Reusing computations

Jordi Cortadella
Department of Computer Science

• There are certain algorithms that require a repeated sequence of similar computations. For example:

\[ S(x) = \sum_{i=0}^{k} t_i(x) \]

• Strategy: try to reuse the work done from previous computations as much as possible.

Calculating \( \pi \)

\( \pi \) can be calculated using the following series:

\[ \frac{\pi}{2} = \sum_{n=0}^{\infty} \frac{n!}{(2n + 1)!!} \]

where \( n!! = 1 \times 3 \times 5 \times 7 \times \cdots \times n \) (n odd)

\[ \frac{\pi}{2} = \frac{0!}{1!!} + \frac{1!}{3!!} + \frac{2!}{5!!} + \frac{3!}{7!!} + \cdots \]

Since an infinite sum cannot be computed, it may often be sufficient to compute an approximation with a finite number of terms.
Calculating $\pi$: naïve approach

// Pre: nterms > 0
// Returns an estimation of $\pi$ using nterms terms
// of the series.

double Pi(int nterms) {
    double sum = 1;    // Approximation of $\pi/2$
    double term = 1;  // Current term of the sum

    // Inv: sum is an approximation of $\pi/2$ with k terms,
    //      term is the k-th term of the series.
    for (int k = 1; k < nterms; ++k) {
        term = factorial(k)/double_factorial(2*k + 1);
        sum = sum + term;
    }
    return 2*sum;
}

Calculating $\pi$: reusing computations

$$
\frac{\pi}{2} = 1 + \frac{1}{1 \cdot 3} + \frac{1 \cdot 2}{1 \cdot 3 \cdot 5} + \frac{1 \cdot 2 \cdot 3}{1 \cdot 3 \cdot 5 \cdot 7} + \cdots
$$

$$
\frac{\pi}{2} = t_0 + t_1 + t_2 + t_3 + \cdots
$$

$$
t_n = \frac{n!}{(2n + 1)!!}
$$

Calculating $\pi$

// Pre: nterms > 0
// Returns an estimation of $\pi$ using nterms terms
// of the series.

double Pi(int nterms) {
    double sum = 1;    // Approximation of $\pi/2$
    double term = 1;  // Current term of the sum

    // Inv: sum is an approximation of $\pi/2$ with k terms,
    //      term is the k-th term of the series.
    for (int k = 1; k < nterms; ++k) {
        term = term*k/(2.0*k + 1.0);
        sum = sum + term;
    }
    return 2*sum;
}

Calculating $\pi$

• $\pi = 3.14159265358979323846264338327950288…$

• The series approximation:

<table>
<thead>
<tr>
<th>nterms</th>
<th>Pi(nterms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.000000</td>
</tr>
<tr>
<td>5</td>
<td>3.098413</td>
</tr>
<tr>
<td>10</td>
<td>3.140578</td>
</tr>
<tr>
<td>15</td>
<td>3.141566</td>
</tr>
<tr>
<td>20</td>
<td>3.141592</td>
</tr>
<tr>
<td>25</td>
<td>3.141593</td>
</tr>
</tbody>
</table>
Taylor and McLaurin series

- Many functions can be approximated by using Taylor or McLaurin series, e.g.:

\[
 f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} x^n = f(0) + f'(0)x + \frac{f''(0)}{2!} x^2 + \frac{f^{(3)}(0)}{3!} x^3 + \ldots + \frac{f^{(n)}(0)}{n!} x^n + \ldots
\]

- Example: \( \sin x \)

\[
 \sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots
\]

- McLaurin series:

\[
 \sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots
\]

- It is a periodic function (period is \( 2\pi \))

\[
 \text{Calculating } \sin x
\]

- Convergence improves as \( x \) gets closer to zero

\[
\text{Calculating } \sin x
\]

- Reducing the computation to the \((-2\pi, 2\pi)\) interval:

\[
k = \left\lfloor \frac{x}{2\pi} \right\rfloor, \quad \sin x = \sin(x - 2k\pi).
\]

- Incremental computation of terms:

\[
t_i = \frac{(-1)^i x^{2i+1}}{(2i + 1)!}, \quad t_{i+1} = \frac{(-1)^{i+1} x^{2i+3}}{(2i + 3)!} = -t_i \cdot \frac{x^2}{(2i + 2)(2i + 3)}
\]

```cpp
#include <cmath>

// Returns an approximation of sin x.
double sin_approx(double x) {
    int k = int(x/(2*M_PI));
    x = x - 2*k*M_PI; // reduce to the (-2\pi, 2\pi) interval
    double term = x;
    double x2 = x*x;
    int d = 1;
    double sum = term;
    while (abs(term) >= 1e-8) {
        term = -term*x2/((d+1)*(d+2));
        sum = sum + term;
        d = d + 2;
    }
    return sum;
}
```
Calculating $\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots1}$

- Naïve method: $2n$ multiplications and 1 division (potential overflow with $n!$)

- Recursion:
  \[
  \binom{n}{0} = \binom{n}{n} = 1 \\
  \binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1} = \frac{n-k+1}{k}\binom{n}{k-1} \\
  = \frac{n}{n-k}\binom{n-1}{k} = \binom{n-1}{k} + \binom{n-1}{k}
  \]

Combinations: recursive solution

// Pre: n ≥ k ≥ 0
// Returns combinations(n k)
int combinations(int n, int k) {
  if (k == 0) return 1;
  return n \* combinations(n - 1, k - 1) / k;
}

Problem: Integer division $c + n/k$ may not be integer

Conclusions

- Try to avoid the brute force to perform complex computations.
- Reuse previous computations whenever possible.
- Use your knowledge about the domain of the problem to minimize the number of operations.