Trabalho Prático 3: Legado da Copa

Algoritmos e Estruturas de Dados III – 2017/1

Jéssica Taís Carvalho Rodrigues

10 de Julho de 2017

1 Introdução

Uma determinada rua, chamada, "Pai no Bar"possui diversos bares em que o as respectivas casas de seus proprietários estão na mesma rua (sendo do lado oposto ao bar). Cada dono de bar irá fornecer uma bandeirola para enfeitar a rua, de modo que uma ponta esteja ligada ao bar e a outra ligada a casa do dono do bar.
O problema consiste em não deixar que tenham bandeirolas cruzadas e para isso, ver qual o número máximo possível de bandeirolas que podem ser utilizadas, considerando que cada dono irá contribuir com uma e que esta não pode interceptar nenhuma outra.
Para isso, são fornecidos, três modos de resolver esta questão, que serão explicados no decorrer deste documento.

2 Solução do problema

Inicialmente foi construída uma estrutura para representar as bandeirolas, contendo dois inteiros representando o lado par e o lado ímpar.
Conforme solicitado, foram feitos três algoritmos diferentes para gerar a solução, o guloso, o que usa programação dinâmica e o de força bruta.

Para o guloso, calcula-se as interseções dois a dois e se cria uma matriz com o registro desses dados, que é simétrica pois se o lado par ter interseção com o seu respectivo ímpar o mesmo acontece para o inverso. Um vetor armazena o total dessas interseções. Essas interseções são decrementaras e são descartadas da solução, por fim, verifica-se os casos que não se interceptam e estes são somados, gerando o resultado final.

Para o de programação dinâmica, os lados pares foram ordenados e após foi usada a ideia de LIS (Longest Increasing Subsequence) para o lado dos ímpares, que consiste em pegar a quantidade máxima de números para uma sequência ordenada. Fazendo desta maneira a solução vem por consequência, já que na situação apresentada e para a sequência ímpar ordenada (e a par já estava inicialmente), não terão colisões entre as bandeirolas.

Para o força bruta, verifica os possíveis resultados, dentre os que dão certo ou não (ou seja, com colisões ou não), de forma recursiva, marcado 1 para está

na solução ou 0 caso contrário. Passando por todas as opções e então se pega o máximo dentre eles que é o resultado de máximo de bandeirolas.

3 Análise de complexidade

Variável de relevância para a análise:

Variável	Significado
n	Número de casas/bares
b	estrutura de bandeirolas

A complexidade da função main depende do algoritmo a ser executado. Sendo que, considerando o mais custoso, seria

$$O(2^n),$$

por ser o maior tempo dentre as funções executadas no programa.

3.1 Tempo

3.1.1 Bandeirola

Função	Complexidade
max	O(1)
intecepta	O(1)
ordenaBandeirolasPar	O(nlogn)

3.1.2 Guloso

Função	Complexidade
resolveGuloso	$O(n^2)$

3.1.3 Programação Dinâmica

Função	Complexidade
resolveProgDinamica	$O(n^2)$

3.1.4 Força Bruta

Função	Complexidade
resolveForcaBruta	O(2^n)

3.2 Espaço

A complexidade de espaço em todo programa é baseada na estrutura de bandeirolas, que possui os atributos inteiros do lado par e do lado ímpar e na quantidade N delas. Logo a complexidade pode ser dada por O(b*n).

4 Avaliação experimental

Para a avaliação experimental, foram realizados testes simples, chamados de testes toys e outros mais complexos (entradas maiores), chamados de testes gerados para cada um dos algoritmos.

4.1 Testes toys

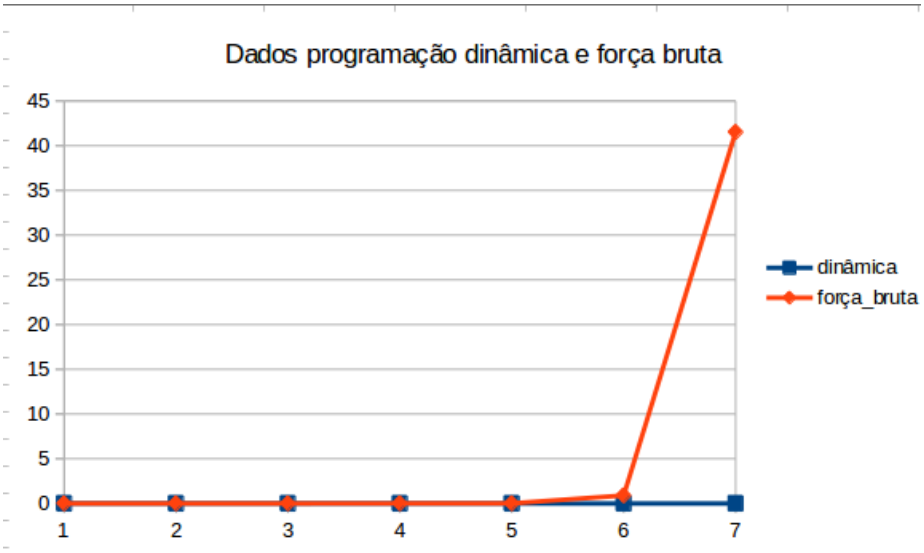
Nestes testes, foram 10 entradas diferentes e a quantidade de pares de casas/bares estava entre 8 a 100. Todos estes testes executaram em menos de 1 segundo para o guloso e para programação dinâmica. No caso da força bruta, tem menos de 1 segundo até 20 pares, para 30 pares gasta 41,524 segundos e então cresce exponencialmente conforme a entrada.

4.2 Testes gerados

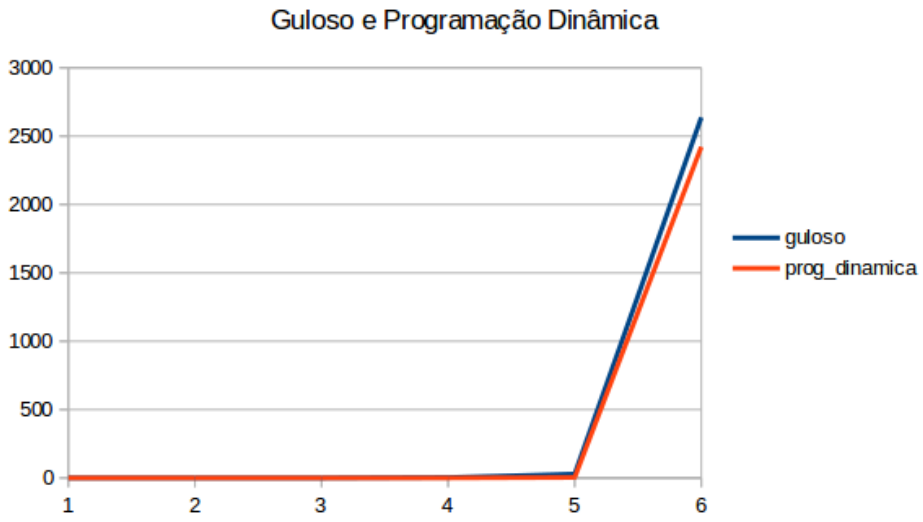
Nestes testes, foram 4 entradas diferentes e a quantidade de pares de casas/bares estava entre 2600 a 32709.
O força bruta não foi testado para estas entradas, devido ao seu alto custo de execução para entradas grandes.
Os outros dois algoritmos retornaram tempos próximos, sem muita discrepância.

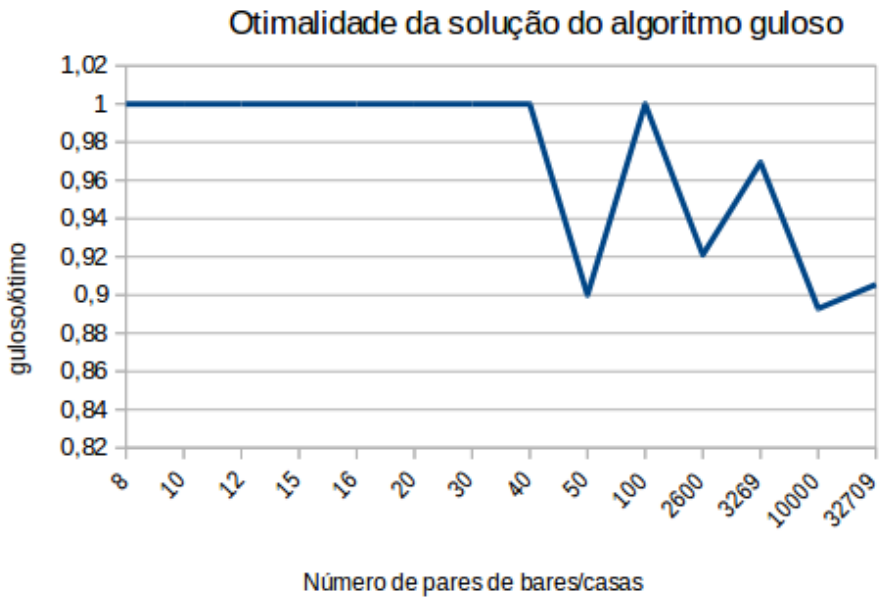
4.3 Análise

E importante, observar que o número de pares bares/casas interferem diretamente no tempo de execução, principalmente para o força bruta que é bem sensível ao tamanho da entrada.
O de programação dinâmica e o guloso são polinomiais e tem custo semelhante, enquanto o força bruta é exponencial.
Não são todos os casos em que o guloso resulta na solução ótima, mas sempre tem o resultado aproximado.
Conforme pode ser visto pelos gráficos abaixo, há a confirmação da análise de complexidade de tempo e algumas comparações interessantes.



Neste último gráfico é possível perceber que o guloso retorna resultados com uma aproximação muito boa, em torno de 90% e 100%.


Guloso e Programação Dinâmica
3000
2500
2000
1500
1000
500
0
1
2
3
4
5
6
guloso
prog_dinamica


Otimalidade da solução do algoritmo guloso
guloso/ótimo
1,02
1
0,98
0,96
0,94
0,92
0,9
0,88
0,86
0,84
0,82
8
10
12
15
16
20
30
40
50
100
2600
3269
10000
32709
Número de pares de bares/casas

5 Conclusão

Foi de imenso aprendizado realizar esse trabalho. Pude entender melhor os conceitos adquiridos em aula e ver qual o funcionamento deles na prática.
Com a comparação dos algoritmos, compreendi as diferenças que formas de implementações podem resultar no programa final.

6 Referências

- Projeto de Algoritmos - Nivio Ziviani
- Longest Increasing Subsequence, disponível em `http://www.geeksforgeeks.org/dynamic-programming-set-3-longest-increasing-subsequence/`