"Onde estão os doces?" Soluções para o Problema da Rua Encantada

Rossana Baptista Queiroz[1]
[1] Pontifícia Universidade Católica do Rio Grande do Sul (PUC-RS)
Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Porto Alegre – RS – Brasil
fellowsheep@gmail.com

Resumo: *Este trabalho apresenta três possíveis soluções para um problema chamado "A Rua Encantada", proposto na disciplina Computabilidade e Complexidade de Algoritmos no segundo semestre de 2008. A primeira solução procura resolver o problema de forma intuitiva através de um algoritmo iterativo, sem a preocupação de otimizar o custo do número de operações. De fato, essa solução é bastante custosa, a ponto de não ser possível computá-la em tempo viável para o tamanho de entrada requerido no problema. Em vista disso, são apresentados dois algoritmos que utilizam conhecimentos matemáticos sobre a natureza do problema. Os resultados mostram que a solução matemática é mais eficiente que a primeira solução proposta, tornando viável a obtenção de todos o resultados para o problema.*

Palavras-chave: complexidade, algoritmos, números triangulares, quadrados perfeitos

I Introdução

Este trabalho apresenta a resolução de um problema chamado "A Rua Encantada", proposto como primeiro trabalho da disciplina *Computabilidade e Complexidade de Algoritmos*[1], ministrada no segundo semestre de 2008. O objetivo do trabalho é investigar o problema e propor um algoritmo que o solucione, buscando otimizar recursos como, por exemplo, o número de operações executadas.

Apresentado em forma de uma extensão do conto infantil "Chapeuzinho Vermelho", o enunciado do problema apresenta o seguinte cenário: a personagem *Chapeuzinho Vermelho* é incumbida por sua mãe de buscar os doces que serão presenteados a sua avó na confeitaria de sua cidade. No entanto, o número do estabelecimento é esquecido por sua mãe, que lembra apenas da seguinte propriedade relativa a ele: partindo da primeira casa da rua, sempre somando os números das casas subseqüentes até a casa imediatamente antes da confeitaria, e da casa logo após a confeitaria até a última casa da rua, ambos os resultados são iguais. Uma vez que os personagens da estória parecem ter limitações para a resolução de problemas matemáticos, o desafio de descobrir o número da confeitaria é repassado aos alunos da disciplina. O enunciado ainda especifica que o número da primeira casa da rua é 1, e determina que a busca deve ser feita em ruas com até 400.000.000 casas.

A Figura 1 ilustra o problema, considerando uma rua de tamanho 8 (número de casas). Como mostrado abaixo, observa-se que a confeitaria fica na casa de número 6. De fato, somando-se os números das casas 1 a 5 (casas anteriores) e de 7 a 8 (casas posteriores), obtém-se o valor 15.

[1] Disciplina obrigatória do Programa de Pós Graduação em Ciência da Computação da Pontifícia Universidade Católica do Rio Grande do Sul

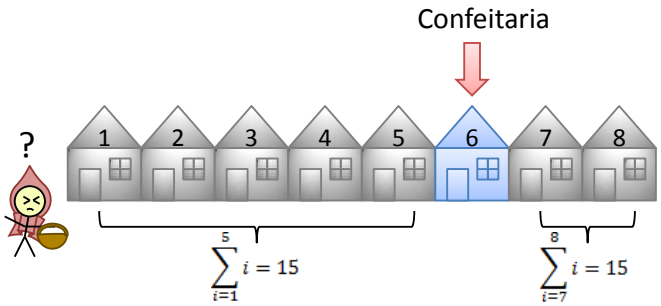

Figura 1: Ilustração do problema, considerando uma rua com 8 casas. Neste caso, observa-se que a confeitaria é a casa de número 6.

O artigo está organizado da seguinte maneira: a Seção II apresenta três possíveis soluções para o problema. Primeiro, é proposto um algoritmo "simples" (Subseção A), desenvolvido sem uma análise mais cuidadosa da natureza do problema. Em seguida, é apresentada uma solução obtida através de uma análise mais aprofundada do problema, apontando características que conferem com conceitos de Números Poligonais encontrados na Matemática, mais precisamente, em Teoria dos Números. Dessa análise, cujo produto é a modelagem matemática do problema, são apresentados dois algoritmos que o solucionam: (Subseção B). Em seguida, é apresentada a solução dos possíveis números de confeitaria para ruas de tamanho 1 a 400.000.000 e um quadro comparativo dos tempos de execução de cada um dos algoritmos apresentados. Por fim, são feitas algumas considerações com respeito à complexidade dos algoritmos propostos e como isso refletiu na prática (tempo de execução).

II Análise do Problema e Soluções

Esta seção apresenta três soluções encontradas para o problema apresentado na seção anterior. É importante salientar que essas não são as três únicas soluções; diferentes outros algoritmos podem ser escritos que resolvam o problema, com diferentes custos computacionais.

A Subseção A mostra um algoritmo construído sem levar em consideração possíveis propriedades (matemáticas, por exemplo) que o problema pudesse apresentar. Em outras palavras, trata-se um algoritmo que procurou modelar a lógica do problema de forma intuitiva, sem preocupação com a sua performance. O motivo de manter essa solução no artigo é para mostrar que uma solução aparentemente "simples" pode possuir um custo computacional tão alto que a torna impossível de obter os resultados para todos os tamanhos de entradas do problema em tempo viável. A Subseção B apresenta uma análise matemática do problema, que o torna possível de ser modelado através de algoritmos mais eficazes.

A Uma solução simples

Partindo do enunciado do problema, uma maneira de solucioná-lo é através de um algoritmo que percorre todo o espaço correspondente ao domínio apresentado (ruas de tamanho 1 até 400.000.000) seguindo o seguinte raciocínio: para determinado número i, verifica-se se ele pode ser o número da confeitaria em uma rua de tamanho t, desconhecido *a priori*. Para isso, calcula-se a soma de 1 até o número anterior a i, e depois verifica-se se, somando a partir do número sucessor de i, a soma dos seus subseqüentes será igual à dos seus anteriores. O algoritmo pára sua verificação para i no momento que a soma dos posteriores for maior ou igual à dos anteriores. Se for igual, então i é um número da confeitaria válido. São fornecidos, então, os valores i e t, que é i mais o número de sucessores que foram somados até se obter a igualdade esperada.

O Algoritmo 1 apresenta a solução descrita. O algoritmo proposto possui dois laços e atua sobre quatro variáveis. As quatro variáveis (chamadas de i, *tamanhoDaRua*, *SomaAntesDeI* e *somaDepoisDeI*) armazenam, respectivamente, o número "candidato" a número de confeitaria, o tamanho da rua, e as somas dos números antes e depois de i, como variáveis auxiliares. O primeiro laço executa enquanto ainda não se houver chegado ao tamanho máximo de rua. Dentro desse laço, i sempre é incrementado de uma unidade e acumula-se a soma dos números antes de i. O segundo laço, portanto, vai somando números sucessores a i até que esse número seja maior ou igual à soma dos seus números anteriores. O tamanho da rua recebe no primeiro laço o valor de i e é acrescido de uma unidade a cada iteração do segundo laço. Se a soma dos números posteriores a i for igual à soma de seus números sucessores (até o tamanho da rua), então foi encontrado um número de confeitaria .

```
Entrada: o número max máximo de casas da rua
Saída: todas as possibilidades de números de confeitaria em ruas de tamanho 1 a max, se
       houver
i ← 1;
enquanto tamanhoDaRua ≤ max faça
    somaAntesDeI +← i-1;
    somaDepoisDeI ← 0;
    tamanhoDaRua ← i;
    enquanto somaDepoisDeI ≤ somaAntesDeI faça
        tamanhoDaRua +←1;
        somaDepoisDeI +← tamanhoDaRua;
    fim
    se somaAntesDeI = somaDepoisDeI então
        Imprimir que achou a confeitaria i no tamanho de rua tamanhoDaRua
    fim
    i +←1;
fim
```

Algoritmo 1: Primeiro algoritmo proposto

Para cada candidato a número de confeitaria i, são efetuadas x_i iterações, até que a soma dos números posteriores fique maior ou igual à soma dos números anteriores a i. Esse número x_i é um número menor que i, uma vez que, como os números posteriores a i são maiores que os anteriores, então para que a soma obtida seja maior ou igual à soma dos anteriores são necessários menos que $i-1$ números. Pode-se expressar esse número x_i em função de i, como por exemplo, através da razão entre o número de operações[2] do segundo laço (em i) e i. Dessa forma, número de operações em cada laço é $i \times x_i$. Portanto, o número de operações total desse algoritmo pode ser expresso como $1x_1 + 2x_2 + 3x_3 + \ldots + nx_n$, sendo que n refere-se ao número de iterações executadas no primeiro laço. Outra forma de escrever isso é $\sum_{i=1}^{n} ix_i$, sendo que $x_i < i - 1$. Ao executar o algoritmo, percebe-se que essa razão entre o número de operações executadas e i é aproximadamente uma constante, a saber, $0,41$. Considerando então x_i uma constante C, têm-se que o número de operações para uma entrada de tamanho n é $C \times \sum_{i=1}^{n} i$, ou seja, aproximadamente $C \times \frac{n^2+n}{2}$ operações. Trata-se, portanto, de um algoritmo com comportamento quadrático $O(n^2)$, apesar de que o número de operações sempre será menor que n^2.

É importante observar também que o número n de iterações do primeiro laço, é sempre um número menor que o tamanho máximo da rua, uma vez que o tamanho da rua é incrementado sempre no segundo laço. Mesmo assim, vê-se que o algoritmo proposto nessa seção, cujo número de operações cresce com comportamento quadrático, torna-se inviável para o cálculo em cima do tamanho de entrada máximo proposto (Seção III apresenta uma discussão a esse respeito). Em vista disso, fêz-se uma análise mais detalhada do problema, cujo produto foi a descoberta de

[2]Neste contexto, o número de operações refere-se ao número de iterações, isto é, o número de vezes que o algoritmo entrou nos laços.

algumas propriedades matemáticas dele e a proposta de outros algoritmos que fazem o uso dessas propriedades, otimizando assim o número de operações de forma surpreendente.

B Solução Matemática

Com o objetivo de se encontrar uma solução viável, foram investigadas formas de modelar o problema matematicamente. Como observado na solução proposta na seção anterior, que para cada entrada de tamanho n, são necessárias cerca de $\frac{n^2+n}{2}$ operações, que é o somatório de 1 até n.

Em Matemática, existe o conceito de *números poligonais*, que são aqueles números que podem ser arranjados como um polígono regular [1]. Um número triangular, por exemplo, é um número que pode ser representado por um triângulo equilátero [4], como mostra a Figura 2. Para se calcular o n-ésimo número triangular de uma seqüência, basta fazer a soma dos números de 1 até n. Percebe-se, portanto, que o problema investiga a possibilidade de se encontrar um número de confeitaria dentro de uma seqüência de números triangulares (soma dos números do início ao fim da rua é sempre triangular).

O problema pode ser expresso da seguinte maneira: para que seja encontrada uma confeitaria de número s em uma rua de tamanho t, é necessário que exista um s inteiro cuja soma de 1 até $s-1$ e de $s+1$ até t sejam iguais. A Equação 1 expressa essa igualdade. No problema proposto, observa-se que, por definição, a soma dos números anteriores ao número da confeitaria é um número triangular. Conseqüentemente, a soma dos números posteriores também será. A Figura 2 exemplifica o problema e mostra o arranjo triangular dos números anteriores à confeitaria de número 6, encontrada na rua de tamanho 8.

$$\sum_{i=1}^{s-1} i = \sum_{i=s+1}^{t} i \tag{1}$$

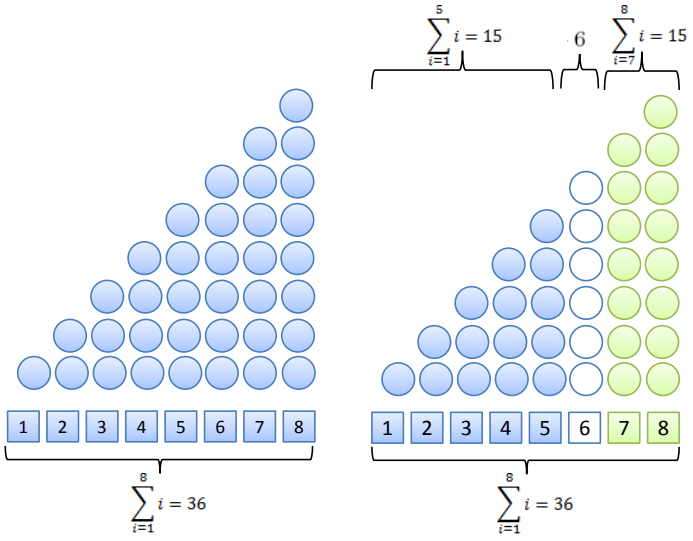

Figura 2: Ilustração do problema, considerando uma rua com 8 casas. A primeira imagem mostra que a soma dos números de 1 até 8, por definição, é um número triangular. A segunda imagem demonstra, ainda na representação triangular, que existe um número inteiro entre 1 e 8 cuja soma de seus números anteriores e de seus posteriores é igual.

Como ilustra a segunda parte da Figura 2, o somatório do tamanho da rua pode ser expresso, de acordo com o problema, da seguinte maneira:

$$\sum_{i=1}^{t} i = \sum_{i=1}^{s-1} i + s + \sum_{i=s+1}^{t} i \tag{2}$$

Para que s seja inteiro, é necessário que a soma de 1 até t seja um quadrado $s \times s$. Isso quer dizer que, para que se encontre uma confeitaria s em uma rua de tamanho t, o somatório de 1 até t precisa ser igual a s^2. A Figura 3 ilustra essa propriedade pelo arranjo da soma dos números do exemplo anterior. Em outras palavras, pode-se afirmar que o somatório de 1 a t, além de triangular (por definição), precisa ser também um número quadrado. Trata-se, portanto, da seqüência de números que são triangulares e quadrados perfeitos ao mesmo tempo.

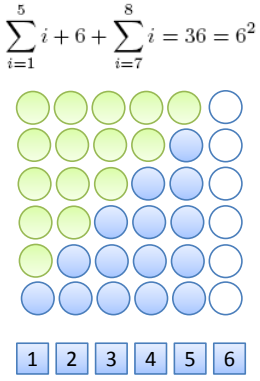

Figura 3: Ilustração do problema, arranjado em forma de um quadrado perfeito. A soma dos números de 1 a 6 e de 7 a 8 são números triangulares, por conseqüência da definição do problema. Acrescentando-se o número 6 para completar a soma dos números de 1 a 8 (tamanho da rua), forma-se um número quadrado.

Com essas informações, é possível construir um algoritmo (Algoritmo 2, abaixo) que, para cada entrada n, calcula o somatório de 1 a n e depois verifica se esse número é um quadrado perfeito (raiz quadrada inteira).

Entrada: o número max máximo de casas da rua
Saída: todas as possibilidades de números de confeitaria em ruas de tamanho 1 a max, se houver

```
i ← 1;
somatorio ← 0;
enquanto i ≤ max faça
    somatorio ←i;
    se sqrt(somatorio) for inteira, então
        Imprimir que achou a confeitaria sqrt(i) no tamanho de rua i
    fim
    i +←1;
fim
```

Algoritmo 2: Segundo algoritmo proposto

Para se obter uma fórmula matemática que determine a seqüência dos números que sejam triangulares e quadrados ao mesmo tempo, parte-se da igualdade $\frac{t^2+t}{2} = s^2$. Uma fórmula geral para encontrar o n-ésimo número triangular e quadrado perfeito de uma seqüência é, segundo [3, 2]:

$$TQ_n = \frac{1}{32}\left[(17 + 12\sqrt{2})^n + (17 - 12\sqrt{2})^n - 2\right] \tag{3}$$

Outra maneira de se calcular é pela seguinte recursão linear [3, 2]:

$$TQ_n = 34TQ_{n-1} - TQ_{n-2} + 2, \tag{4}$$

com $TQ_0 = 0$ e $TQ_1 = 1$ como base da recursão. O n-ésimo número triangular quadrado obtido é igual ao s-ésimo quadrado perfeito (número da confeitaria) e t-ésimo número triangular (tamanho da rua) da seguinte maneira:

$$s_n = \sqrt{TQ_n} \tag{5}$$

$$t_n = \lfloor \sqrt{2TQ_n} \rfloor \tag{6}$$

Utilizando-se uma das fórmulas das equações 3 e 4 como função $TriangularQuadrado(n)$, pode-se obter outra solução para o problema, expressa no Algoritmo 3.

```
Entrada: o número max máximo de casas da rua
Saída: todas as possibilidades de números de confeitaria em ruas de tamanho 1 a max, se
       houver
i ← 1;
t ← 0;
enquanto t ≤ max faça
    tq ←TriangularQuadrado(i);
    s ←sqrt(tq);
    t ←sqrt(2*tq);
    se t < max então
        Imprimir que achou a confeitaria s no tamanho de rua t
    fim
    i +←1;
fim
```

Algoritmo 3: Terceiro algoritmo proposto

Observa-se que os algoritmos 2 e 3 possuem comportamento linear $O(n)$ (apenas um laço para os tamanhos de rua de 1 a n). O Algoritmo 2, de fato, executa exatamente max (400.000.000) iterações. No Algoritmo 3, no entanto, o teste do laço é feito sobre t (tamanho da rua), assim como no Algoritmo 1. Nesse algoritmo, no entanto, são calculados diretamente, através da aplicação da fórmula da Equação 6, somente os números de rua cujo somatório é um número triangular e quadrado perfeito. Portanto, pode-se dizer que este algoritmo executa somente o número de soluções possíveis de números de confeitaria para ruas de 1 a max (apresentado na próxima seção) acrescido de uma unidade (ele executa uma vez para t maior que max). As funções que calculam o n-ésimo número triangular quadrado também são lineares. Optou-se implementar a função recursiva para esse trabalho. A próxima seção apresenta a solução do problema e os tempos de execução dos algoritmos propostos.

III Resultados

Pela execução dos algoritmos 2 e 3, obteve-se os seguintes resultados (para $max = 400.000.000$):

```
Rua de tamanho 1: a confeitaria está na casa 1
Rua de tamanho 8: a confeitaria está na casa 6
Rua de tamanho 49: a confeitaria está na casa 35
Rua de tamanho 288: a confeitaria está na casa 204
```

```
Rua de tamanho 1.681: a confeitaria está na casa 1.189
Rua de tamanho 9.800: a confeitaria está na casa 6.930
Rua de tamanho 57.121: a confeitaria está na casa 40.391
Rua de tamanho 332.928: a confeitaria está na casa 235.416
Rua de tamanho 1.940.449: a confeitaria está na casa 1.372.105
Rua de tamanho 11.309.768: a confeitaria está na casa 7.997.214
Rua de tamanho 65.918.161: a confeitaria está na casa 46.611.179
Rua de tamanho 384.199.200: a confeitaria está na casa 271.669.860
```

O algoritmo 1, como já discutido brevemente na Subseção A, possui comportamento quadrático. Foram feitas algumas análises (coleta do número de operações e tempo[3] dentro do segundo laço do algoritmo) e apenas resultados parciais foram obtidos. Rodando-se o algoritmo com $max = 100.000$, o tempo total foi de 3 minutos e 28 segundos. Durante a execução do algoritmo, e em uma outra execução, com $max = 200.000$, foi calculada a razão entre o número de operações realizadas no segundo laço para cada i e i. Essa razão ficou muito próxima ao número 0.41 para todos os números do intervalo, exceto os 20 primeiros (que no entanto, a menor razão foi 0.33 e a maior 0.5). Da mesma forma, foi calculada a razão entre o tempo de execução do segundo laço e i. O resultado também foi próximo de uma constante, a saber, 0.08. Em cima dessa constante, multiplicada a $\frac{n^2+n}{2}$, estimou-se que, para i variando de 1 a 400.000.000, o algoritmo levaria cerca de 202 anos para terminar. A Tabela 1 mostra os tempos que o algoritmo levou para executar até chegar nas soluções (com $max = 100.000$) e a estimativa dos tempos que o algoritmo levaria até encontrar todas as soluções.

Tabela 1: Resultados dos tempos para as soluções encontradas no Algoritmo1 (* estimativa)

Número da confeitaria	Tempo
1	0
6	0,001ms
35	0,050ms
204	1,736ms
1.189	58,726ms
6.930	33s
40.391	1,12min
235.416	36min*
1.372.105	21h*
7.997.214	29 dias*
46.611.179	2,75 anos*
271.669.860	93,6 anos*

Os tempos de execução dos outros dois algoritmos também foram coletados. A Tabela 2 apresenta um comparativo dos tempos de execução totais dos três algoritmos.

Tabela 2: Resultado do tempo de execução dos 3 algoritmos propostos (* estimativa)

Algoritmo	Tempo de execução
1	202 anos*
2	52s
3	2ms

[3]Os algoritmos foram implementados em C++, compilados com o gcc e executados em uma máquina com processador AMD Athlon 64×2 Dual Core 4200+ 2.21GHz e 1GB de RAM. O tempo dentro dos laços foi coletado em microssegundos, utilizando-se a função *gettimeofday()* da biblioteca *sys/time.h*.

IV Considerações Finais

Este trabalho apresentou três algoritmos que solucionam o problema da "Rua Encantada", apresentando o raciocínio empregado para a construção e considerações a respeito da complexidade de cada um. Resumindo, encontrou-se três algoritmos com diferentes comportamentos: o primeiro é quadrático ($O(n^2)$), enquanto os outros dois são lineares ($O(n)$). No entanto, como discutido na Subseção B, enquanto o segundo algoritmo executa o laço exatamente o número de vezes de sua entrada, o terceiro executa apenas dentro do espaço de soluções possíveis para a entrada determinada. No caso do número máximo de casas da rua estabelecido no problema (400.000.000), o Algoritmo 3 executou apenas 13 iterações (das 12 soluções mais 1, antes do teste do laço). Isso impactou no tempo de execução, como mostrado na Tabela 2 da seção anterior.

Pela experiência do Algoritmo 1, observa-se que nem sempre o algoritmo mais imediato é viável na prática. A análise mais aprofundada do problema, como mostrada na Subseção B, de fato pode levar à soluções mais eficientes ou mesmo viabilizar a busca de soluções para determinados tamanhos de entrada, como visto nesse trabalho.

Referências

[1] Polygonal numbers. Disponível em `http://en.wikipedia.org/wiki/Square_triangular_number`, 2008. Acesso em 21 de setembro de 2008.

[2] Square triangular number. Disponível em `http://mathworld.wolfram.com/SquareTriangularNumber.html`, 2008. Acesso em 21 de setembro de 2008.

[3] Square triangular numbers. Disponível em `http://en.wikipedia.org/wiki/Square_triangular_number`, 2008. Acesso em 21 de setembro de 2008.

[4] Triangular numbers. Disponível em `http://en.wikipedia.org/wiki/Square_triangular_number`, 2008. Acesso em 21 de setembro de 2008.