Trabalho Prático 3: MyTex

Cleiton Neves Santos

June 26, 2016

1 Introdução

O trabalho consiste em, dado um arquivo de texto qualquer, justificá-lo usando três técnicas diferentes: programação dinâmica, método exaustivo e uma heurística gulosa. Minimizando a seguinte expressão de custo:

$$k * (H - |l|)^x + \sum_{\forall li \in l} k * (L - length(li))^x.$$

(L comprimento máximo de uma linha; H tamanho da página em linhas;$|l|$ número de linhas; $length(li)$ comprimento da linha i; k e x parâmetros dados na entrada.)

2 Solução do Problema

2.1 heurística gulosa

Para resolução com heurística gulosa foi implementado o programa descrito no algoritmo a seguir:

```
input  : texto a ser dividido em diversas linhas; máximo de linhas na página H;
         máximo de caracteres na linha L
output: texto justificado
begin
    while não é fim do texto do
        lê palavra s do texto;
        if tamanho do texto escrito na saida + tamanho de s < L then
            escreve L na saída
        else
            coloca quebra de linha e escreve na saída
        end
    end
end
```

Algorithm 1: greedy

Que coloca uma quebra de linha assim que o número de caracteres na linha vá ultrapassar a quantidade máxima permitida.
Esse algoritmo sempre tenta maximizar o número de caracteres por linha, isso nem sempre é uma boa estratégia como podemos ver no seguinte exemplo:

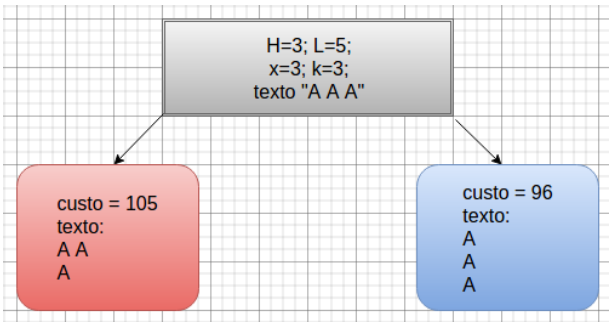

Figure 1: em vermelho, a esquerda, resultado usando algoritmo guloso

Isso ocorre pois a política gulosa minimiza apenas a segunda parte da função de custo (o somatório) assim quando $H > L$ (H máximo número de linhas da página; L número máximo de caracteres na linha) o algoritmo não será ótimo.

2.2 força bruta

A resolução do problema utilizando força bruta se deu da seguinte maneira:

°Foi criado um vetor com o tamanho da quantidade de espaços no texto em que cada posição contém "1" ou "0" indicando se o espaço deve ou não ser substituído por uma quebra de linha;

°A função $brute\_force()$ recebe esse vetor calcula o custo de acordo com a função a ser minimizada, caso o custo calculado seja menor que o mínimo encontrado até o momento o mínimo é atualizado;

°A função chama a si mesma duas vezes setando uma posição do vetor de espaços para "1" e na outra "0".

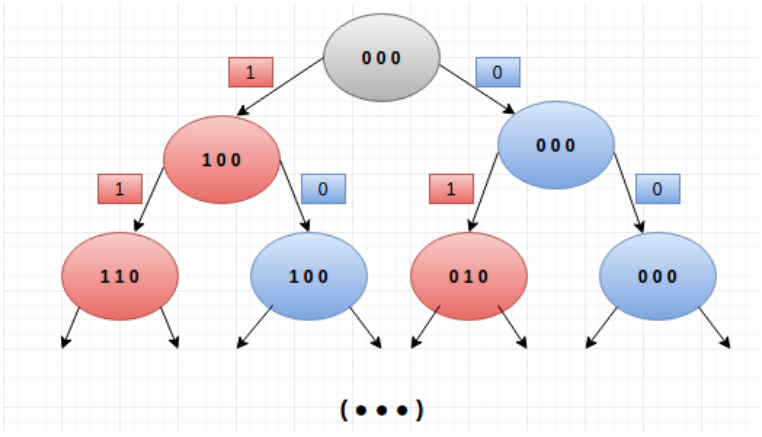

Figure 2: processo força bruta

3 Análise de Complexidade

3.1 heurística gulosa

3.1.1 tempo

A complexidade do método de resolução com heurística gulosa implementado é $O(N)$ sendo N o número de caracteres do texto. Pois para cada *string* lida do texto é calculado o tamanho da mesma em $O(n)$ sendo n o tamanho da palavra enquanto concomitantemente é calculada o valor da função de custo.

3.1.2 espaço

A complexidade de espaço do método é $O(H * (L + 1))$ onde L e H são respectivamente o limite de caracteres por linha e o limite de linhas por página. Pois de acordo com os valores de L e H dados na entrada uma matriz que comporte o tamanho máximo do texto é alocada.

3.2 força bruta

3.2.1 tempo

A complexidade da resolução por força bruta é de $O(N * 2^E)$, sendo E o número de espaços no texto e N o total de caracteres. Pois para cada espaço do texto a função $brute\_force()$ irá se chamar recursivamente duas vezes executando o cálculo da função de custo em $O(N)$ quando for folha da árvore de recursão.

3.2.2 espaço

A complexidade de espaço utilizado na na resolução é $O(N)$, sendo N o total número de caracteres que é o tamanho do vetor onde é armazenado o texto.

4 Análise Experimental

Para realizar os experimentos foi utilizado um gerador de casos de teste cedido por um colega e para medir o tempo foi usado o comando "time" no terminal linux. Cada instância fornecida pelo gerador foi executada 5 vezes e o tempo de execução foi obtido retirando a média de tempo das 5 execuções.Os testes foram realizados em uma máquina com sistema operacional Ubuntu 14.04, processador intel celeron 1.80GHz e com 4GB de memória ram.

O primeiro gráfico a seguir exibe como o tempo de execução do método $forCa\_bruta$ cresce com relação ao *guloso*.

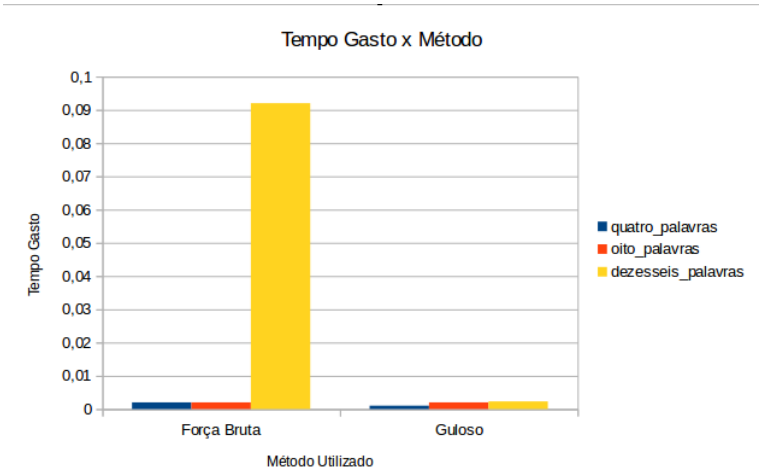

Figure 3: tempo x método

Como já era previsto, em troca de uma solução ótima, o algoritmo de força bruta gasta um tempo muito maior.

5 Conclusão

Neste trabalho foi resolvido o problema de encontrar uma configuração de quebra de linhas em um texto que minimiza uma função dada utilizando um método guloso e a força bruta. O método guloso se mostrou rápido porém não preciso na maior parte dos casos. O método de força bruta é preciso, porém muito demorado. A solução para esse dilema seria a utilização de programação dinâmica.

6 Referências

- Gerador de casos de teste - https://gist.github.com/ivanjr0/2e49198c09b9139be0f253270e74bc28