

Universidade de São Paulo – USP  
Instituto de Ciências Matemáticas e de Computação – ICMC  
Departamento de Ciências de Computação – SCC

SCC-218 – Algoritmos Avançados e Aplicações

Estagiários PAE

André Maletzke (andregustavom@gmail.com)  
Filomen Incahuanaco Quispe (fincahuanaco@usp.br)

Professor Gustavo Batista  
gbatista@icmc.usp.br

**Projeto 2 - Programação Dinâmica**  
**Data de Entrega 9/12/2016**

## Objetivo

O objetivo deste trabalho é implementar um algoritmo de *Word Wrap* que minimize a soma dos quadrados dos espaços remanescentes de cada linha, utilizando o paradigma de Programação Dinâmica, e calcular e justificar sua complexidade.

## Introdução

*Word Wrap* é o nome dado à quebra automática de linhas, habitualmente realizada por programas que exibem conteúdo textual, como processadores de texto e navegadores de internet, de forma que tal conteúdo possa ser exibido em determinado espaço sem necessidade de uma barra de rolagem horizontal.

Dada uma lista de palavras que corresponde a linha original a ser quebrada, uma abordagem gulosa para o problema é a inserção das palavras da lista, uma a uma, em uma linha até que não haja mais espaço para a palavra seguinte, para a qual se cria uma nova linha, vazia. Essa abordagem possui como propriedade interessante a obtenção do menor número de linhas necessárias para exibir todas as palavras da lista.

Por exemplo, se o tamanho máximo para a linha for 6 e a linha original for “aaa bb cc dddd”, obtém-se por meio desse algoritmo o seguinte resultado:

aaa bb	6 espaços utilizados, 0 remanescente
cc	2 espaços utilizados, 4 remanescentes
dddd	5 espaços utilizados, 1 remanescente

Observa-se que o espaçamento entre as palavras também é considerado.

Há, porém, formas de obter quebras de linha que ofereçam uma melhor estética ao texto. Uma delas é a minimização da soma dos quadrados dos espaços

remanescentes de cada linha. Aplicando esta minimização ao exemplo anterior, tem-se o seguinte resultado:

aaa	3 espaços utilizados, 3 remanescentes
bb cc	5 espaços utilizados, 1 remanescente
ddddd	5 espaços utilizados, 1 remanescente

O exemplo obtido pela minimização tem como soma dos quadrados dos espaços remanescentes

$$(3 * 3) + (1 * 1) + (1 * 1) = 11$$

Enquanto a solução gulosa oferece como resultado

$$(0 * 0) + (4 * 4) + (1 * 1) = 17$$

Neste trabalho, é esperado o desenvolvimento de um algoritmo que, dada uma sequência de palavras e um valor inteiro  $L$ , ofereça como saída linhas contendo as mesmas palavras, na mesma ordem, da sequência fornecida, separadas por um único espaço em branco ou por uma quebra de linha, de forma que nenhuma linha ultrapasse  $L$  caracteres (incluindo os espaços em branco), e que a soma dos quadrados dos espaços livres remanescentes de todas as linhas seja mínimo.

## Parte 1 (30%): relatório e função de recorrência

A primeira parte do trabalho corresponde a apresentação, em um relatório, da função de recorrência que resolve o problema, seguindo o paradigma de programação dinâmica. Também é exigida uma breve explicação sobre as formulações apresentadas e a complexidade de tempo e de espaço implicadas.

Existem diversas soluções para este problema. Para se obter a nota máxima você deve encontrar uma solução que implique em um consumo de memória proporcional a  $O(n)$  e de tempo proporcional a  $O(n^2)$ , sendo  $n$  o número de palavras da lista.

Escreva um relatório de uma página sobre a sua solução e uma breve explicação da relação de recorrência.

## Parte 2 (70%): implementação

Nesta parte, é exigida a codificação um programa que implemente a função obtida na parte anterior, de forma iterativa **ou** recursiva. A implementação deve lidar com a entrada e oferecer a saída conforme especificadas a seguir:

### Entrada

A entrada do programa começa com um valor inteiro  $L$  tal que  $0 < L < 81$ , e que corresponde ao tamanho máximo, em número de caracteres, das linhas exibidas na saída.

Após  $L$ , segue um número  $N$  que corresponde ao número de palavras.

Logo em seguida, seguem  $N$  linhas cada uma com uma única palavra. A quantidade de palavras é sempre menor ou igual a 1000, e o tamanho de qualquer palavra é menor ou igual a  $L$ .

**Obs:** Por palavra, entende-se qualquer caractere aceito pelo padrão `[^ \t\r\n]+`.

## Saída

A saída do programa é formada por linhas de tamanho menor ou igual a  $L$  (contando espaços em branco), contendo as palavras da entrada na mesma ordem que foram fornecidas, separadas por um espaço em branco (' ') ou por uma quebra de linha ('\n'), de forma que as somas dos quadrados dos espaços livres remanescentes de todas as linhas seja mínimo.

**Obs:** A última linha deve preceder um '\n', que deve ser o último caractere da saída.

## Exemplo de entrada

```
6
4
aaa
bb
cc
ddddd
```

## Exemplo de saída

```
aaa
bb cc
ddddd
```