Ayudantía 2 CC 2023-2

Javier Pérez

August 20, 2023

1 Perdida de importancia

- Considerando una representación de punto flotante de 8 bits, con uno para el signo, 3 para el exponente y 4 bits para la mantisa. El shift es definido como $2^{3-1} - 1$

  1. calcule ϵ_{mach} para esta representación
     considerando la cantidad de bits disponibles para la mantisa, el ϵ_{mach} será $= 2^{-4}$
  2. Considerando representación subnormal. ¿Cuál es el número más pequeño que se puede representar?
     siguiendo las reglas de representación subnormal tendremos que será el mínimo exponente, además de tener un 0 en el inicio de la representación, y un 1 en el final de la mantisa, quedando así $2^{-4} \cdot 2^{-2} = 2^{-6}$
  3. El número 0.9 en binario es $(.1\widehat{1100})_2$ determine $fl(0.9)$ y verifique si el error relativo de redondeo es menor que $\frac{\epsilon_{mach}}{2}$.
     tendremos que normalizadamente el 0.9 se escribirá como $2^{-1} \cdot 1.110011001100...$ pero tenemos solo cuatro bits de mantisa, por lo que tendremos que seguir las reglas de redondeo, teniendo así $1.1100|1100... \cdot 2^{-1}$ ahora viendo esto podemos determinar que el cuarto bit sera 1, por las reglas de redondeo mencionadas anteriormente.
     Ahora debemos calcular a que equivale este valor, teniendo

     $$2^{-1} + 2^{-2} + 2^{-3} + 2^{-5} = \frac{29}{32} = 0.90625$$

     teniendo esto usamos la formula del error relativo, quedando

     $$\frac{|0.9 - 0.90625|}{|0.9|} = \frac{0.00625}{0.9} = \frac{1}{144} \leq \frac{1}{32} = 2^{-5} = \frac{1}{2} \cdot \epsilon_{mach}$$

     Por lo que vemos que se cumple que el error relativo de redondeo es menor que $\frac{\epsilon_{mach}}{2}$.

4. dado los algoritmos d_1 y d_2, calcule el **out** de cada uno y explique si existe alguna diferencia entre ellos, considerando a = 1/2, b = 1/16 y $\sqrt{1+2^{-4}} \approx 1+2^{-4}$
   primero anotemos que hace cada algoritmo.
   para el algoritmo numero uno tendremos que calcula lo siguiente:

$$\sqrt{a^2+b^2}$$

   para el segundo algoritmo tendremos que se calcula lo siguiente (asumiendo a como el numero mas grande entre los dos):

$$a\sqrt{\frac{b^2}{a^2}+1}$$

   notemos que ambos algoritmos hacen lo mismo, solo que uno factoriza por a^2, mientras que el otro no.

   teniendo esto en cuenta desarrollemos cada algoritmo, revisando que almacena cada uno.
   para d_1:
   - $l_1 = 1/2 \cdot 1/2 = 1/4$
     podemos notar que podemos almacenar perfectamente 1/4 en la notación punto flotante que tenemos, quedando así: $+1.0000 \cdot 2^{-2}$
   - $l_2 = 1/16 \cdot 1/16 = 1/256 = 2^{-8}$
     dado que el numero mas pequeño representable es el 2^{-6} este numero quedara almacenado como un 0.
   - $l_3 = 1/4 + 0 = 1/4$
     como ya vimos este numero si se podía almacenar sin ningún problema
   - **out** $= \sqrt{1/4} = 1/2$
     tampoco nos genera ningún problema almacenar 1/2, quedando $+1.0000 \cdot 2^{-1}$

   para d_2:
   - $M = 1/2$
     no había problemas almacenando 1/2
   - $m = 1/16$
     no hay problemas almacenando 1/16 siempre que usemos notacvion subnormal, quedando $+0.0100 \cdot 2^{-2}$.
   - $l_3 = 2/16 = 1/8$
     al igual que antes usamos la notación subnormal $+0.1000 \cdot 2^{-2}$.
   - $l_4 = 1/8 \cdot 1/8 = 1/64 = 2^{-6}$
     este es el numero mas pequeño que podemos representar, quedando $+0.0001 \cdot 2^{-2}$

- $l_5 = 1 + 2^{-6}$
  acá tendremos un problema, dado que al 1 sumarle un numero que es mucho mas pequeño (considerando la precisión que tenemos), el 2^{-6} no se considerará, quedando así solo el 1, aca la suma especifica: $1.000000 + 0.000001 = 1.000001 \rightarrow 1.0000|01$, como vemos el redondeo hará que el 2^{-6} desaparezca
- $l_6 = \sqrt{1} = 1$
  almacenar el uno tampoco supone ningún problema, quedando así: $+1.0000 \cdot 2^0$
- **out** $= 1/2 \cdot 1 = 1/2 = 2^{-1}$

como vemos, ambos algoritmos dan lo mismo, a pesar de ser un proceso distinto, esto se debe a que ambos tienen problemas con la perdida de significancia, el primero al almacenar un numero demasiado pequeño y el otro al intentar sumar números de muy distintas magnitudes para la representación, perdiendo así precisión.

Algorithm 1 d_1

1:	$l_1 = a \times a$	▷ =.....
2:	$l_2 = b \times b$	▷ =.....
3:	$l_3 = l_1 + l_2$	▷ =.....
4:	**out** $= \sqrt{l_3}$	▷ =.....

Algorithm 2 d_2

1:	$M = max(a, b)$	▷ =.....
2:	$m = min(a, b)$	▷ =.....
3:	$l_3 = m/M$	▷ =.....
4:	$l_4 = l_3 \times l_3$	▷ =.....
5:	$l_5 = 1 + l_4$	▷ =.....
6:	$l_6 = \sqrt{l_5}$	▷ =.....
7:	**out** $= M \times l_6$	▷ =.....

- En la próxima pelicula de los Avengers (Avengers: Age of Ultron), Tony Stark no ha podido perfeccionar su nueva armadura llamada Hulkbuster, la cual la necesita para enfrentar a Hulk. El único problema que le queda por resolver es la siguiente ecuacón de 4to grado: $x^4 - ax^2 - m = 0$, donde a, m > 0. Para la cual necesita de su ayuda.

  1. Implemente un algoritmo que calcule las 2 raíces reales de la ecuación de 4to grado y que no sufra de pérdida de significancia.
  2. Obtenga las raíces reales cuando a = 10 y m = 1.
  3. Obtenga las raíces reales cuando a = 10^8 y m = 10^{-4}. ¿Es 0 una raíz?.

  Este ejercicio queda propuesto al igual que en la guía anterior.