Exercícios de Revisão

Prof. Daniel Weingaertner
Prof. Armando L.N. Delgado

Além dos exercícios abaixo, faça também os exercícios existentes nas notas de aula.

Questão 1

Considere uma função para calcular $d = A \times B \times s$, onde $\{s, d\} \in \mathbb{R}^N$ são dois vetores de tamanho N e $\{A, B\} \in \{\mathbb{R}^N \times \mathbb{R}^N\}$ duas matrizes de tamanho $N \times N$. Considere ainda que esta função é executada muitas vezes, e que todas estruturas cabem na cache do processador. Responda:

(a) Supondo que a ordem das operações não seja relevante neste caso, qual a forma mais eficiente de computar o valor de d ? Justifique sua resposta.
   i. $d = (A \times B) \times s$
   ii. $d = A \times (B \times s)$

(b) Escreva o código que efetue o cálculo de d de acordo com sua opção no item anterior.

```
void updCell(double *s, double *A, double *B, double *d, long SIZE)
{
  double *aux;
  // aloca estrutura aux, seja ela qual for (não precisa alocar)
  ...
  // inicia os cálculos

}
```

Questão 2

Observe o código abaixo que calcula a seguinte integral pelo método de Monte Carlo:

$$\iint_a^b f(x, y)\, dx\, dy, \quad onde \quad f(x, y) = 10^5 x^2 + y^2 - (x^2 + y^2)^2 + 10^{-5}(x^2 + y^2)^4$$

```
double calc_integral_mc(int n, double a, double b){
   double sum, x, y;
   for(int i=0; i < n; i++) {
      x = a + (double) rand() * ( (double) 1.0 / (RAND_MAX * (b – a)) );
      y = a + (double) rand() * ( (double) 1.0 / (RAND_MAX * (b – a)) );
      sum += 1e5 * pow(x, 2) + pow(y, 2) – pow( pow(x,2) + pow(y,2), 2 )
             + 1e-5 * pow(pow(x,2.0) + pow(y,2), 4)
 }
 return (b - a)*(b – a) * sum / n;
}
```

Otimize este código o máximo possível. Destaque **cada** melhoria implementada que aumente a eficiência do seu código, **justificando-a!** A eficiência do código resultante é o principal critério de avaliação. A corretude é atributo indispensável.

Questão 3

Sejam um conjunto de pontos $(x_i, y_i) \in \mathbb{R}^2$, $x_i < x_{i+1}$, $1 \leqslant i \leqslant N$ e $f(x_i) = y_i$ representando uma função a ser utilizada por determinada aplicação. Você foi contratada(o) para implementar um programa que retorne/calcule o valor de $f(z)$, $z \in \mathbb{R}$ para $x_2 < z < x_{N-1}$. Caso o ponto $(z, f(z))$ não esteja definido no conjunto, você deve interpolar a função através de um polinômio de grau 4 (quatro) utilizando os pontos x_i mais próximos de z . Considerando que os pontos não são uniformemente espaçados, responda:

(a) Quantos pontos são necessários para calcular um valor interpolado $f(z)$?

(b) Qual dos métodos de interpolação deve ser utilizado: Newton ou Newton-Gregory? **Justifique!**

(c) Qual o problema em se utilizar um único polinômio interpolador definido a partir de todos os pontos, quando o número de pontos é muito grande?

(d) Considerando uma implementação eficiente do programa definido no enunciado, qual será o maior custo computacional: acesso à memória ou uso de CPU? Justifique.

Questão 4

a) Por que o código abaixo é ineficiente? **Justifique!**
b) Reescreva-o de forma a sanar o problema.

```
...
/* n é muito grande */
for(i=0; i<n; i++)  {
   if( x == A ) {
       FuncaoA(i);
   }
   else if( x == B ) {
       FuncaoB(i);
   }
   else {
       FuncaoC(i);
   }
}
```

Questão 5

Para cada par de códigos abaixo, indique qual das versões de código é mais rápida e por que?

(a)

Versão A	Versão B

`int p[SIZE];`
`for (long x=0; x<NUM_COL; ++x) {`
`   for (long y=0; y<NUM_LIN; ++y) {`
`      p[x+y*NUM_COL]++`
`   }`
`}`
	`int p[SIZE];`
	`for (long y=0; y<NUM_LIN; ++y) {`
	`   for (long x=0; x<NUM_COL; ++x) {`
	`      p[x+y*NUM_COL]++`
	`   }`
	`}`

(b)

Versão A	Versão B

`...`
`for (i=0; i<N; ++i) {`
`   if (p==1)`
`      norm += fabs(x[i]);`
`   else if (p==2)`
`      norm += x[i] * x[i];`
`   else`
`      norm += pow(x[i],p);`
`}`
	`...`
	`if (p==1)`
	`   for (i=0; i<N; ++i)`
	`      norm += fabs(x[i]);`
	`else if (p==2)`
	`   for (i=0; i<N; ++i)`
	`      norm += x[i] * x[i];`
	`else`
	`   for (i=0; i<N; ++i)`
	`      norm += pow(x[i],p);`

Questão 6

Sejam um conjunto de pontos $(x_i, y_i) \in \mathbb{R}^2,\ x_i < x_{i+1},\ 1 \le i \le N$ e $f(x_i) = y_i$ uma função utilizada por determinada aplicação. Você foi contratada(o) para implementar um programa que retorne/calcule o valor de $f(z), z \in \mathbb{R}$ para $x_2 \le z \le x_{N-1}$. Caso o ponto $(z, f(z))$ não esteja definido no conjunto, você deve interpolar a função através de um polinômio de grau 3 (três) utilizando os pontos x_i mais próximos de z. Responda:

Lagrange: $p_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$ e $L_i(x) = \prod_{j=0, j \ne i}^{n} \frac{(x - x_j)}{(x_i - x_j)}$

Newton: $p_n(x) = d_0 + d_1(x - x_0) + d_2(x - x_0)(x - x_1) + \cdots + d_n(x - x_0)\cdots(x - x_{n-1})$, onde $d_k, k = 0, 1, \ldots, n$ são as diferenças divididas de ordem k.

Newton-Gregory: $p_n(x) = d_0 + \frac{d_1}{h}(x - x_0) + \frac{d_2}{2h^2}(x - x_0)(x - x_1) + \cdots + \frac{d_n}{n!\,h^n}(x - x_0)\cdots(x - x_{n-1})$,

onde $d_k, k = 0, 1, \ldots, n$ são as diferenças ordinárias de ordem k e $h = x_{i+1} - x_i, i = 0, 1, \ldots, n$.

a) Quantos pontos são necessários para calcular um valor interpolado $f(z)$?

b) Qual dos métodos de interpolação pode ser utilizado: Newton, Laplace ou Newton-Gregory? **Justifique!**

c) Qual das estruturas de dados abaixo seria mais eficiente para armazenar o conjunto de pontos caso você utilizasse o polinômio interpolador de Lagrange? E se você usasse Newton? **Justifique!**

Estrutura A

```
struct Ponto {
   double x,y;
}

struct Ponto p[MAXPTOS];
```

Estrutura B

```
struct Pontos {
   double x[MAXPTOS];
   double y[MAXPTOS];
}

struct Pontos p;
```

Questão 7

a) Reescreva o código abaixo de forma a melhorar seu desempenho. Por que sua versão do código é mais eficiente? **Justifique!**

```
double a[n],x[n],y[n],b[n],z[n];
...
for (i=0; i<n; i++){
  a[i]=x[i]+y[i]*sin((i%8)*M_PI);
}
for (i=0; i<n; i++){
  b[i]=1.0/x[i]+z[i];
}
```

b) Considerando que a instrução “`pragma unroll (8)`” desenrola o laço 8 vezes, por que motivo o **Código A** tem um desempenho <u>pior</u> do que o **Código B**?

Código A

```
#pragma unroll (8)
for (i=0; i<n; i++)
{
   a[i] = b[i]+c[i]*d[i];
   e[i] = f[i]-g[i]*h[i]+p[i];
   q[i] = r[i]+s[i];
}
```

Código B

```
#pragma unroll (8)
for (i=0; i<n; i++)
   a[i] = b[i]+c[i]*d[i];

#pragma unroll (8)
for (i=0; i<n; i++)
   e[i] = f[i]-g[i]*h[i]+p[i];

#pragma unroll (8)
 for (i=0; i<n; i++)
    q[i] = r[i]+s[i];
```

Questão 8

Analise o código apresentado abaixo.

```
double a[nd1][nd2], y[nd2], x[nd1]
...
for (long i=0; i < nd2; ++i)
{
  double t = 0.0;
  for (long j=0; j < nd1; ++j)
    t = t + a[j][i]*x[j];
  y[i] = t;
}
```

(a) Descreva dois **problemas** na implementação do código acima que o tornam ineficiente

(b) Reescreva o código de forma a sanar/diminuir estas deficiências:

(c) Explique por que sua versão melhora cada um dos problemas apresentados no item (a)

Questão 9

Seja uma função $f:\mathbb{R}^2 \rightarrow \mathbb{R}$. Escreva um programa em linguagem C que calcule a integral $\iint_a^b f(x,y)\,dx\,dy$ utilizando o Método dos Retângulos. O número de pontos n inicial (para cada dimensão) é dado, e o espaçamento entre os pontos h é igual em ambas dimensões e dado por $h=b-a/n \Rightarrow \{x_i, y_i\}=a+hi$. O programa deve executar diversas iterações, reduzindo o valor do intervalo ao meio a cada iteração, até que o Erro Aproximado Absoluto da integral seja menor do que ϵ dado.

Método dos Retângulos: $\int_b^a f(x)\,dx \approx \int_b^a p_0(x)\,dx = h\left(\sum_{i=0}^{n-1} f(x_i)\right)$

```
double f (double x, double y);
...
double integral (double a, double b, double epsilon, uint n)
{

}
```

(a) O Método dos Retângulos é apropriado para calcular a integral de funções de alta dimensionalidade? Justifique.

Questão 10

Observe o código para o método de Jacobi em duas dimensões apresentado abaixo.

```
double phi[iMAX][jMAX][2];
int t0=0, t1=1;
...
for (long it=1; it<ITER; ++it) {
   for (long i=0; i < iMAX; ++i) {
      for (long j=0; j < jMAX; ++j) {
         phi[i][j][t1] = 1.0/4.0 * (phi[i-1][j][t0] + phi[i+1][j][t0] +
                                    phi[i][j-1][t0] + phi[i][j+1][t0]);
      }
   }
   aux = t0; t0 = t1; t1 = aux; /* troca os vetores */
}
```

Reimplemente este código utilizando a técnica de "loop blocking" e **Justifique** por que o "loop blocking" torna este código mais eficiente!

Questão 11

A versão A do código abaixo demora o dobro do tempo para executar do que a versão B. Por que isso ocorre?

Versão A

```
struct DATA {
   int a, b, c, d;
};
DATA p[N];

for (long i=0; i<N; ++i) {
   p[i].a = p[i].b
}
```

Versão B

```
struct DATA {
   int a, b;
};
DATA p[N];

for (long i=0; i<N; ++i) {
   p[i].a = p[i].b
}
```

Questão 12

```
for (int i=0; i<N; ++i)
   for (int j=0; j<N; ++j)
      c[i] = c[i] + A[i][j] * b[j]
```

Considere o código acima:

a) Reimplemente este código aplicando apropriadamente a técnica de "loop unroll" com tamanho quatro.

b) O código com o laço desenrolado é mais eficiente que o código original em uma arquitetura x64? Justifique sua resposta.

Questão 13

Responda às seguintes questões:

a) Qual o problema de se utilizar muitos pontos para calcular o polinômio interpolador de uma função tabulada? Como proceder para calcular um valor interpolado a partir de um grande conjunto de pontos?
b) Porque o acesso em coluna é ineficiente para matrizes bidimensionais em linguagem C?
c) Por que a integração numérica pelo método dos trapézios não é uma boa solução para problemas de alta dimensionalidade?

Questão 14

Dado um conjunto de **n** pontos **(xi , yi)**, escreva um procedimento para calcular a tabela completa de diferenças divididas de Newton. Faça também um procedimento que recebe o valor do grau **p** de um polinômio interpolador entre dois pontos **(xk , yk)** e **(xj , yj) , p < n**, e calcula os coeficientes do polinômio interpolador de Newton de grau **p**, usando a tabela de diferenças divididas gerada pelo primeiro procedimento. Finalmente faça um procedimento que calcule o valor do polinômio interpolador para um valor qualquer.