Tentativa e Erro (Backtracking)

Norton T. Roman

Apostila baseada no trabalho de Delano M. Beder, David Matuszek e Nivio Ziviani

Tentativa e Erro

- Suponha que você tem que tomar uma série de decisões dentre várias possibilidades, onde
  - Você não tem informação suficiente para saber o que escolher
  - Cada decisão leva a um novo conjunto de escolhas
  - Alguma seqüência de escolhas (possivelmente mais que uma) pode ser a solução para o problema
- Tentativa e erro é um modo metódico de tentar várias seqüências de decisões, até encontrar uma que funcione

Tentativa e Erro

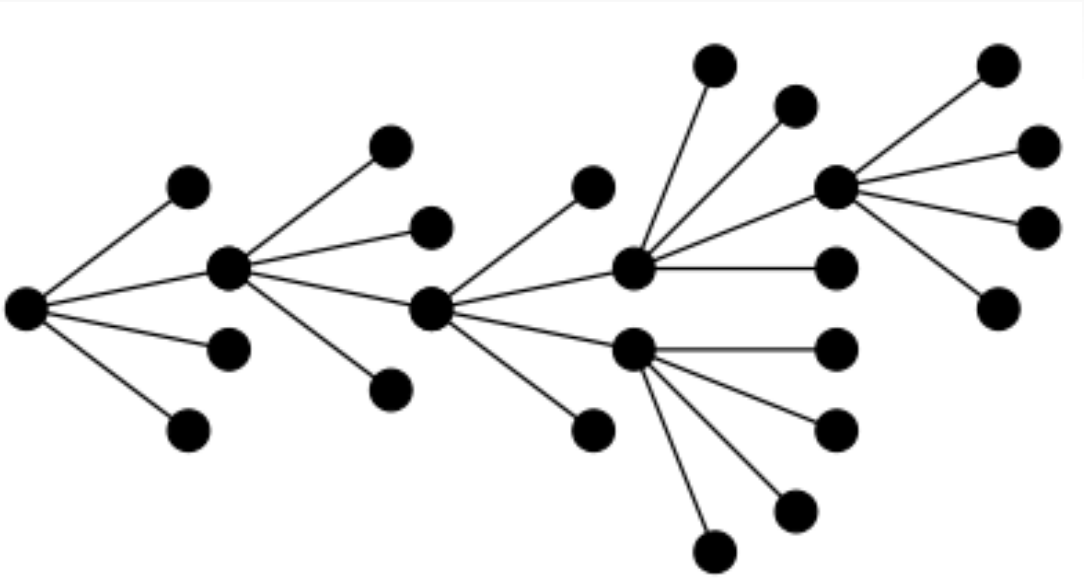
- Técnica de solução de problemas
  - Usada quando se quer achar soluções para problemas para os que não se conhece uma regra fixa de computação
- Passos
  - Escolher uma operação plausível;
  - Executar a operação com os dados;
  - Se a meta não foi alcançada, repita o processo até que se atinja a meta ou se evidencie a insolubilidade do problema.

Tentativa e Erro

- Tentativa e erro é uma técnica que utiliza recursividade
  - A recursividade pode ser usada para resolver problemas cuja solução é do tipo tentar todas as alternativas possíveis.
- Idéia para algoritmos tentativa e erro é decompor o processo em um número finito de sub-tarefas parciais (expressas de forma recursiva).
  - Explorá-las exaustivamente
  - A construção de uma solução é obtida através de tentativas (ou pesquisas) da árvore de sub-tarefas.

Tentativa e Erro

- O processo de tentativa gradualmente constrói e percorre uma árvore de sub-tarefas.



Tentativa e Erro

- Funcionamento:
  - Passos em direção à solução final são tentados e registrados em uma estrutura de dados;
  - Caso esses passos tomados não levem à solução final, eles podem ser retirados e apagados do registro.
- A busca na árvore de soluções pode crescer rapidamente (exponencialmente)
  - Necessário usar algoritmos aproximados ou heurísticas que não garantem a solução ótima mas são rápidas.

Tentativa e Erro

- Exploramos cada possibilidade como segue:
  - Se a possibilidade é a resposta, retorne “sucesso”
  - Se a possibilidade não for resposta, e não houver outra a ser testada a partir dela, retorne “falha”
  - Para cada possibilidade, a partir da atual:
    - Explore a nova possibilidade (recursivo)
    - Se encontrou a resposta, retorne “sucesso”
  - Retorne “falha”

Exemplos

- Dado um labirinto, encontre um caminho da entrada à saída
  - Em cada interseção, você tem que decidir se:

Exemplos

- Dado um labirinto, encontre um caminho da entrada à saída
  - Em cada interseção, você tem que decidir se:
    - Segue direto
    - Vai à esquerda
    - Vai à direita
  - Você não tem informação suficiente para escolher corretamente
  - Cada escolha leva a outro conjunto de escolhas
  - Uma ou mais seqüência de escolhas pode ser a solução

Exemplos

- Você deseja colorir um mapa com no máximo 4 cores: Vermelho, amarelo, verte e azul
  - Países adjacentes devem ter cores diferentes
  - Em cada iteração, você deve decidir ...

Exemplos

- Você deseja colorir um mapa com no máximo 4 cores: Vermelho, amarelo, verte e azul
  - Países adjacentes devem ter cores diferentes
  - Em cada iteração, você deve decidir que cor pinta um país, dadas as cores já atribuídas aos vizinhos
  - Você não tem informação suficiente para escolher as cores
  - Cada escolha leva a outro conjunto de escolhas
  - Uma ou mais seqüencia de passos pode ser a solução

Tentativa e Erro

- Recursão é a maneira mais natural de se implementar tentativa e erro:
- Considere o método recursivo:
  - int f(int a) {

```
...
b = f(c);
...
```

}

Tentativa e Erro

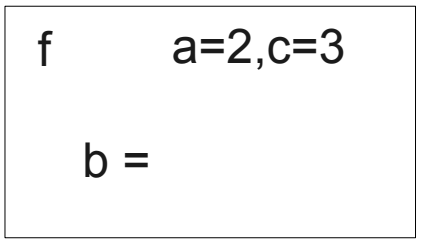
- Considere o método recursivo:
  - int f(int a) {

    ...

    b = f(c);

    ...

    }
  - Da primeira vez que f é chamada (com 2), cria-se espaço em memória para seus parâmetros, variáveis locais etc
    - Suponha que c torna-se 3



Tentativa e Erro

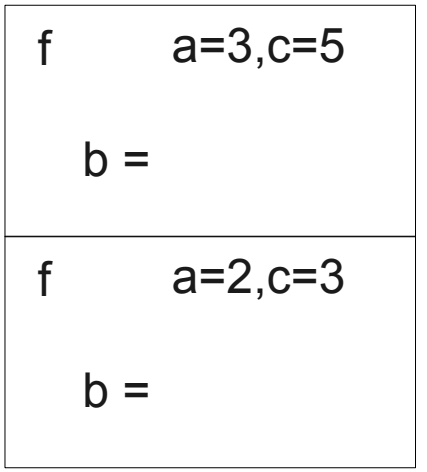
- Considere o método recursivo:
  - int f(int a) {

    ...

    b = f(c);

    ...

    }
  - Quando chega em f(c), a situação será como na figura
    - Suponha que c torna-se 5, em algum momento



Tentativa e Erro

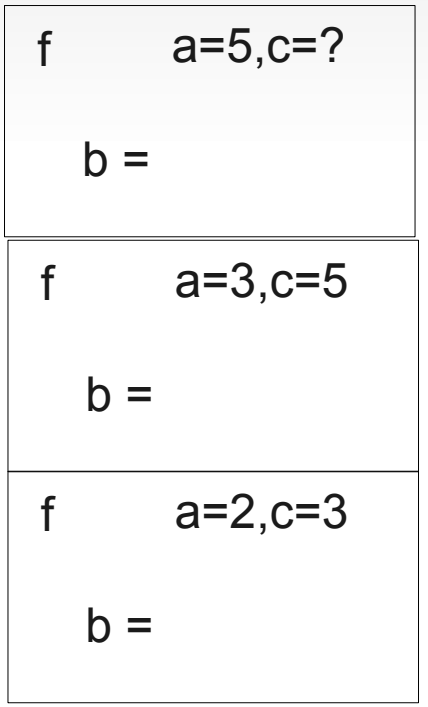
- Considere o método recursivo:
  - int f(int a) {

    ...

    b = f(c);

    ...

    }
  - Quando chega em f(c) novamente, a situação será como na figura
    - Suponha que, agora, é irrelevante o valor de c



Tentativa e Erro

- Considere o método recursivo:
  - int f(int a) {

    ...

    b = f(c);

    ...

    }
  - Antes de dar valor a c, f(5) retorna o valor 4, por exemplo
    - E "volta" à porção de memória no passo anterior da recursão – backtracking

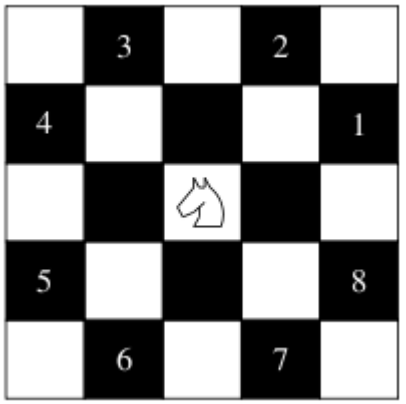
f	a=3,c=5
b = 4	
f	a=2,c=3
b =	

Tentativa e Erro

- Se conseguirmos codificar o problema de modo a que:
  - Cada nova decisão seja uma chamada recursiva
  - Cada backtracking corresponda a voltar de uma chamada recursiva
- Então a solução do problema, se feita recursivamente, naturalmente gerenciará o mecanismo de tentativa e erro

Exemplo – Passeio do Cavalo

- Passeio do cavalo no tabuleiro de xadrez.
  - Dado um tabuleiro com n × n posições, o cavalo se movimenta segundo as regras do xadrez.
  - A partir de uma posição inicial (x_0 , y_0), o problema consiste em encontrar, se existir um passeio do cavalo com n^2 - 1 movimentos, visitando todos os pontos do tabuleiro uma única vez



Passeio do Cavalo

- O tabuleiro ⇒ matriz n × n.
- Situação de cada posição:
  - t[x,y] = 0, <x,y > não foi visitada
  - t[x,y] = i, <x,y > visitada no i-ésimo movimento, $1 \leq i \leq n^2$.
- As regras do xadrez são utilizadas para os movimentos do cavalo

Passeio do Cavalo

```
procedimento tenta
    BEGIN
    inicializa seleção de movimentos
    WHILE movimento não bem sucedido AND existem candidatos a movimento DO
        seleciona próximo candidato ao movimento
        IF aceitável THEN
            registra movimento
            IF tabuleiro não está cheio THEN
                tenta novo movimento (chamada recursiva)
                IF não sucedido THEN
                    apaga registro anterior
                FI
            FI
        FI
    OD
END
```

Ex: Marca no tabuleiro

O movimento anterior não leva à solução. Deve voltar (backtracking)

Ao final, o tabuleiro conterá a resposta

Passeio do Cavalo

Para cálculo das coordenadas dos movimentos possíveis do cavalo

```
public class KnightsTour {
    final int[] dx = { 2, 1, -1, -2, -2, -1, 1, 2 };
    final int[] dy = { 1, 2, 2, 1, -1, -2, -2, -1 };
    final int num; //número de posições do tabuleiro
    final int numSqr; //número total de casas
    int[][] table;
    public KnightsTour(int num) {
       this.num = num;
       this.numSqr = num * num;
       this.table = new int[num][num];
    }
    boolean isAcceptable(int x, int y) {
       boolean result = (x >= 0 && x <= num - 1);
       result = result && (y >= 0 && y <= num - 1);
       result = result && (table[x][y] == 0);
       return result;
    }
...
}
```

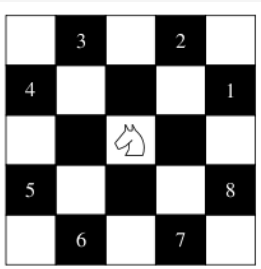
Aceitável se estiver dentro do tabuleiro e a casa ainda não tiver sido vizitada

Passeio do Cavalo

Tenta o i-ésimo movimento em (x,y), $1 \leq i \leq n^2$

```
boolean tryMove(int i, int x, int y) {
   //Verifica a quantidade de movimentos
   boolean done = (i > numSqr);
   int k = 0;
   int u, v;
   while (!done && k < 8) {
      u = x + dx[k];
      v = y + dy[k];
      if (isAcceptable(u, v)) {
         table[u][v] = i;
         done = tryMove(i + 1, u, v); //tenta outro movimento
         if (!done) {
            table[u][v] = 0; //sem sucesso. Descarta movimento
         }
      }
      k = k + 1; //passa ao próximo movimento possível
   }
   return done;
}
```

Coordenadas dos 8 movimentos possíveis em volta do cavalo



Passeio do Cavalo

Mostra todos os movimentos a partir de (x,y)

```
void showTour(int x, int y) {
    table[x][y] = 1;
    boolean done = tryMove(2, x, y);
    if (done) {
       for (int i = 0; i < num; i++) {
           for (int j = 0; j < num; j++) {
                System.out.print(table[i][j] + " ");
           }
           System.out.println();
       }
    } else {
           System.out.println("Não há passeio possível");
    }
}
 public static void main(String[] args) {
   int n = Integer.parseInt(args[0]);
   int x = Integer.parseInt(args[1]);
   int y = Integer.parseInt(args[2]);
   new KnightsTour(n).showTour(x, y);
 }
```

Passeio do Cavalo

- Resultado:

1	60	39	34	31	18	9	64
38	35	32	61	10	63	30	17
59	2	37	40	33	28	19	8
36	49	42	27	62	11	16	29
43	58	3	50	41	24	7	20
48	51	46	55	26	21	12	15
57	44	53	4	23	14	25	6
52	47	56	45	54	5	22	13