Fontes principais

1. Cormem T. H.; Leiserson C. E.; Rivest R.: Stein C.. *Introduction to Algorithms*, 3^a edição, MIT Press, 2009

2. Análise de algoritmo - IME/USP (prof. Paulo Feofiloff) `http://www.ime.usp.br/~pf/analise_de_algoritmos`

Backtracking

algoritmos de enumeração

Backtracking

É um refinamento da técnica de força bruta.
Considera uma série de tomadas de decisões entre várias opções.
Algumas consequências de decisões podem conduzir a uma solução do problema.

Backtracking

Ideia geral:

(1) Soluções representadas por n tuplas ou vetores de solução $(s_1, s_2, \cdots, s_n)$ de maneira que cada s_i é escolhido a partir de um conjunto finito de opções v_i;

(2) Inicia um vetor vazio;

Backtracking

(3) Em cada nova etapa do vetor é extendido com um novo valor;

(4) O novo valor deve ser avaliado: se não for solução parcial, então o último valor do vetor é eliminado e outro candidato é considerado.

Note que em (4) pode ou não ocorrer retrocesso (backtracking).

Backtracking

Restrições

- Restrições explícitas:
  Correspondem às regras que restringem cada s_i em tomar valores de um determinado conjunto (relacionado ao problema e escolhas possíveis).

- Restrições implícitas:
  Determinam como os s_i's se relacionam entre si.

Algoritmo de backtracking genérico

backtrack($s[1 \cdot\cdot k]$)

$\triangleright$ s é um vetor promissor de tamanho k
se s é a solução **então**
    escreva(s)
senão
    para cada vetor promissor w de tamanho $k+1$ **faça**
        backtracking($w[1 \cdot\cdot k+1]$)
    $\triangleright$ $w[1 \cdot\cdot k] = s[1 \cdot\cdot k]$

Sequências

Sequências

Suponha o seguinte problema:

Gerar todas as sequências possíveis de 3 dígitos com os dígitos 1,2 e 3 armazenados no vetor $v[0 \cdots n]$, onde $n = 3$

Solução: 111, 112, 113, 121, 122, 123, 131, 132, 133, 211, 212, 213, 221, 222, 223, 231, 232, 233, 311, 312, 313, 321, 322, 323, 331, 332, 333

A quantidade é de $3^3 = 27$ sequências.

Sequências

algoritmo mostra_sequencias(s, i, v, n)

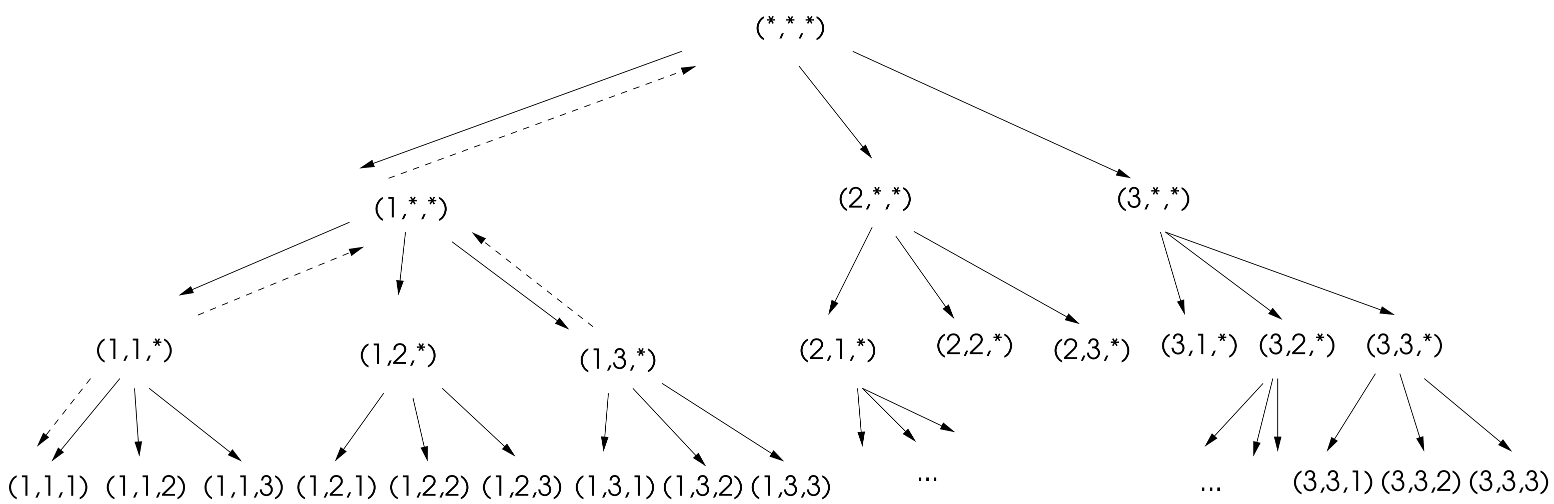
entrada: vetor solução: $s[0 \cdot\cdot n-1]$
inteiro $i = 0$ que controla a profundidade da recursão,
vetor de entrada: $v[0 \cdot\cdot n-1]$
inteiro n o tamanho do vetor a

saída: apresenta todas as sequências de n elemntos de a

Sequências

```
mostra_sequencias(s, i, v, n)
se i = n então
    escreva(s, n)
senão
    para j = 0 até n − 1 faça
        s[i] = v[j]
        mostra_sequencias(s, i + 1, v, n)
```

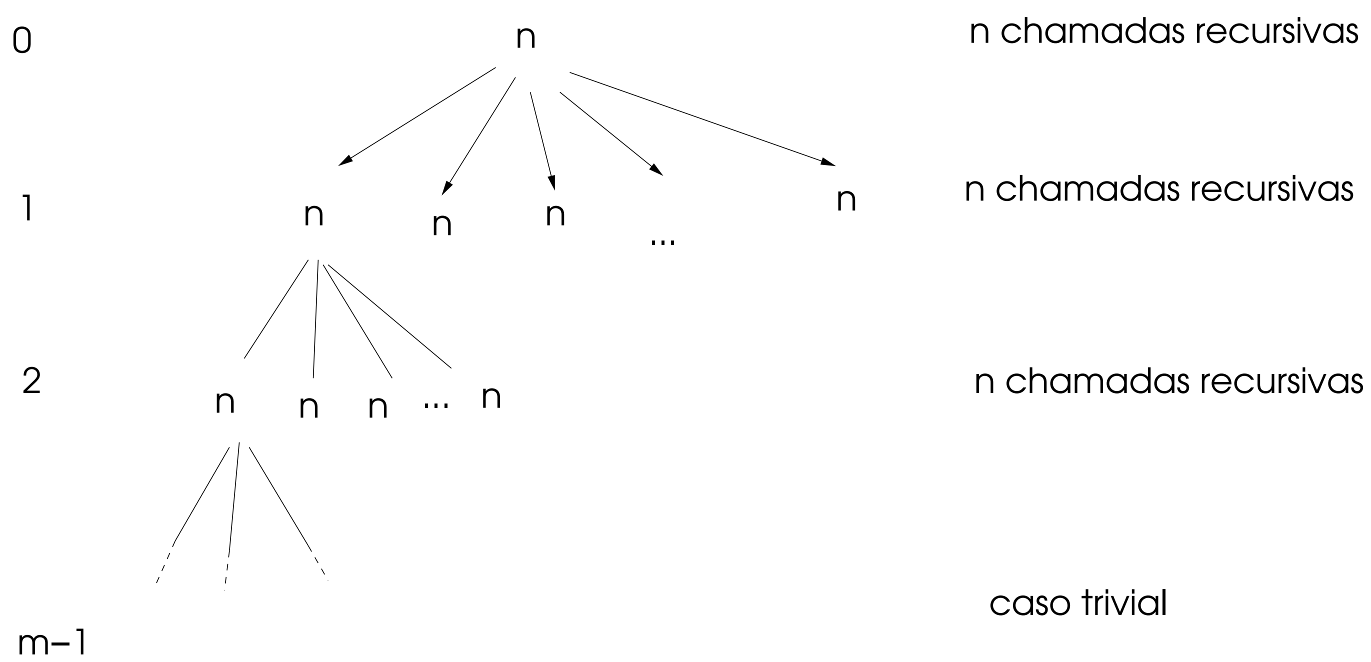
Sequências


(*,*,*)
(1,*,*)
(2,*,*)
(3,*,*)
(1,1,*)
(1,2,*)
(1,3,*)
(2,1,*)
(2,2,*)
(2,3,*)
(3,1,*)
(3,2,*)
(3,3,*)
(1,1,1)
(1,1,2)
(1,1,3)
(1,2,1)
(1,2,2)
(1,2,3)
(1,3,1)
(1,3,2)
(1,3,3)
...
...
(3,3,1)
(3,3,2)
(3,3,3)

Sequências

Suponha que agora queremos gerar todas as combinações de tamanho 3 utilizando os dígitos $a = \{1, 2, 3, 4\}$. Nesta caso, temos que definir $s[0 \cdot\cdot m-1]$, com $m = 3$, e o vetor $a[0 \cdot\cdot n-1]$, com $n = 4$.

Sequências



Complexidade do algoritmo: $O(n^m)$

Permutações

Permutações

Uma permutação da sequência $1 \cdot\cdot n$ é qualquer rearranjo dos termos dessa sequência. Em outras palavras, uma permutação da sequência $1 \cdot\cdot n$ troca os elementos de lugar gerando uma nova sequência com estes mesmos elementos.

Permutações - Exemplo

Dado uma vetor v, escreva um programa que imprima todas as permutações. Se v tem n elementos, então o programa deve imprimir $n!$ permutações.

entrada: $v = \{1, 2, 3\}$
saída: $\{1, 2, 3\}$, $\{1, 3, 2\}$, $\{2, 1, 3\}$, $\{2, 3, 1\}$, $\{3, 1, 2\}$, $\{3, 2, 1\}$.

Permutações

algoritmo permuta(s, i, v, n, x)

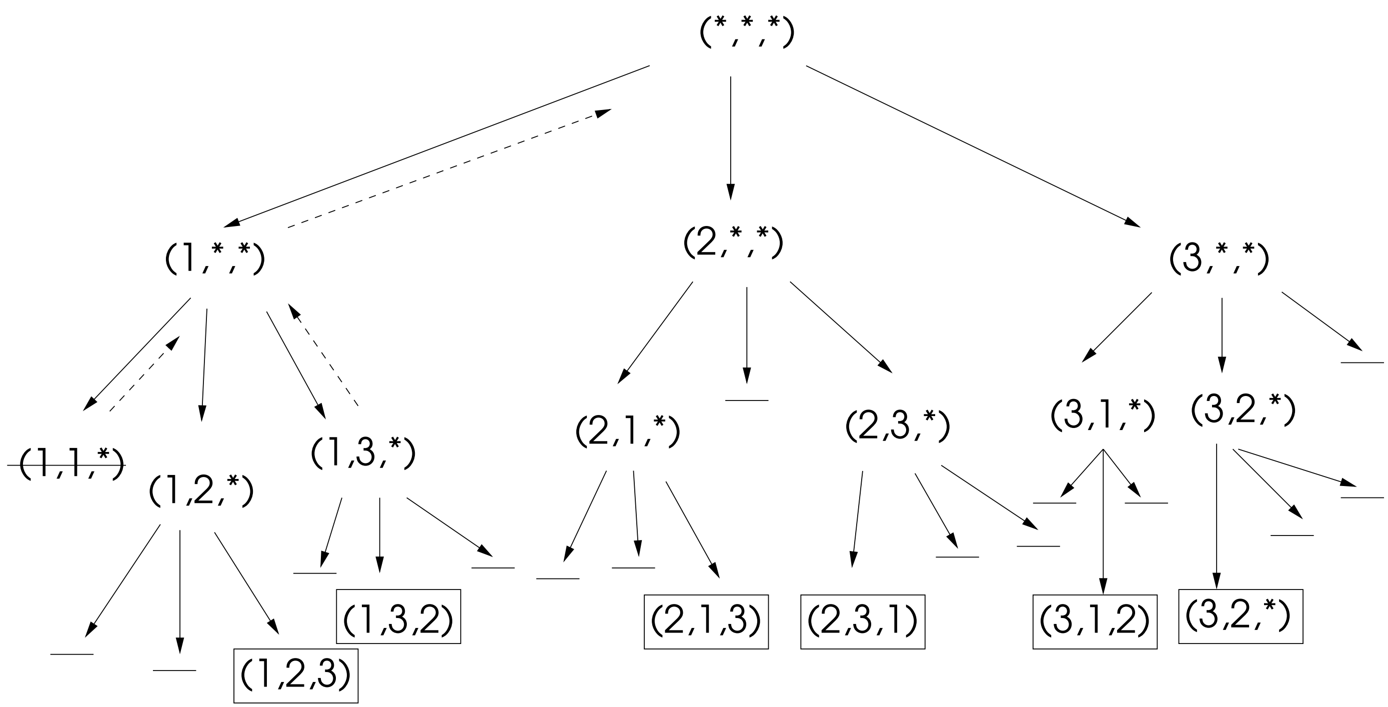
entrada: vetor solução: $s[0 \cdot\cdot n-1]$
inteiro $i = 0$ que controla a profundidade da recursão,
vetor de entrada: $v[0 \cdot\cdot n-1]$
inteiro n o tamanho do vetor a
vetor de marcação: $x[0 \cdot\cdot n-1]$ marca os elementos que já foram inseridos no conjunto solução.

saída: apresenta a permutação dos elementos do vetor a

Permutações

```
permuta(s, i, v, n, x)
    se i = n então
        escreva(s, n)
    senão
        para j = 0 até n − 1 faça
            ▷ verifica se a solução é promissora
            se x[j] == 0 então
                x[j] = 1 ▷ inclui na solução
                s[i] = v[j]
                permuta(s, i + 1, v, n, x)
                x[j] = 0
```

Permutações



Permutações

Restrições explícitas: O conjunto solução s tem uma permutação ao chegar ao caso base.

Restrições explícitas: Na permutação não há repetição de elementos.

Obrigado