TÉCNICAS DE OTIMIZAÇÃO DE PROGRAMAÇÃO DINÂMICA

Centro Universitário Serra dos Órgãos
22 de Novembro de 2017
Aluno: Gabriel Duarte
Orientador: Rafael Monteiro

Sumário

Introdução

- As maratonas de programação são competições que exigem criatividade, trabalho em equipe e a capacidade de resolver problemas sob pressão (PIEKARSKI et al., 2015)
- Pode-se destacar: Grafos, Estruturas de Dados, Geometria Computacional e **Programação Dinâmica**
- *Branch Assignment* [1] e *Fundraising* [2]
- Otimizações são temas pouco explorados na literatura

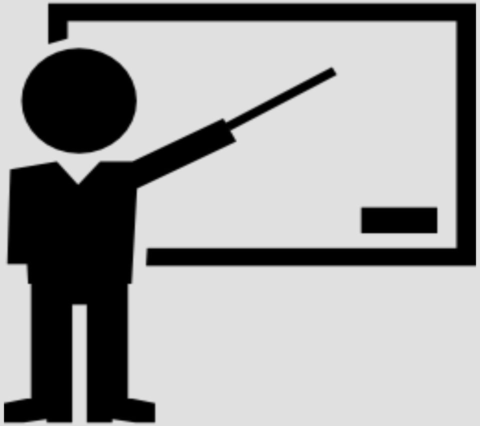
[1]https://icpc.baylor.edu/worldfinals/problems/icpc2016.pdf
[2]http://maratona.ime.usp.br/resultados17/contest.pdf

Objetivo

O objetivo deste trabalho é a criação de um material didático que formalize e explique algumas das principais otimizações de programação dinâmica.

Fundamentação Teórica

Pode ser divida em duas etapas

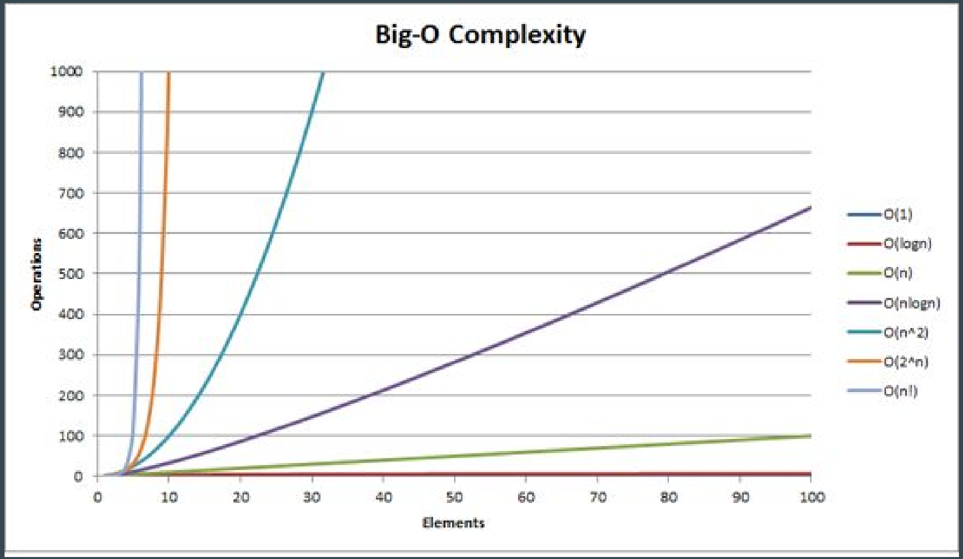


Complexidade de Algoritmos

Relacionado à quantidade de recursos necessários para a execução de um algoritmo, podendo ser quantidade de memória, largura da banda e uso de hardware. Porém, mais frequentemente a preocupação maior é medir o tempo computacional gasto. (CORMEN *et at.*, 2009)

Classes de Complexidade

Ao analisar a complexidade de algum algoritmo, é possível identificar a qual classe este está relacionado.



(PERRETT, 2010)

Programação Dinâmica

Assim como a técnica de divisão e conquista, programação dinâmica funciona dividindo um problema grande em diversos problemas menores e os resolvendo separadamente. Porém, aqui muitos do problemas menores se repetem.

Fibonacci

Observado por Leonardo de Pisa no ano de 1202

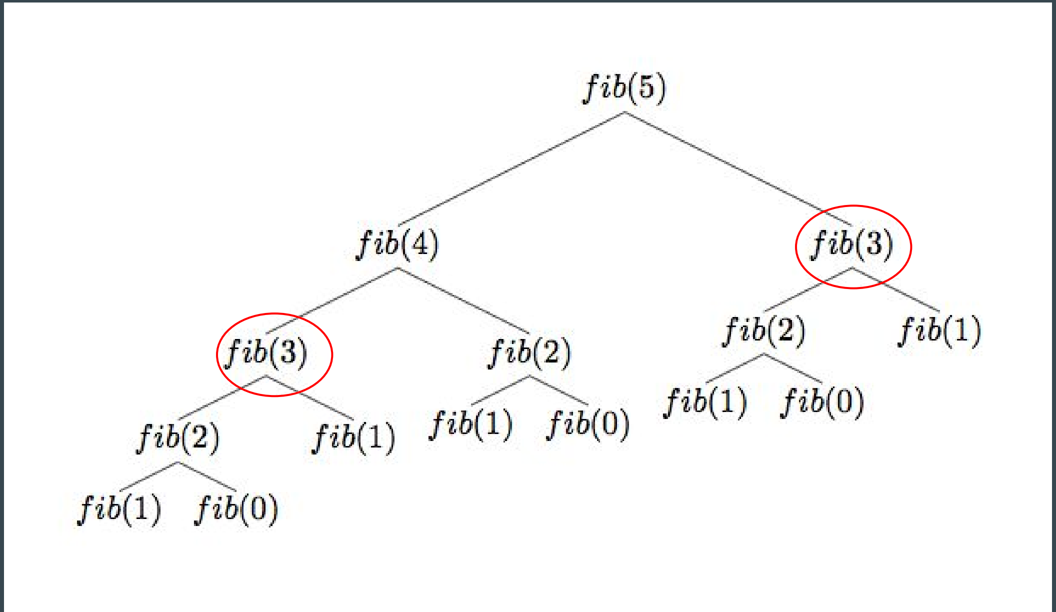
$$0,\ 1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89,\ \ldots$$

Para encontrar o *i*-ésimo elemento da sequência, pode-se utilizar a seguinte recorrência

$$fib(i) = \begin{cases} i & \text{se } i \leq 1, \\ fib(i-1) + fib(i-2) & \text{se } i > 1. \end{cases}$$

Problemas Sobrepostos

Ao calcular o quinto elemento da sequência, vários outros elementos se repetem



(SCHWARTZ, 2011)

Otimizações

Apesar da programação dinâmica já conseguir melhorar a complexidade de muitas soluções, nem sempre apenas o seu uso é suficiente

Nome	**Característica**
Redução de espaço	Redução espacial
Estrutura de dados	$O(n^2) \rightarrow O(n*logn)$
Divide and Conquer	$O(k*n^2) \rightarrow O(k*n*logn)$
Knuth Optimization	$O(n^3) \rightarrow O(n^2)$
Convex Hull Trick	$O(n^2) \rightarrow O(n)$

Ensino de Algoritmos

Diversos trabalhos com o foco de sistematizar a forma de ensinar algoritmos e programação

(SZLÁVI e ZSAKÓ, 2003)

(VIHAVAINEN, PAKSULA e LUUKKAINEN, 2011)

Trabalhos Relacionados

(TOMMASINI, 201-)

(DALALIO, 2013)

(COUTO, 2016)

Metodologia

- Problema
- Solução ingênua
- Análise de particularidades
- “Nome da técnica”
- Benefícios
- Código final

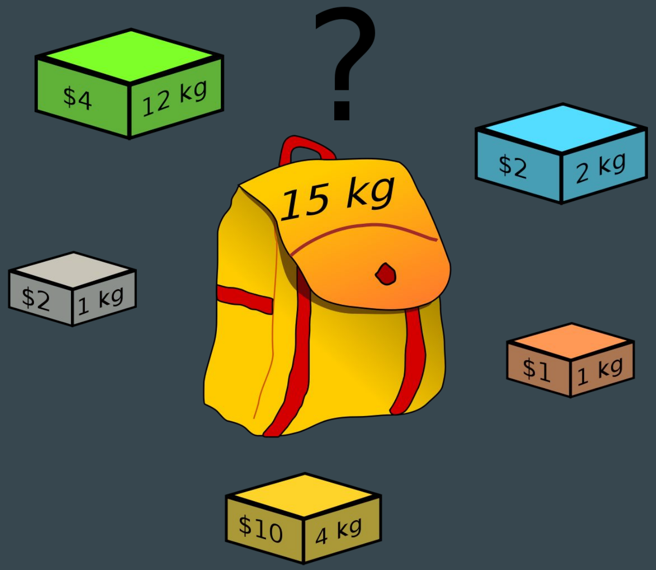
Sugestão de tarefa ao leitor...

Desenvolvimento

Redução de espaço

Colocar na mochila o máximo de itens que maximizem o valor total

Uma solução seria utilizar programação dinâmica e a cada momento testar se é melhor colocar ou não o item na mochila



(WIKIPEDIA, 2017)

$$dp[i][j] = \begin{cases} 0 & se \quad i=0 \;\; ou \;\; j=0, \\ max(v[i-1]+dp[i-1][j-p[i-1]], dp[i-1][j]) & se \;\; p[i-1] \leq j, \\ dp[i-1][j] & se \;\; p[i-1] > j \end{cases}$$

Particularidades

$$dp[i][j]=\begin{cases} 0 & se \quad i=0 \;\; ou \;\; j=0, \\ max(v[i-1]+dp[i-1][j-p\,[i-1]],dp[i-1][j]) & se \;\; p[i-1] \;\leq\; j\,, \\ dp[i-1][j] & se \;\; p[i-1] \;>\; j \end{cases}$$

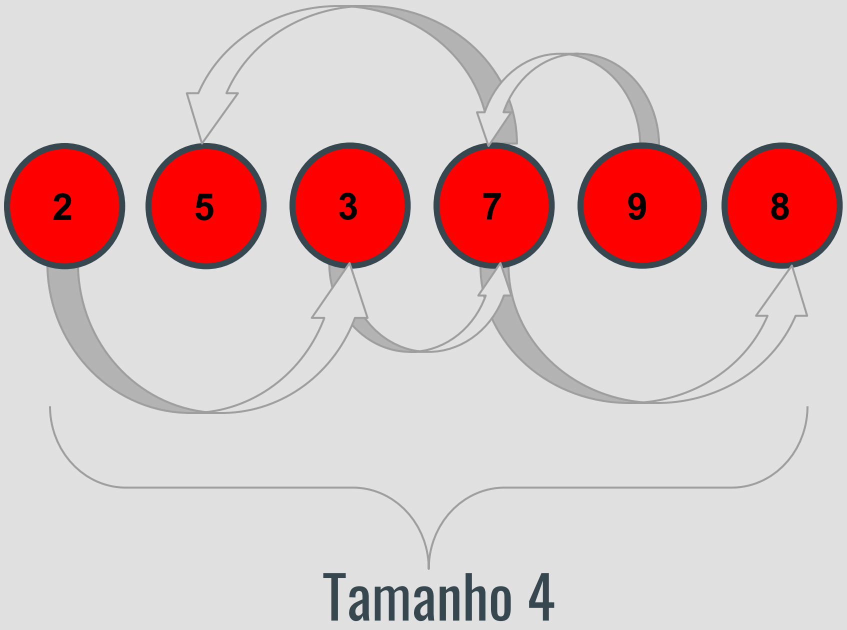
$$\Downarrow$$

$$dp[i\&1][j]=\begin{cases} 0 & se \quad i=0 \;\; ou \;\; j=0, \\ max(v[i-1]+dp[\sim i\&1][j-p\,[i-1]],dp[\sim i\&1][j]) & se \;\; p[i-1] \;\leq\; j\,, \\ dp[\sim i\&1][j] & se \;\; p[i-1] \;>\; j \end{cases}$$

Estrutura de dados RMQ

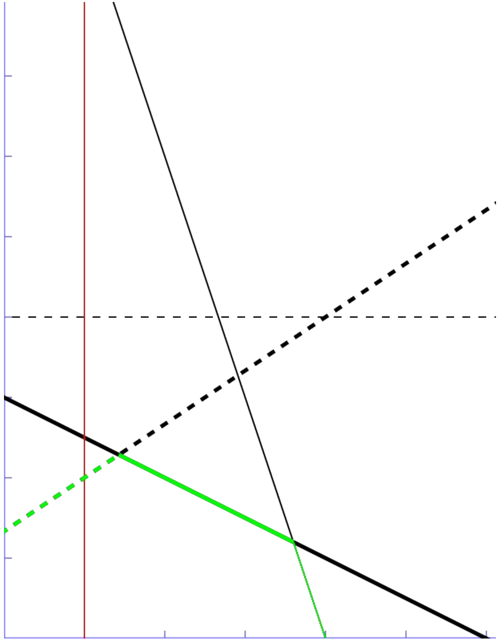
Dado um *array v* deve-se determinar o tamanho da LIS (do inglês, *Longest Increasing Subsequence*)

$$dp[i] = \begin{cases} 1 & se \;\; i=0 \\ max(dp[j]+1)_{0 \leq j < i} & se \;\; i \neq 0 \;\; e \;\; v[j] \leq v[i] \end{cases}$$



Convex Hull Trick

Determinar a melhor reta para um determinado valor de x.



(PEGWIKI, 2016)

Variações

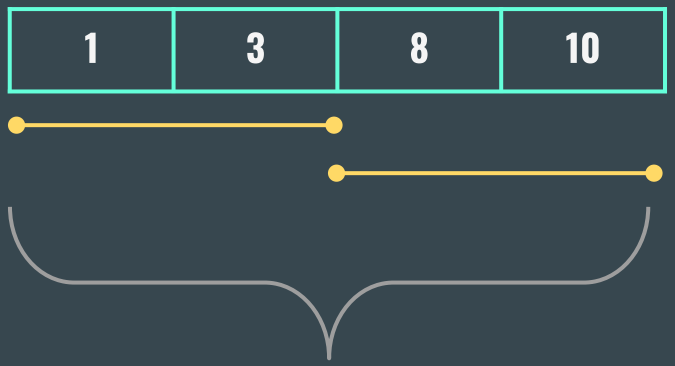
- Tipo 1: Retas ordenadas e as consultas são crescentes
- Tipo 2: Retas ordenadas e sem informação sobre as consultas
- Tipo 3: Retas não ordenadas e sem informação sobre as consultas

O Problema

Deseja-se realizar uma cobertura em alguns pontos de uma estrada com o menor custo possível.

O custo associado para cobrir do ponto x até o y é: $\boldsymbol{(x-y)^2 + c}$

Para C = 5



$(1-3)^2 + 5 + (8-10)^2 + 5 = \mathbf{18}$

Solução

$$dp[i]=\begin{cases} 0 & se \;\; i=0 \\ min_{1\le j\le i}(dp[j-1]+(v[i]-v[j])^2+c) & se \;\; i\neq 0 \end{cases}$$

$$\Downarrow$$

$$dp[j-1]+(v[i]-v[j])^2+c \;\Rightarrow$$

$$-2*v[j]*v[i]+v[j]^2+dp[j-1]+v[i]^2+c \;\Rightarrow$$

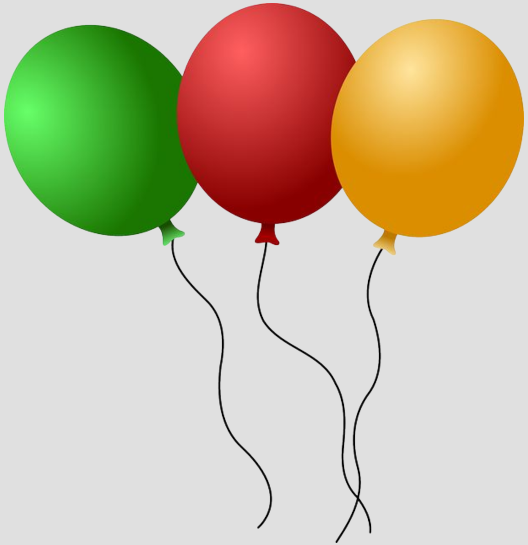
$$Sendo \;\; v[i]=x \;\; e \;\; v[j]=y \;\Rightarrow$$

$$-2*y*x+y^2+dp[j-1]+x^2+c \;\Rightarrow$$

$$Sendo \;\; a=-2*y \;\; e \;\; b=y^2+dp[j-1] \;\Rightarrow$$

$$dp[j-1]+(v[i]-v[j])^2+c \;\Longleftrightarrow\; \boxed{f(x)=ax+b} \;+\; (x^2+c)$$

Considerações Finais



Referências

- CORMEN, T. H. et al. Introduction to Algorithms. Favoritenstrasse 9/4th Floor/1863: The MIT Press, 2009.
- HOM, E. J. What is the Fibonacci Sequence? 2013. Disponível em: <http://www.livescience.com/37470-fibonacci-sequence.html>. Acesso em: 02 abr. 2017.
- PEGWIKI. Convex hull trick - PEGWiki. 2016. Disponível em: <https://wcipeg.com/wiki/Convex_hull_trick>. Acesso em: 09 out. 2017.
- PERRETT, D. CompSci 101 - Big-O Notation. 2010. Disponível em: <http://www.daveperrett.com/articles/2010/12/07/comp-sci-101-big-o-notation/>. Acesso em: 23 abr. 2017.

Referências

- PIEKARSKI, A. E. T. et al. A metodologia das maratonas de programação em um projeto de extensão: um relato de experiência. In: CBIE & LACLO 2015 - IV Congresso Brasileiro de Informática na Educação e X Conferência Latino-Americana de Objetos e Tecnologias de Aprendizagem. [s.n.], 2015. p. 1246–1254. Disponível em: <http://www.br-ie.org/pub/index.php/wcbie/article/view/6276>.
- SCHWARTZ, H. R. Memoization using Closures. 2011. Disponível em: <https://harryrschwartz.com/2011/01/06/memoization-using-closures.html>. Acesso em: 02 abr. 2017.

Referências

- SZLáVI, P.; ZSAKó, L. Methods of teaching programming. 2003. Disponível em: <https://www.researchgate.net/publication/235925815>.
- VIHAVAINEN, A.; PAKSULA, M.; LUUKKAINEN, M. Extreme apprenticeship method in teaching programming for beginners. In: Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education. New York, NY, USA: ACM, 2011. (SIGCSE '11), p. 93–98. ISBN 978-1-4503-0500-6. Disponível em: <http://doi.acm.org/10.1145/1953163.1953196>.

Obrigado!