Introdução a Ciências da Computação II - Trabalho 3

Gabriel Fontes, Leandro Satoshi

26 de Novembro de 2018

O relatório a seguir refere-se ao terceiro trabalho solicitado na disciplina de introdução a ciências da computação II, no qual foram designados três problemas para ter suas soluções implementadas, utilizando uma ou mais heurísticas apresentadas durante as aulas. Os problemas apresentados são: encontrar o menor número de palavras em uma partição palíndroma, encontrar a maior progressão aritmética de um conjunto de números inteiros e por fim encontrar de quantas maneiras é possível obter certo número jogando determinado número de dados. Os problemas serão descritos mais detalhadamente posteriormente, em conjunto com a solução encontrada.

1 Problema 1 - Palíndromos

Um palíndromo é uma palavra que permanece a mesma quando lida de trás pra frente. Uma partição palíndroma de uma palavra é uma divisão de uma palavra em um sequência de palíndromos. Tendo em vista a definição dada, é necessário implementar um programa que receba uma *string* e retorne uma partição palíndroma com o menor número de palavras.

1.1 A solução

A solução encontrada para o problema foi um algoritmo guloso, também conhecido como *greedy*, no qual são verificados os extremos da palavra e progredindo em direção ao centro, verificando se as letras são iguais, caso seja verificado que não é um palíndromo, a letra inicial é alterada para a letra seguinte na *string* e o processo se reinicia de forma iterativa. Ao encontrar um palíndromo, essa seção da *string* é cortada e armazenada.

O Programa recebe a *string* como argumento na iniciação, utilizando $argv$.

1.2 Resultados

Como esperado de um algoritmo guloso, ele não é capaz de sempre encontrar a melhor solução, porém é capaz de encontrar uma solução próxima à ideal e com um custo muito menor que um algoritmo de força bruta. Possui um gasto de tempo da ordem de $O(N^2)$, enquanto um algoritmo de força bruta para esse problema seria algo em torno de $O(N^3)$, pois realizaria o algoritmo anterior ao menos N vezes para encontrar todos possíveis divisões da *string*.

2 Problema 2 - Progressões aritméticas

Dado um conjunto de números inteiros não ordenados, é requisitado encontrar o maior sub-conjunto desses números que formam uma progressão aritmética(PA).

2.1 A solução

Para esse problema, o método implementado foi o de força bruta, primeiramente ordenando o vetor de números utilizando o algoritmo *quicksort* e em seguida, a partir do primeiro elemento, utiliza a diferença entre este e seus sucessores para encontrar candidatos a razão da progressão iniciada nesse elemento, armazenando o elemento inicial, a razão e o número de elementos da maior PA. Esse processo é realizado para todos elementos do conjunto. Uma otimização utilizada foi guardar o primeiro elemento, a razão e o número de elementos, ao invés de guardar toda a sequencia.

2.2 Resultados

Como um algoritmo de força bruta, tem um custo alto para encontrar a maior PA, mesmo com otimizações, pois verifica todas as possíveis sequencias, porém é capaz de encontrar com garantia o maior sub-conjunto que forma uma progressão aritmética. Possui um custo de tempo da ordem de $O(N^2 + N \log N)$, sendo um N do caso geral do algoritmo de *quicksort* somado ao gasto de N^2 para verificar as PAs.

3 Problema 3 -Dados

O terceiro e último problema apresentado se trata de um problema com dados. É preciso encontrar de quantas maneiras é possível obter um número X rolando um certo número D de dados, cada um com F faces.

3.1 A solução

A solução encontrada para este problema também foi a utilização de um algoritmo de força bruta, listando todas as possíveis jogadas, somando seus valores e por fim contando quantas resultavam no valor desejado.

3.2 Resultados

Apesar de possuir uma implementação simples e direta, essa solução ainda é um algoritmo de força bruta, e possui um custo e uma ordem de crescimento extremamente alta, sendo F o número de faces, e D o de dados, o algoritmo cresce na ordem de F^D, por ser uma função de duas variáveis, não é possível utilizar a notação de Big-O.