ele exista, que seja solução do problema.

11. (SUBSEQUÊNCIA CRESCENTE MÁXIMA) Suponha que $a[1 \cdot \cdot n]$ é uma sequência de números. Uma subsequência de $a[1 \cdot \cdot n]$ é o que sobra depois que um conjunto arbitrário de termos é apagado.

    Exemplo :

    - $s[] = \{1, 2, 3, 4, 5, 6\}$ é subsequência crescente máxima de $a[] = \{1, 2, 3, 9, 4, 5, 6\}$
    - $s[] = \{5, 6, 6, 7\}$ é uma subsequência crescente máxima de $a[] = \{9, 5, 6, 3, 9, 6, 4, 7\}$
    - A subsequência crescente $s[] = \{5, 6, 9\}$ de $a[] = \{9, 5, 6, 3, 9, 6, 4, 7\}$ é maximal (ou seja, não pode ser prolongada) mas não é máxima.

    Escreva um algoritmo, usando backtracking que encontra a subsequência crescente máxima de $a[1 \cdot \cdot n]$.