Programação Dinâmica

Stefano Tommasini
Supervisor: Carlos Eduardo Ferreira

Instituto de Matemática e Estatística - USP

17 de novembro de 2014

Objetivos

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

Introdução

- Técnica utilizada para resolver problemas de otimização.
- Consiste em quebrar o problema em subproblemas menores.
- Definir uma função recursiva para resolver o problema
- Utilizar memorização para reduzir a complexidade

Sobre o Material

- Definição de funções recursivas e estado.
- Muitos exemplos e exercícios com diferentes abordagens de problemas recursivos.
- Importância da Memorização.
- Cálculo de Complexidade.

Descrição

- Grupo de n amigos e n caixas númeradas de 0 a $n-1$, onde cada caixa tem um nome de cada amigo.
- Cada jogador abre K caixas de sua escolha e se não achar seu nome todos perdem.
- Probabilidade de vencer abrindo K caixas aleatórias é $(\frac{K}{n})^n$.
- Algoritmo: Cada jogador abre a caixa correspondente ao seu número enquanto não encontrar seu número se dirige a caixa correspondente ao número que acabou de entrar.
- Definir a probabilidade dos amigos vencerem.

Mais sobre o problema

- Problema trata-se de uma permutação aleatória que corresponde aos nomes escondidos dentro das caixas.
- Estratégia dos amigos corresponde a seguir o caminho no grafo definido pela permutação aleatória.
- Amigo perde se e somente se o ele não voltar para sua própria caixa nesse caminho.
- Problema se resume a achar a probabilidade de uma permutação aleatória qualquer de tamanho n ter seu maior cilco de tamanho menor ou igual a K.

Mais sobre o problema

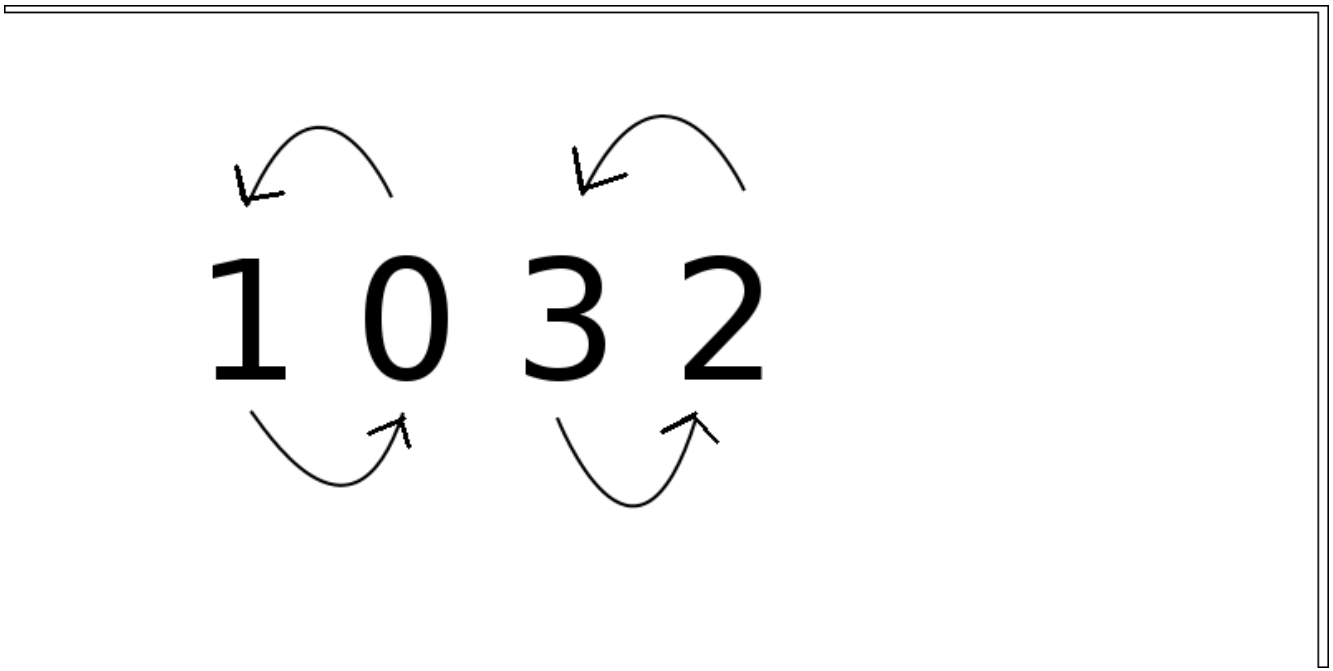


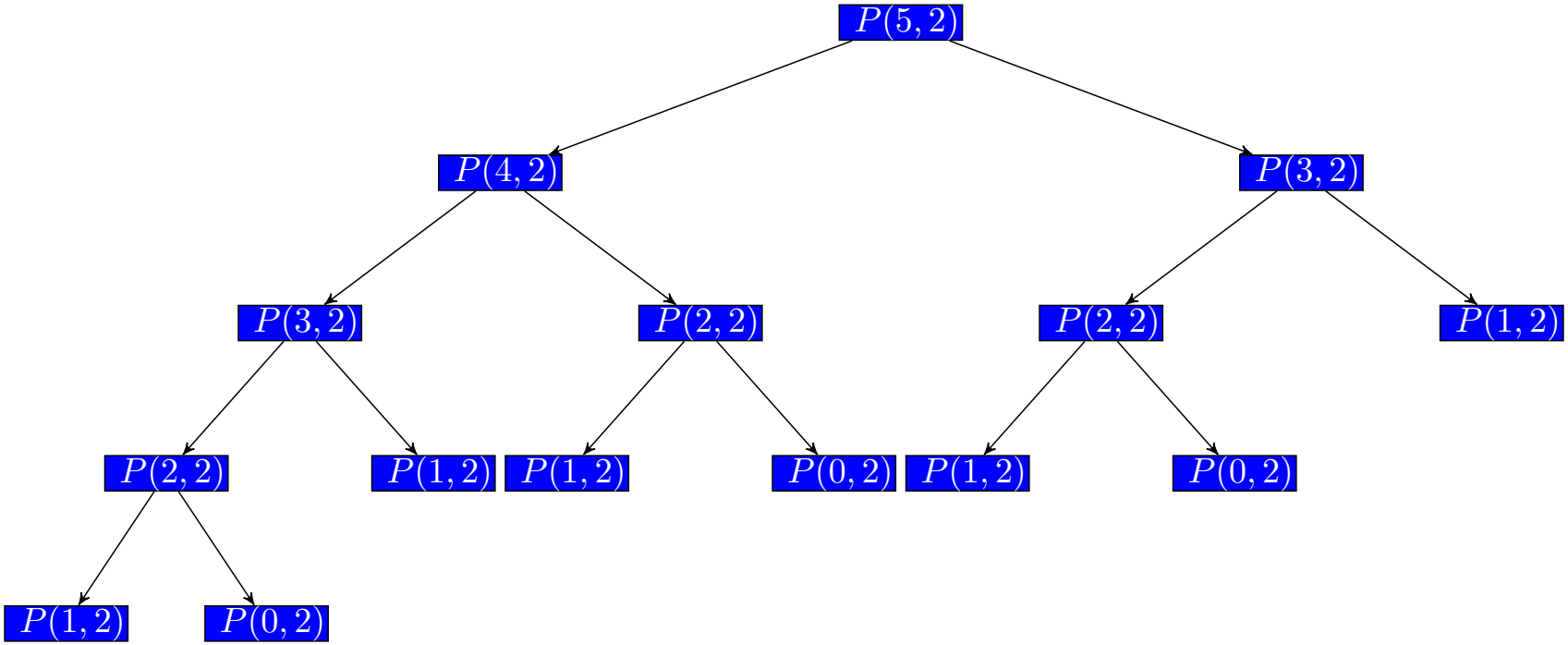
Figura : Grafo de Permutação com $n = 4$

Solução

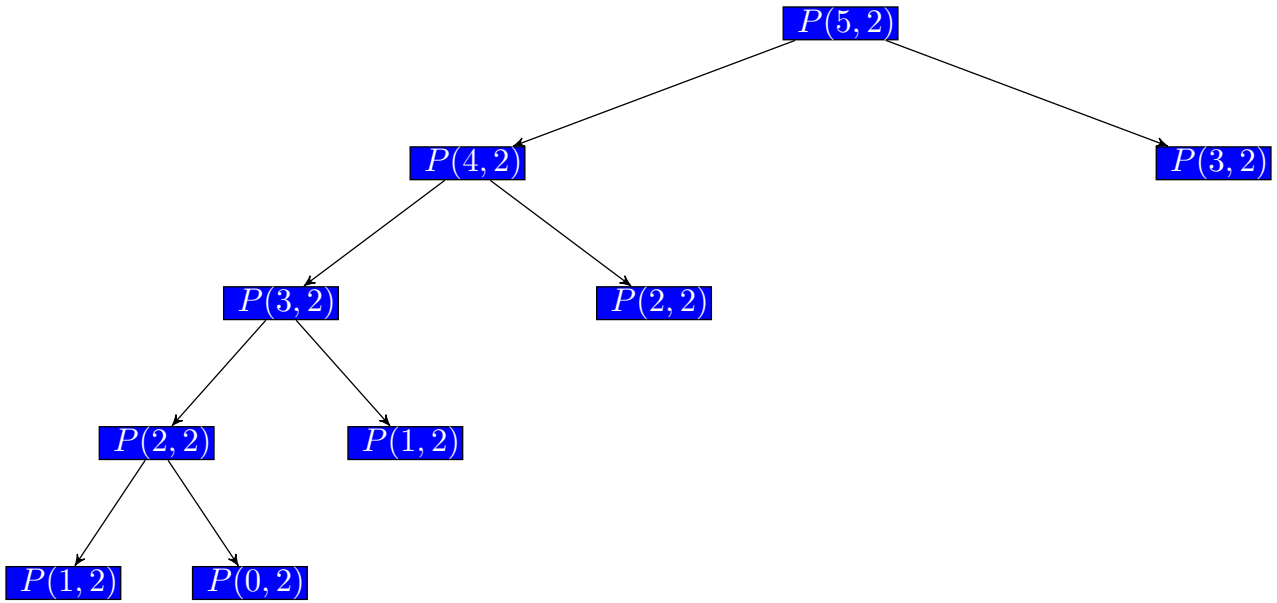
- Cada caixa pode pertencer à um ciclo de qualquer tamanho.
- Fixada a primeira caixa temos de calcular a probabilidade dessa estar num ciclo de tamanho t.
- Pode-se mostrar que essa probabilidade é $\frac{1}{n}$ para todo t, $1 \leq t \leq n$.
- Então, definimos $P(n, k)$ como a probabilidade de uma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

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

- Segue a árvore de recorrência para $P(5, 2)$.



- 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 $O(n^2)$.

Melhorando a Solução

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

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

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

Obrigado!