Compreendendo e usando números de ponto flutuante

por Jeff Bezanson

Os números são certamente o tipo de dados mais prevalente em programas de computador. Eles são tão fundamentais que as pessoas não perdem muito tempo falando sobre eles – certamente todo mundo sabe como usar números em seus programas. Bem, uma das coisas maravilhosas sobre programação é que em quase todos os lugares que você olha, você encontra mais do que aparenta.

A maioria dos programadores já ouviu ou observou uma coisa estranha ou outra sobre números de ponto flutuante. Por exemplo, muitas vezes descobrimos que números de ponto flutuante que parecem iguais não satisfazem necessariamente o teste "==" de C. Novos programadores geralmente são ensinados a nunca usar == para números de ponto flutuante por esse motivo. Ocasionalmente nos deparamos com outros casos excepcionais, por exemplo, fórmulas matematicamente sólidas que, quando implementadas usando ponto flutuante, produzem resultados aparentemente aleatórios ou decepcionantemente imprecisos.

O que está acontecendo? Na verdade, há muitas coisas contra-intuitivas sobre números de ponto flutuante, e é útil saber uma ou duas coisas se você quiser usá-los de forma produtiva em seus programas.

Os números de ponto flutuante fornecem uma espécie de ilusão; parecem números "reais", com decimais e possivelmente magnitudes muito grandes ou pequenas. Na realidade, um número de ponto flutuante de 4 bytes, por exemplo, pode conter *menos* valores distintos do que um número inteiro de 4 bytes. A razão para isto é, obviamente, que a representação interna de números de ponto flutuante não é simples. Os bits que representam um número inteiro são interpretados literalmente como um número binário, enquanto os bits em um número de ponto flutuante têm uma interpretação mais complicada. Falarei longamente sobre essa interpretação, mas primeiro gostaria de discutir algumas diferenças conceituais entre inteiros e flutuantes.

I. Precisão vs. Precisão

Exatidão e precisão são dois conceitos de medição que capturam perfeitamente as diferentes propriedades de ints e floats (em qualquer sistema, independentemente da representação de ponto flutuante específica usada). "Precisão" refere-se a quão próxima uma medição está do valor verdadeiro, enquanto "precisão" tem a ver com a quantidade de informação que você tem sobre uma quantidade, com que precisão você a define.

A aritmética inteira goza da propriedade de precisão total: se eu tiver o número inteiro “2”, é exatamente 2, em ponto, e ninguém pode contestar isso. Além disso, se eu adicionar 1 a ele, sei que obterei exatamente 3. Quaisquer que sejam as operações que eu fizer com números inteiros, desde que não haja overflow, sempre obterei um número que corresponda à resposta correta, bit por bit. No entanto, os números inteiros carecem de precisão. Dividir 5 e 4 por 2, por exemplo, resultará em 2. Os números inteiros não conseguem acompanhar a parte fracionária, então a informação de que eu tinha um número um pouco maior que 4 (ou seja, 5) é perdida no processo. Os números inteiros são muito "grossos" para representar essas gradações mais sutis; usá-los é como construir com

tijolos. Se você quiser fazer um cubo, você sabe que os tijolos podem representá-lo perfeitamente, mas uma esfera não ficaria tão bem (naturalmente, se a esfera for significativamente maior que cada tijolo, você poderá chegar perto o suficiente) .

Os números de ponto flutuante são exatamente o oposto dos números inteiros no que diz respeito à exatidão e precisão. Eles têm boa precisão, pois nunca descartam deliberadamente informações que representam seus números. Se você tivesse bits suficientes, poderia reverter qualquer cálculo de FP para obter o número original, da mesma forma que, com bits suficientes, você poderia representar um número inteiro arbitrariamente grande. Mas os números de ponto flutuante têm pouca precisão. Se os ints são como tijolos, então os floats podem ser considerados uma massa boba. Você tem controle suficiente para moldá-los em formas curvas complexas, mas eles deixam a desejar quando se trata de formar um formato de alvo específico. Imagine tentar fazer um cubo perfeito com uma massa boba - você nunca deixará esses cantos tão afiados quanto deveriam. Na verdade, em muitos casos não há literalmente *nenhuma esperança* de que uma resposta de ponto flutuante corresponda à resposta correta bit por bit.

Agora, como posso fazer uma afirmação estranha como essa? Os números de ponto flutuante são inerentemente diferentes dos números inteiros, pois nem todas as frações podem ser representadas exatamente em binário, mas qualquer número inteiro pode. A aritmética binária não tem culpa; a mesma restrição se aplica a qualquer sistema básico. Considere o número 1/3. Nenhuma representação de dígitos decimais finitos (por exemplo, 0,333333) pode ser igual a 1/3; nunca teremos 3 suficientes. Portanto, é mais do que provável que o resultado calculado de que você precisa *não possa* ser representado com precisão por uma variável finita de ponto flutuante – você estará errado pelo menos um pouco, não importa o que faça. Sabemos que os carros alegóricos ainda são úteis; só temos que evitar que esses pequenos erros de arredondamento nos atrapalhem. Para esse fim, e também para edificação geral, é bom saber como os carros alegóricos realmente funcionam. Próximo: Representação de Ponto Flutuante