MO417 – Complexidade de Algoritmos

Nathana Facion - RA 191079

21/03/2017

Exercício 1

Questão 1. *Encontre uma expressão para a i-ésima linha linha do seguinte triângulo, que é chamado de triângulo de Pascal, e prove que sua afirmação está correta usando indução. Os lados do triângulo são 1s e cada um dos outros itens tem valor igual à soma dos dois itens imediatamente acima.*

R:

Intuição

Podemos perceber que as somas das primeiras linhas são:
Linha 0: 1
Linha 1: 2
Linha 2: 4
Linha 3: 8
Linha 4: 16
Linha 5 : 32
Linha n : 2^n
Provaremos por indução. Então que :

$$\binom{n}{0} + \binom{n}{1} + ... + \binom{n}{n} = 2^n$$

Base

Base: $2^0 = 1$

Hipótese

Assumiremos que para n : 2^n é verdadeiro.

Passo

Provaremos para n + 1.

$$\binom{n+1}{0} + \binom{n+1}{1} + ... + \binom{n+1}{n+1} = 2^{n+1}$$

Como sabemos que em Pascal a identidade é :

$$\binom{n+1}{k+1} = \binom{n}{k} + \binom{n}{k+1}$$

Poderemos reescrever a nossa prova como :

$$\binom{n+1}{1} + \binom{n+1}{2} + ... + \binom{n+1}{n} =$$

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{1} + \binom{n}{2} ... + \binom{n}{n-1} + \binom{n}{n} =$$

$$\binom{n}{0} + 2\binom{n}{1} + 2\binom{n}{2} ... + 2\binom{n}{n-1} + \binom{n}{n} =$$

Como podemos perceber na somatória acima não incluimos:

$$\binom{n+1}{0} e \binom{n+1}{n+1}$$

Para esses casos o valor é 1 :

$$\binom{n+1}{0} = \binom{n+1}{n+1} = 1$$

eles não foram aplicados a identidade de Pascal portanto os adicionaremos na soma agora.
Então teremos:

$$1 + \binom{n}{0} + 2\binom{n}{1} + 2\binom{n}{2} ... + 2\binom{n}{n-1} + \binom{n}{n} + 1 =$$

Para os seguintes casos o valor é 1 também:

$$\binom{n}{0} = \binom{n}{n} = 1$$

Portanto podemos escrever os valores de

$$\binom{n+1}{0} = \binom{n}{0} e \binom{n+1}{n+1} = \binom{n}{n}$$

dessa forma, pois o resultado final é o mesmo:

$$2\binom{n}{0} + 2\binom{n}{1} + 2\binom{n}{2} ... + 2\binom{n}{n-1} + 2\binom{n}{n} = 2^{n+1}$$

Exercício 2

Questão 2. *Considere a sequência 1, 2, 3, 4, 5, 10, 20, 40, ..., que começa como uma progressão aritmética, mas após os cinco primeiros termos é progressão geométrica. Mostro que todo inteiro positivo pode ser escrito como a soma de números distintos dessa sequência.*

R:

Intuição

Considere que n é o número o qual queremos escrever.
Caso 1: n é um número até 5. Então basta selecionarmos o número correspondente da série que é uma PA : 1, 2, 3, 4, 5 .
Caso 2: n é um número entre 5 e 10. Então fazemos n - 5, se for 9-5 = 4, 8-5 = 3, 7-5 = 2, 6-5=1 .
Caso 3: n é um número entre 10 e 20. Então fazemos n - $2^0 * 10$, será um número entre 1-9, que já foi mostrado que pode ser realizado a soma.
Caso 4: n é um número entre 20 e 40. Então fazemos n - $2^1 * 10$.
Caso 5: n é um número entre 40 e 80. Então fazemos n - $2^2 * 10$.
Caso i: n é um número entre k e 2k . Então fazemos n - $2^x * 10$.

Provaremos por indução que a soma pode ser realizada com diferentes números:

Base:

Para n até 10. Temos sem precisar realizar somas: 1,2,3,4,5,10. Os números 9 = 5 + 4, 8 = 5 + 3, 7 = 5 + 2, 6 = 5 + 1.

Hipótese:

Considerando o uso da indução forte, sabemos que os números menores que *n* podem ser escritos pela soma de diferentes números da PA e PG.

Passo:

Com a intuição acima podemos perceber que a sequência da PG poderia ser escrita como $2^x * 10$, onde o x indicaria a posição na PG, por exemplo elemento0 = $2^0 * 10 = 10$, elemento1 = $2^1 * 10 = 20$, assim por diante.

Seja n um número entre dois números da sequência da PG. Seja $2^k * 10$ e $2^{k+1} * 10$ os dois números da sequência da PG . Assim $2^k * 10$ mais b é o valor de n. Logo: n = $2^k * 10$ + b . Podemos dizer que b = n - $2^k * 10$, b será menor que n, e de acordo com a hipótese de indução números menores do que n podem ser escrito pela forma de PA e PG.

Além disso podemos afirmar que os números da soma não se repetirão pois a hipótese nos afirma isso. Se $2^k * 10$ não for único então existem 2 números desses o que implicaria em $2 * 2^k * 10$, então o número teria de estar entre $2^{k+1} * 10$ e $2^{k+2} * 10$, o que não estaria correto, devido aos intervalos selecionados. Portanto uma contradição.

Exercício 3

Questão 3. *Seja f(x) uma função real contínua e suponha que ela tem uma ou mais raízes entre a,b (uma raiz é um número r E (a,b) com f(r) = 0) .Dado ε > 0, uma ε-aproximação de uma raiz, é um número a E (a,r) tal que a (r ε,r+ε) em que r é uma raiz. O método de aproximação da raiz baseado em busca binária executa uma busca.*

(a)

Intuição

Para identificar o tempo de execução consideremos uma entrada válida. A cada passo iteração do laço o algoritmo tem uma divisão por dois, ou a ou b é dividido por dois. Isso ocorre até que $\varepsilon \geq$(b-a). Logo :

$\varepsilon \geq$(b-a)* $\frac{1}{2} * \frac{1}{2} * \frac{1}{2} * \frac{1}{2}$...

Consideremos que teremos $\frac{1}{2}$ em n vezes:

$\varepsilon \geq$(b-a)* $\frac{1}{2^n} = (b-a) * 2^{-n}$

Complexidade

$\log_\varepsilon \geq \log$(b-a)*$2^{-n}$
$\log_\varepsilon \geq \log$(b-a) + $\log 2^{-n}$
$\log_\varepsilon \geq \log$(b-a) + $\log 2^{-n}$
- $\log 2^{-n} \geq \log$(b-a) - $\log_\varepsilon$
n $\geq$ ($\log$(b-a) / $\log 2$) - ($\log_\varepsilon$ / $\log 2$)
n $\geq \log$(b-a) - $\log_\varepsilon$

Como temos c*($\log$(b-a) - $\log_\varepsilon$)) operações no loop, sendo c uma constante . A complexidade pode ser descrita como: $\theta(\log$(b-a) - $\log_\varepsilon$).

b)
Considerando a condição if (b - a) > ε

Análise ε

Se $\varepsilon < 0$ o algoritmo não funciona, pois como o próprio enunciado diz a raiz estará entre um intervalo de $(r - \varepsilon, r + \varepsilon)$.
Se$\varepsilon = 0$ o algoritmo encontraria exatamente a ráiz, entretanto como a ou b se divide várias vezes podemos ter um ε muito pequeno mas ele nunca ser zero. O que fará com que a condição do enquanto ficasse em um loop infinito, assim o algoritmo nunca finalizaria.
Logo $\varepsilon > 0$ para que conseguimos ter raizes que respeitem o intervalo da raíz e o algoritmo consiga parar.

Análise a e b

Se a = b, então o loop(Enquanto) nunca será executado e teremos (a+b)/2 , que é uma solução que está correta, já que r = a = b.
Se a > b, então o loop(Enquanto) nunca será executado e teremos (a+b)/2 , que é uma solução que pode estar correta ou errada.
Se b > a , a raíz deve estar no intervalo entre a e b, então continuamos o enquanto (b - a) > ε.A cada iteração conseguimos um número mais próximo da raiz.

Análise de f

Considerando a condição if(f(a + b)/2) ≤ 0. Considere que f(a) e f(b) são negativos, então o if acima é válido e a = ((a+b)/2). Entretanto o lado que deveria mover no domínio é o lado de b, pois

a raiz está mais próxima de a. Assim a raiz fica cada vez mais distante e nunca alcançará o valor correto. O mesmo ocorre se considerarmos f(a) e f(b) positivos ou f(a) positivo e f(b) negativo.
Para o algoritmo funcionar temos que f(a) tem que ser negativo e f(b) tem que ser positivo.Pois assim o domínio do intervalo de a $\leq r \leq b$ sempre muda corretamente a ou b. Isso deve-se a falta de verificação de sinal de f(a) e f(b), fazendo que o algoritmo funcione corretamente apenas nessa situação.

Análise raiz

A raiz deve estar entre o intervalo de a e b, caso contrário não será encontrado. Então a $\leq r \leq b$.

Conclusão sobre entradas

Para o algoritmo funcionar temos que f(a) tem que ser negativo e f(b) tem que ser positivo. A raiz r tem que estar no intervalo a $\leq r \leq b$, sendo b $\geq$ a, e $\varepsilon > 0$.

Análise com entradas válidas

Ínicio: Considerando que a entrada seja válida entraremos no loop(Enquanto). Na primeira iteração teremos a ou b reduzido pela metade do seu valor. Ou seja, a variante funciona. Com a redução do intervalo entre a' $\leq r \leq$ b' , a raiz está mais próxima.
Manutenção: A cada iteracão a ou b vão diminuindo pela metade e com isso nosso domínio de intervalo para encontrar a raiz diminui, ou seja, nos aproximamos da raiz. Essa iteração é realizada enquanto não temos o valor de erro de ε tolerável.
Término: Se temos b - a $< \varepsilon$, então o algoritmo termina e retornamos a raiz encontrada.