Programação Dinâmica

Stefano Tommasini
Supervisor: Carlos Eduardo Ferreira
Instituo de matemática e estatística - USP

IME-USP

Objetivos

1. Criar um bom material didático sobre programação dinâmica destinado à alunos da graduação.
2. Mostrar no material alguns problemas mais difíceis e avançados em relação aos normalmente estudados na graduação.
3. Facilitar o aprendizado de programação dinâmica.

Introdução

- Programação Dinâmica é uma técnica muito importante para a resolução de problemas de otimização. Ela consiste em quebrar um problema grande em subproblemas menores que se sobrepõem, e obedecem a propriedade de subestrutura ótima. Um problema apresenta uma subestrutura ótima quando uma solução ótima para o problema contém em seu interior soluções ótimas para subproblemas. A superposição de subproblemas acontece quando um algoritmo recursivo reexamina o mesmo problema muitas vezes.
- Apesar de ser uma técnica básica, abordada em disciplinas iniciais de Algoritmos e Estruturas de Dados, muitas vezes os alunos têm dificuldades de resolver problemas em que a técnica é necessária. Nos livros didáticos e materiais disponíveis na internet poucos exemplos são abordados e muitas vezes o aluno não é capaz de aprender todas as técnicas e saber abordar um problema do tipo. Neste trabalho pretendo criar um material didático em português sobre programação dinâmica para auxiliar o aprendizado do tópico, analisando vários problemas de diferentes abordagens e dificuldades.

Exemplo: Jogo das Caixas

- Um grupo de $\mathbf{n}$ amigos está participando de um jogo. Nesse jogo existem $\mathbf{n}$ caixas numeradas de $\mathbf{0}$ a $\mathbf{n-1}$. Dentro de cada caixa encontra-se um nome de um dos amigos, de tal forma que cada amigo tem uma caixa que lhe corresponda. A probabilidade de um amigo encontrar seu nome numa caixa qualquer é de $\frac{1}{n}$, ou seja, obedece uma distribuição uniforme. No jogo cada integrante do grupo abrirá $\mathbf{K}$ caixas de sua escolha, se não encontrar seu nome em uma dessas $\mathbf{K}$ caixas todos os participantes perdem. Cada participante desconhece as escolhas feitas pelos outros participantes. Porém podem conversar antes de começar o jogo e definir alguma estratégia. Se todos os participantes escolherem $\mathbf{K}$ caixas aleatoriamente, a pro babilidade de vencerem é de $(\frac{K}{n})^n$. Não satisfeitos com essa probabilidade, os integrantes do grupo escolheram uma estratégia para abrir as caixas seguindo o seguinte algoritmo: Primeiramente enumeram todos integrantes do grupo de $\mathbf{0}$ a $\mathbf{n-1}$. Cada integrante começa abrindo a caixa correspondente ao seu número e enquanto não encontrar o seu próprio nome, abrirá a caixa cujo número corresponde ao nome que acabou de encontrar.
  Dados $\mathbf{n}$ e $\mathbf{K}$, com $\mathbf{K \leq n}$, determine a probabilidade dos amigos vencerem utilizando o algoritmo descrito.

Mais sobre o problema

- O problema trata-se de uma permutação aleatória que corresponde aos nomes escondidos dentro das caixas.
- A estratégia dos amigos corresponde a seguir o caminho no grafo induzido pela permutação aleatória.
- O jogador perde se e somente se ele não voltar para sua própria caixa após $\mathbf{K}$ movimentos.
- Então, o problema se resume a achar a probabilidade de que o maior ciclo de uma permutação aleatória tenha tamanho menor ou igual a $\mathbf{K}$.

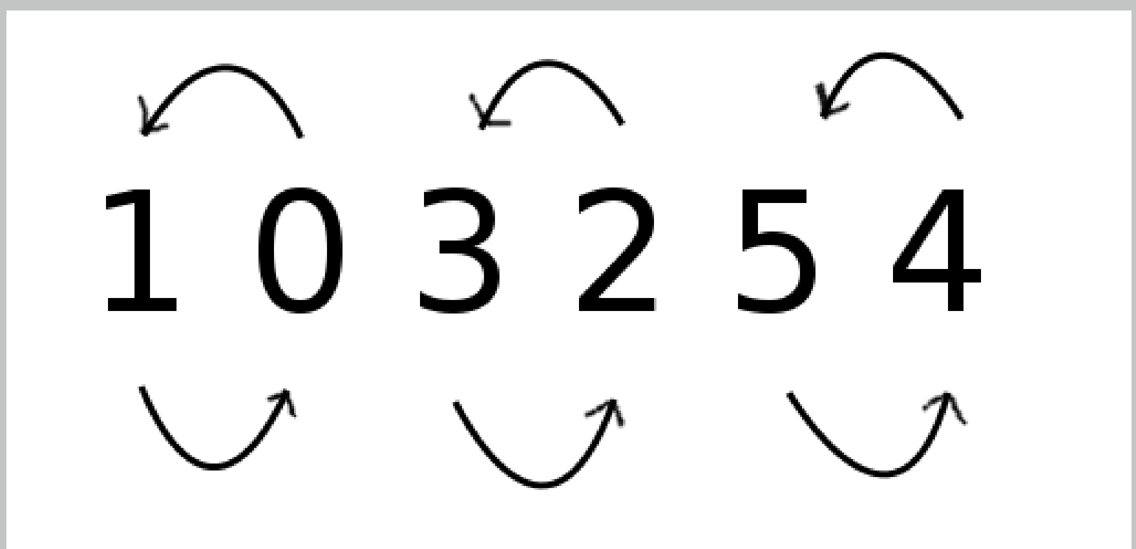


Figure 1: Grafo para permutação 1, 0, 3, 2, 5, 4 onde o maior ciclo tem tamanho 2

Amigo Secreto

- Joguinho disponível na Metamateca da Universidade de São Paulo.
- Mesma Ideia dos ciclos sobre permutações

Figure 2: Foto do joguinho disponível

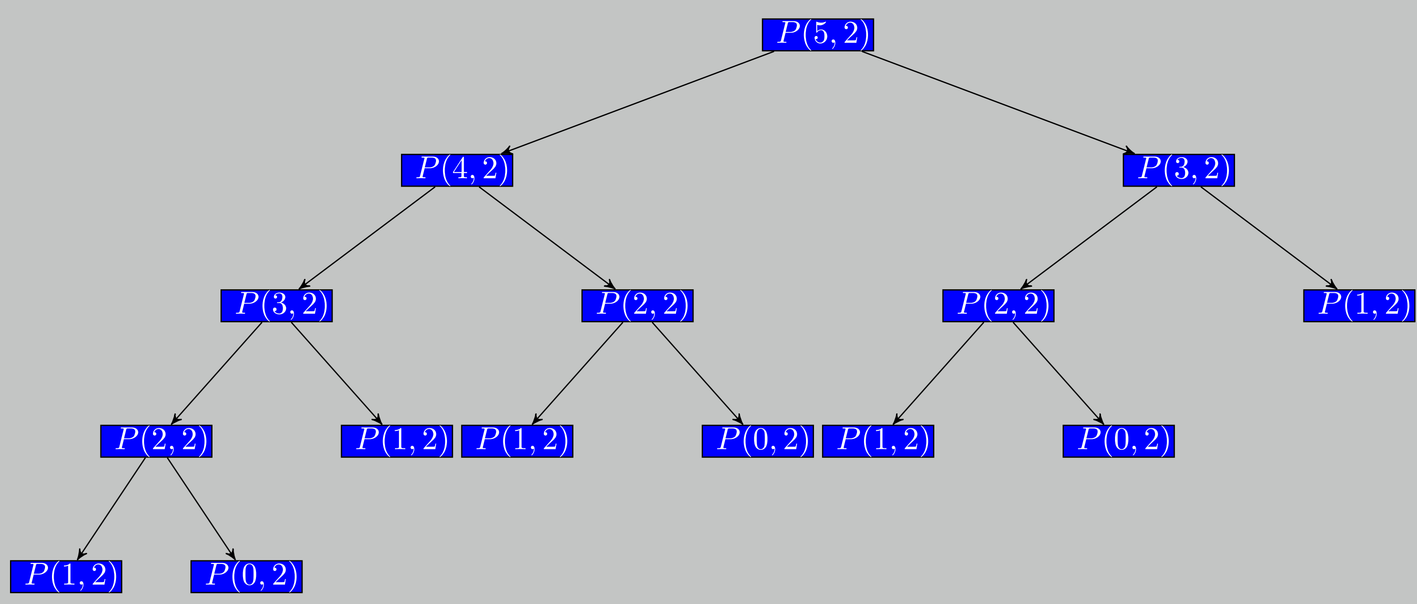
Solução

- Cada caixa pode pertencer à um ciclo de qualquer tamanho.
- Fixada a primeira caixa temos de calcular a probabilidade dessa estar num ciclo de tamanho $\mathbf{t}$.
- Pode-se mostrar que essa probabilidade é $\frac{1}{n}$ para todo $\mathbf{t}$, $\mathbf{1 \leq t \leq n}$.
- Então, definimos $\mathbf{P(n,k)}$ como a probabilidade de uma permutação aleatória de tamanho $\mathbf{n}$ ter seu maior ciclo de tamanho no máximo $\mathbf{k}$.
- Fixado o tamanho $\mathbf{t}$ do primeiro ciclo, temos a seguinte recorrência

$$P(n,k) = \begin{cases} 1, & \text{se } n \leq 1, \\ \sum_{t=1}^{\min(n,k)} \frac{1}{n} P(n-t,k), & \text{caso contrário.} \end{cases}$$

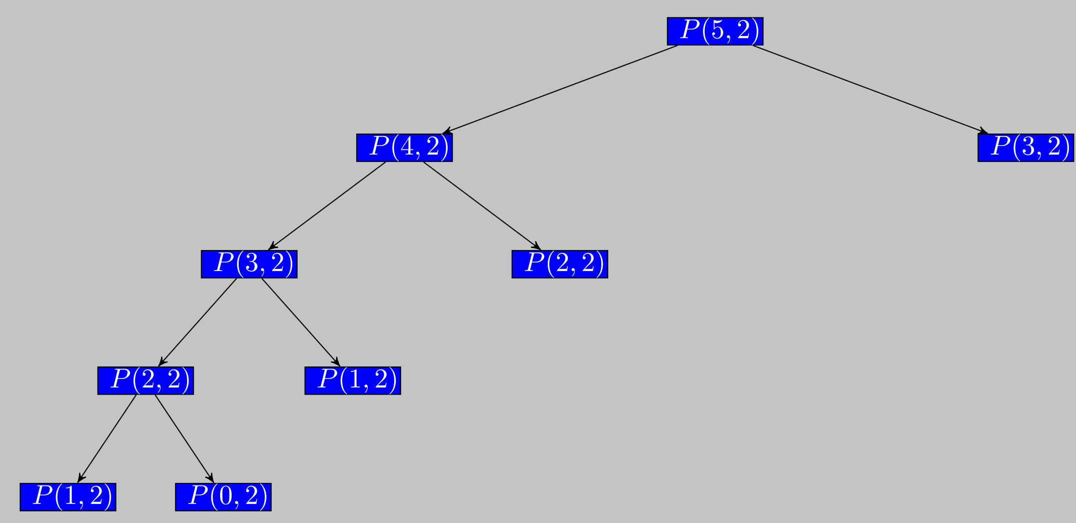
Árvore de recorrência

- Segue a árvore de recorrência para $\mathbf{P(5,2)}$.



Memorização

- Na árvore de recorrência anterior, notamos que os estados são calculados mais de uma vez.
- Usando memorização podemos evitar esse recálculo e tornar o algoritmo mais eficiente.
- Segue a árvore de recorrência com Memorização.



- Usando memorização obtemos um algoritmo $\mathbf{O(n^2)}$.

Melhorando a solução

- Definimos $\mathbf{S(n,k)} = \sum_{i=1}^{n} P(i,k)$.
- Então, $\mathbf{P(n,k)} = \frac{1}{n}S(\min(n,k),k)$.
- Pelas relações anteriores

$$\begin{aligned} S(n,k) &= \tfrac{1}{n}S(\min(n,k),k) + \sum_{i=1}^{n-1} P(i,k) \\ &= \tfrac{1}{n}S(\min(n,k),k) + S(n-1,k). \end{aligned}$$

- Finalmente, $\mathbf{P(n,k) = S(n,k) - S(n-1,k)}$.
- Notamos que $\mathbf{S(n,k)}$ pode ser calculado em tempo $\mathbf{O(n)}$.

Agradecimentos

- Renzo Gomez Dias - revisão do texto.

Informação para contato

- Web: https://linux.ime.usp.br/ stefanot/mac499/
- Email: stefanotommasini26@gmail.com