Matemática

Simulações

Prof. Edson Alves
Faculdade UnB Gama

Simulações em Programação Competitiva

- Muitos problemas matemáticos em programação competitiva consistem em simular exatamente os passos ou critérios descritos no texto
- Neste sentido, a técnica fundamental para a solução de tais problemas é a busca completa (força bruta), sendo que a poda é crucial em problemas mais difíceis
- Vários destes problemas tem solução fechada, na forma de uma expressão geral, a qual pode não ser óbvia
- Nestes casos, se o tamanho da entrada permitir que a solução de busca completa seja aceita dentro do limite de tempo estabelecido, é melhor usar tal solução do que tentar encontrar a solução fechada

Podas em simulações

- A poda pode e deve ser aplicada quando possível
- Nos problemas de simulação, a poda é baseada nas propriedades das expressões que modelam o problema (identidades trigonométricas, propriedades das operações fundamentais, zeros de polinômios, dentre outros)
- Contudo, a poda é um recurso para diminuir o tempo de execução (ou mesmo a complexidade assintótica, em alguns casos), de modo que deve ser considerada apenas quando a solução simples, sem poda, não atingir o limite de tempo estabelecido

Exemplo

Considere o seguinte problema: *dado um inteiro* N *positivo, qual é o maior inteiro* m *tal que* $N \geq m(m+1)/2$*?*

Há três abordagens possíveis para este problema, com diferentes complexidades e características.

Busca completa

Uma possível solução para este problema consiste em testar, um a um, todos os inteiros menores ou iguais a N em busca da resposta, com complexidade $O(N)$

.

```
int solve(int N)
{
    int m = 1;

    for (int i = 2; i <= N; ++i)
        if (i*(i + 1)/2 <= N)
            m = i;

    return m;
}
```

Busca completa

- A implementação apresentada, embora correta para valores pequenos de N, falha no caso geral
- Se $1 \leq N \leq 10^9$, por exemplo, acontecerá um *overflow* na condição do `if`, comprometendo a corretude do resultado
- Além disso, o algoritmo testa vários valores desnecessariamente: se i for maior do que a raiz quadrada de N, a condição do laço sempre será falsa
- Fazendo este ajuste e corrigindo o tipo base para `long long`, a nova solução terá complexidade $O(\sqrt{N})$

Busca completa

```
long long solve2(long long N)
{
    long long m = 1;

    for (long long i = 2; i * i <= N; ++i)
        if (i*(i + 1)/2 <= N)
            i = m;

    return m;
}
```

Embora seja nítido o ganho de performance e de complexidade, ainda é possível melhorar esta complexidade, por meio de uma busca binária.

Busca binária

- O valor de m pode ser determinado por meio de uma busca binária
- Uma vez que a solução se encontra no intervalo $[1, N]$, é possível, a cada etapa, testar o elemento c que ocupa a posição central do intervalo como possível solução
- Caso $c(c+1)/2 > N$, a solução estará no intervalo $[1, c-1]$
- Caso contrário, a resposta deve ser atualizada para c e a busca deve prosseguir no intervalo $[c+1, N]$
- Assim, a solução terá complexidade $O(N \log N)$

```
long long solve3(long long N)
{
    long long m = 1, a = 1, b = N;

    while (a <= b)
    {
        long long c = a + (b - a)/2;

        if (c*(c + 1)/2 <= N)
        {
            m = max(m, c);
            a = c + 1;
        } else
            b = c - 1;
    }

    return m;
}
```

Busca binária

- Esta abordagem, embora não seja a mais eficiente em termos de complexidade, tem uma grande vantagem
- Como a solução utiliza apenas aritmética inteira (a divisão $c(c+1)/2$ resulta sempre em um inteiro), não há possíveis erros devido a precisão
- A solução fechada, em $O(1)$, depende de aritmética de ponto flutuante, o que pode levar ao resultado errado em determinados casos

Solução fechada

- Efetivamente o problema a ser resolvido consiste em determinar o zero positivo do polinômio $p(m) = m^2 + m - 2N$

- Pela Fórmula de Báskara segue que

$$m = \frac{-1 + \sqrt{1 + 8m}}{2}$$

- A solução desejada seria $\lfloor m \rfloor$, o maior inteiro menor ou igual a m

- Como pode ocorrer um erro de precisão numérica, a solução pode ficar errada para menos: isto pode ser testado e corrigido, se necessário

Solução fechada

```
long long solve4(long long N)
{
    long long m = (-1 + sqrt(1 + 8*m))/2;

    return (m + 1)*(m + 2)/2 <= N ? m + 1 : m;
}
```

Problemas

1. OJ
    1. 616 - Coconuts, Revisited
    2. 834 - Continued Fractions
    3. 1225 - Digit Counting
    4. 10346 - Peter's Smokes
    5. 11254 - Consecutive Integers

Referências

HALIM, Steve; **HALIM**, Felix. Competitive Programming 3, Lulu, 2013.