PICAT: Uma Linguagem de Programação Multiparadigma

Miguel Alfredo Nunes, Jeferson L. R. Souza, Claudio Cesar de Sá

miguel.nunes@edu.udesc.br

jeferson.souza@udesc.br

claudio.sa@udesc.br

Departamento de Ciência da Computação
Centro de Ciências e Tecnológias
Universidade do Estado de Santa Catarina

13 de abril de 2019

Contribuições

- Alexandre Gonçalves;
- João Herique Faes Battisti;
- Paulo Victor de Aguiar;
- Rogério Eduardo da Silva;
- Hakan Kjellerstrand – (`http://www.hakank.org/picat/`)
- Neng-Fa Zhou – (`http://www.picat-lang.org/`)
- Outros anônimos que auxiliaram na produção deste documento;

Programação Dinâmica (PD) – I

- Uma poderosa técnica de programação que contorna a complexidade de certos problemas exponenciais

Programação Dinâmica (PD) – I

- Uma poderosa técnica de programação que contorna a complexidade de certos problemas exponenciais
- O problema **deve** apresentar uma *regra de recorrência*

Programação Dinâmica (PD) – I

- Uma poderosa técnica de programação que contorna a complexidade de certos problemas exponenciais
- O problema **deve** apresentar uma *regra de recorrência*
- A idéia é que todos os cálculos feitos a partir desta *regra de recorrência*, são consultados e armazenados numa *tabela dinâmica*

Programação Dinâmica (PD) – I

- Uma poderosa técnica de programação que contorna a complexidade de certos problemas exponenciais
- O problema **deve** apresentar uma *regra de recorrência*
- A idéia é que todos os cálculos feitos a partir desta *regra de recorrência*, são consultados e armazenados numa *tabela dinâmica*
- Esta técnica de utilizar uma *tabela dinâmica* nos cálculos intermediários, evitando a repetição do que já foi calculado anteriormente, é conhecida como: Programação Dinâmica, ou simplesmente: PD

Programação Dinâmica (PD) – II

- Como Picat usa a recursão, na programação em lógica, nada mais natural do que esta ter a PD disponível

Programação Dinâmica (PD) – II

- Como Picat usa a recursão, na programação em lógica, nada mais natural do que esta ter a PD disponível
- O comando que cria uma tabela para um determinado predicado é o *tabling*

Programação Dinâmica (PD) – II

- Como Picat usa a recursão, na programação em lógica, nada mais natural do que esta ter a PD disponível
- O comando que cria uma tabela para um determinado predicado é o *tabling*
- O *tabling* é um dos elementos fortes do planejador do Picat (módulo *planner*)

Programação Dinâmica (PD) – II

- Como Picat usa a recursão, na programação em lógica, nada mais natural do que esta ter a PD disponível
- O comando que cria uma tabela para um determinado predicado é o *tabling*
- O *tabling* é um dos elementos fortes do planejador do Picat (módulo *planner*)
- O exemplo escolhido para ilustrar a PD em Picat, veio do texto *Modeling and Solving AI Problems in Picat*, de Roman Barták e Neng-Fa

Exemplo de Uso da Programação Dinâmica – (PD)

- Seja o binômio $(x + y)^n$, conhecido *Binômio de Newton*

Exemplo de Uso da Programação Dinâmica – (PD)

- Seja o binômio $(x + y)^n$, conhecido *Binômio de Newton*
- Casos particulares são:
- $(x + y)^0 = 1$
- $(x + y)^1 = x + y$
- $(x + y)^2 = x^2 + 2xy + y^2$

Exemplo de Uso da Programação Dinâmica – (PD)

- Seja o binômio $(x+y)^n$, conhecido *Binômio de Newton*
- Casos particulares são:
- $(x+y)^0 = 1$
- $(x+y)^1 = x+y$
- $(x+y)^2 = x^2 + 2xy + y^2$
- $(x+y)^2 = x^2y^0 + 2x^1y^1 + x^0y^2$
- $(x+y)^3 = x^3y^0 + 3x^2y^1 + 3x^1y^2 + x^0y^3$
- $(x+y)^4 = x^4y^0 + 4x^3y^1 + 6x^2y^2 + 4x^1y^3 + x^0y^4$.
- .....

Exemplo de Uso da Programação Dinâmica – (PD)

- Seja o binômio $(x+y)^n$, conhecido *Binômio de Newton*
- Casos particulares são:
- $(x+y)^0 = 1$
- $(x+y)^1 = x+y$
- $(x+y)^2 = x^2 + 2xy + y^2$
- $(x+y)^2 = x^2y^0 + 2x^1y^1 + x^0y^2$
- $(x+y)^3 = x^3y^0 + 3x^2y^1 + 3x^1y^2 + x^0y^3$
- $(x+y)^4 = x^4y^0 + 4x^3y^1 + 6x^2y^2 + 4x^1y^3 + x^0y^4$.
- .....
- Como obter estes coeficientes polinômios?

Exemplo de Uso da Programação Dinâmica – (PD)

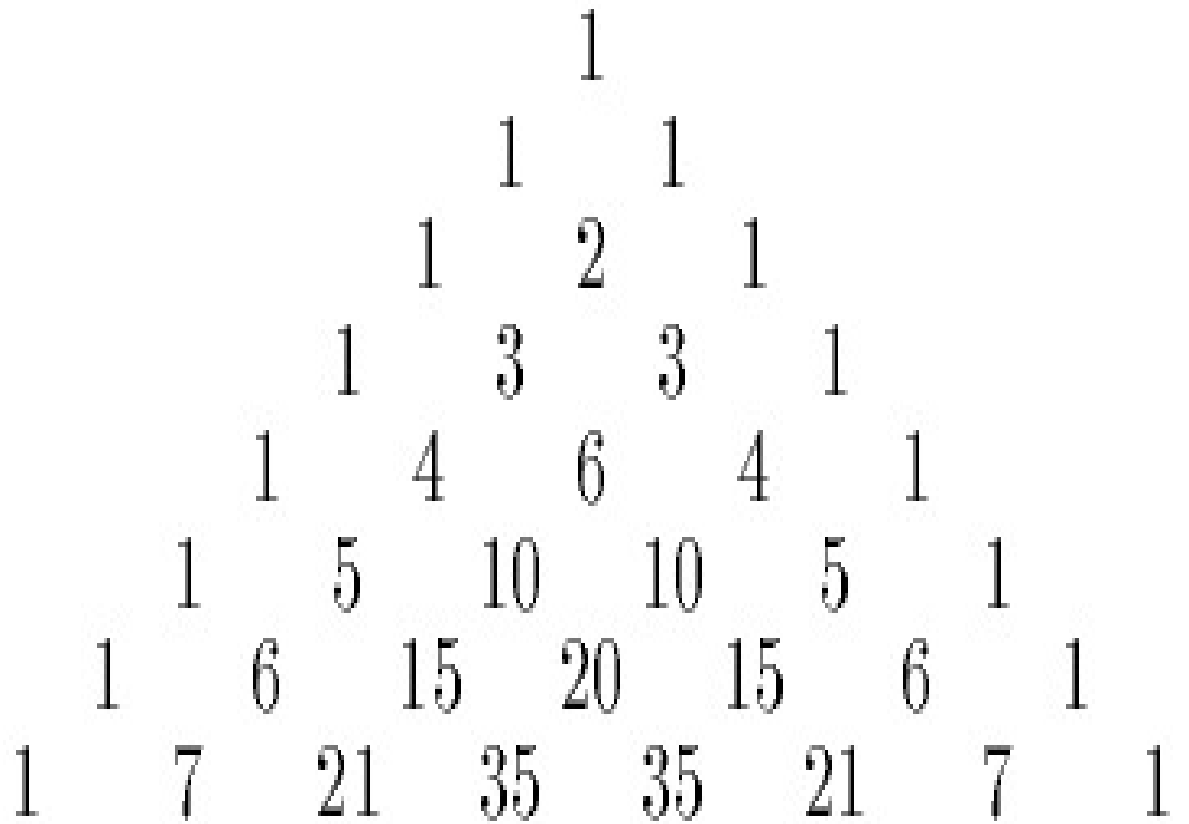

Figura 1: O triângulo de Pascal

Exemplo de Uso da Programação Dinâmica – (PD)

A Triangulo de Pascal

N	p 0	1	2	3	4	5	6	
0	$\binom{0}{0}$							1
1	$\binom{1}{0}$	$\binom{1}{1}$						1 1
2	$\binom{2}{0}$	$\binom{2}{1}$	$\binom{2}{2}$					1 2 1
3	$\binom{3}{0}$	$\binom{3}{1}$	$\binom{3}{2}$	$\binom{3}{3}$				1 3 3 1
4	$\binom{4}{0}$	$\binom{4}{1}$	$\binom{4}{2}$	$\binom{4}{3}$	$\binom{4}{4}$			1 4 6 4 1
5	$\binom{5}{0}$	$\binom{5}{1}$	$\binom{5}{2}$	$\binom{5}{3}$	$\binom{5}{4}$	$\binom{5}{5}$		1 5 10 10 5 1
6	$\binom{6}{0}$	$\binom{6}{1}$	$\binom{6}{2}$	$\binom{6}{3}$	$\binom{6}{4}$	$\binom{6}{5}$	$\binom{6}{6}$	1 6 15 20 15 6 1

Figura 2: O triângulo de Pascal – Coeficientes Binomiais

Formulação Matemática – I

- O *coeficiente binomial*, também chamado de *número binomial*, de um número n, na classe k, consiste no número de combinações de n termos, k a k.

Formulação Matemática – I

- O *coeficiente binomial*, também chamado de *número binomial*, de um número n, na classe k, consiste no número de combinações de n termos, k a k.
- O número binomial de um número n, na classe k, pode ser escrito como:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)(n-2)\cdots(n-k+1)}{k!}$$

Formulação Matemática – II

- Alternativa ao cálculo do fatorial, tem-se a relação de Stiffel:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Formulação Matemática – II

- Alternativa ao cálculo do fatorial, tem-se a relação de Stiffel:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

- O coeficiente binomial é muito utilizado no Triângulo de Pascal, onde o termo na linha n e coluna k é dado por: $\binom{n-1}{k-1}$

Formulação Matemática – II

- Alternativa ao cálculo do fatorial, tem-se a relação de Stiffel:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

- O coeficiente binomial é muito utilizado no Triângulo de Pascal, onde o termo na linha n e coluna k é dado por: $\binom{n-1}{k-1}$
- A fórmula de Stiffel é recorrente, e diretamente escrita em Picat.
  Veja os códigos.

Código em Partes

```
import datetime.    %%% para o statistics
import util.


table
c(_, 0) = 1.
c(N, N) = 1.
c(N,K) = c(N-1, K-1) + c(N-1, K).
```

Código em Partes

```
main  ?=>
    statistics(runtime,_), % faz uma marca do 1o. statistics
    N = 10, %% ateh uns 30 ... são números grandes ... fatorial

     foreach(I in 0  .. N)
        foreach(J in 0  ..  I)
             printf("  %d", c(I,J))
          end,
         printf(" \n"),
      end,

    statistics(runtime, [T_Picat_ON, T_final]),
    T = (T_final) / 1000.0, %%% está em milisegundos
    printf("\n CPU time %f em SEGUNDOS ", T),
    printf("\n OVERALL PICAT CPU time %f em SEGUNDOS ", T_Picat_

    printf(" \n ========================================\n ")
    %%% , fail descomente para multiplas solucoes
    .
```

Código Completo

- Acompanhar as explicações do código de: `https://github.com/claudiosa/CCS/blob/master/picat/coeficiente_binomial_PD.pi`
- Confira a execução

Saída

```
[ccs@gerzat picat]$ picat coeficiente_binomial_PD.pi
  1
  1   1
  1   2   1
  1   3   3   1
  1   4   6   4   1
  1   5   10   10   5   1
  1   6   15   20   15   6   1
  1   7   21   35   35   21   7   1
  1   8   28   56   70   56   28   8   1
  1   9   36   84   126   126   84   36   9   1
  1   10   45   120   210   252   210   120   45   10   1

 CPU time 0.000000 em SEGUNDOS
 OVERALL PICAT CPU time 0.009000 em SEGUNDOS
 ========================================
```

Reflexões

- Há outros métodos para se resolver estes problemas

Reflexões

- Há outros métodos para se resolver estes problemas
- O comando *tabling* é a base do módulo *planner*

Resumindo

- Picat é jovem (nascida em 2013);
- Uma evolução ao Prolog após seus mais de 40 anos de existência e sucesso!
- Sua sintaxe é moderna;
- Código aberto, multi-plataforma, e repleta de possibilidades;
- Uso para fins diversos;
- Muitas bibliotecas específicas prontas: CP, SAT, Planner, etc;
-
- .