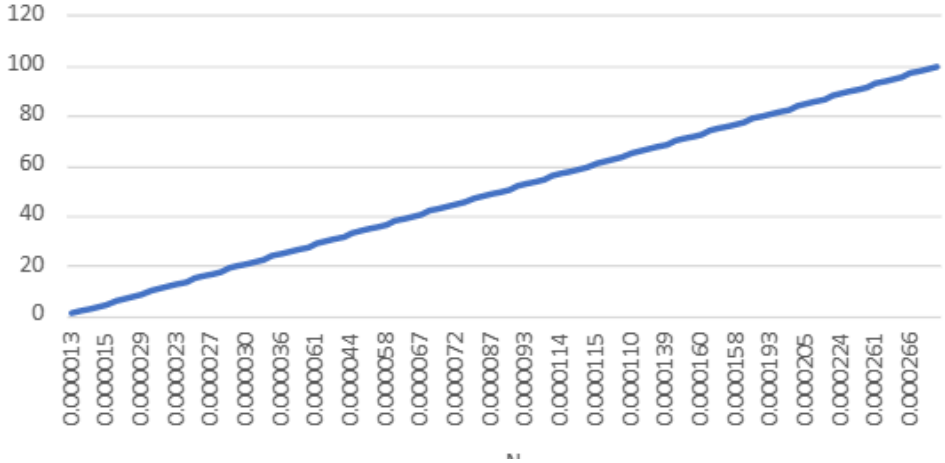
PAA Trabalho 1

Marcos Ani Cury Vinagre Silva – 684903

Relatório

a) Projeto desenvolvido sem problemas
b) Houve a necessidade de realizar algumas mudanças na questão do input pois a plataforma online utilizada para o desenvolvimento não permitia enviar valores pela chamada do executavel
c) O resultado com uma matriz 3x3 foi de 0.000311 segundos
d) A função tem uma complexidade de n de acordo com a imagem abaixo é uma reta



ResultValues.xlsx

e) Soluções propostas:

a.
```
for (i = 0; i < cols; i++)
  for (j = 0; j < rows; j++) {
    element = *(M + j * cols + i);
    DetOutput(element, C, &out_even, &out_odd);
    if ((element % 2) == 0)
      *(Q + j * cols + i) = pow(out_even, 2);
    else
      *(Q + j * cols + i) = out_odd;
  }
```
No loop acima os valores out_odd e out_even só são utilizados dependendo do resultado de element % 2, por conta disso o calculo das funções podem ser feitos somente se necessário evitando assim a execução de funções matemáticas complexas em todos os loops.

```
/*****************************************************************
void DetOutput(unsigned char element, float *C, float *out_even,
               float *out_odd) {
  float sin_element, cos_element;
  DetSinCos(element, &sin_element, &cos_element);
  *out_even = 2.0 * sin_element + exp(cos_element);
  *out_odd = exp(sin_element) * C[element];
}
/*****************************************************************
```
essa proposta não aumenta a utilização da memória pois a única diferença vai ser aonde

será feito a condicional. Após a alteração ficou assim:

```
 for (i = 0; i < cols; i++)
   for (j = 0; j < rows; j++) {
     element = *(M + j * cols + i);
     *(Q + j * cols + i) = DetOutput(element, C, &out_even, &out_odd);
   }
```

```
/*****************************************************************************/
float DetOutput(unsigned char element, float *C, float *out_even,
                float *out_odd) {
  float sin_element, cos_element;
  DetSinCos(element, &sin_element, &cos_element);
  if ((element % 2) == 0) {
    *out_even = 2.0 * sin_element + exp(cos_element);
    return pow(*out_even, 2);
  }
  *out_odd = exp(sin_element) * C[element];
  return *out_odd;
}
/*****************************************************************************/
```

com 100 instâncias o speedup é de 0,000297/0,000262 = 1,13 ou 13%.

b.

```
for (i = 0; i < cols; i++)
  for (j = 0; j < rows; j++) {
    element = *(M + j * cols + i);
    DetOutput(element, C, &out_even, &out_odd);
    if ((element % 2) == 0)
      *(Q + j * cols + i) = pow(out_even, 2);
    else
      *(Q + j * cols + i) = out_odd;
  }
```

```
for (i = 0; i < 256; i++)
 C[i] = (C[i] > 0) ? log(C[i]) : 0.0;
```

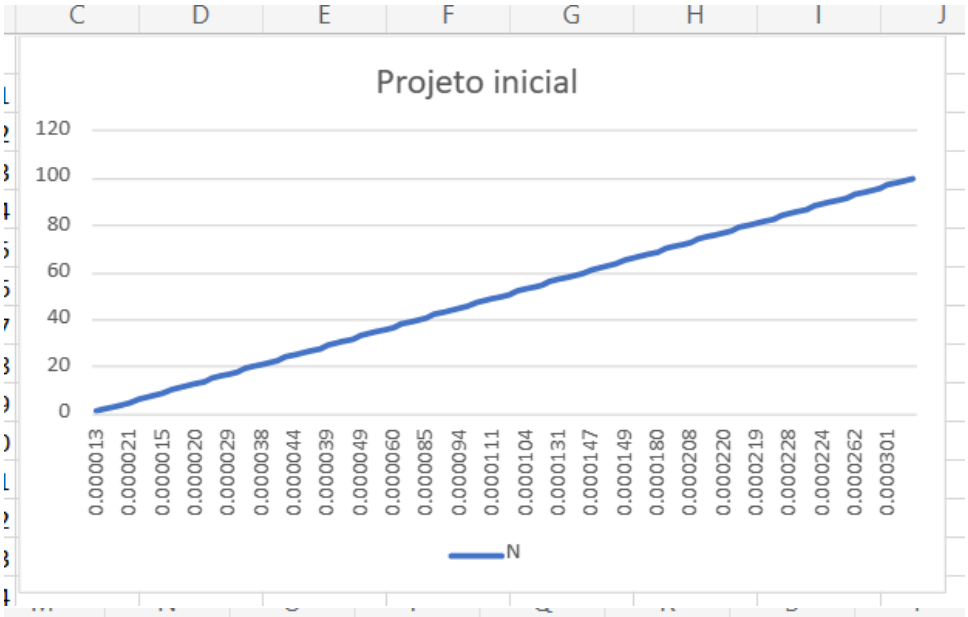
```
void DetOutput(unsigned char element, float *C, float *out_even,
               float *out_odd) {
  float sin_element, cos_element;
  DetSinCos(element, &sin_element, &cos_element);
  *out_even = 2.0 * sin_element + exp(cos_element);
  *out_odd = exp(sin_element) * C[element];
}
```

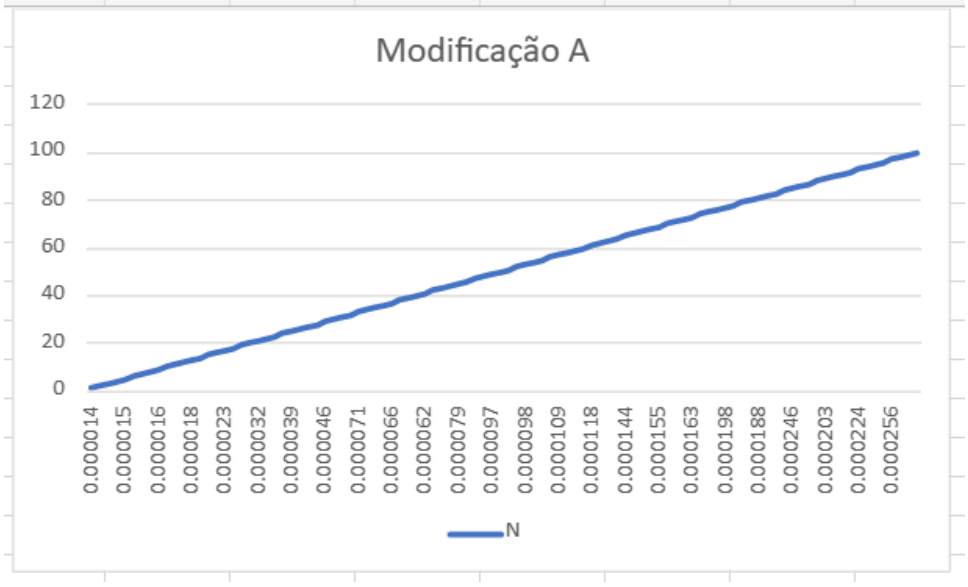
Na imagem a cima observamos que é passado o C como parâmetro e apenas os valores C[element] são utilizados, porém temos um loop de 256 iterações que atribuem ao C seu próprio log baseado no número de valores que possuem na matriz, para evitar esse loop podemos passar por parâmetro para a função DetOutput diretamente o resultado da operação log. A proposta não aumenta a utilização da memória. Após a alteração ficou assim:

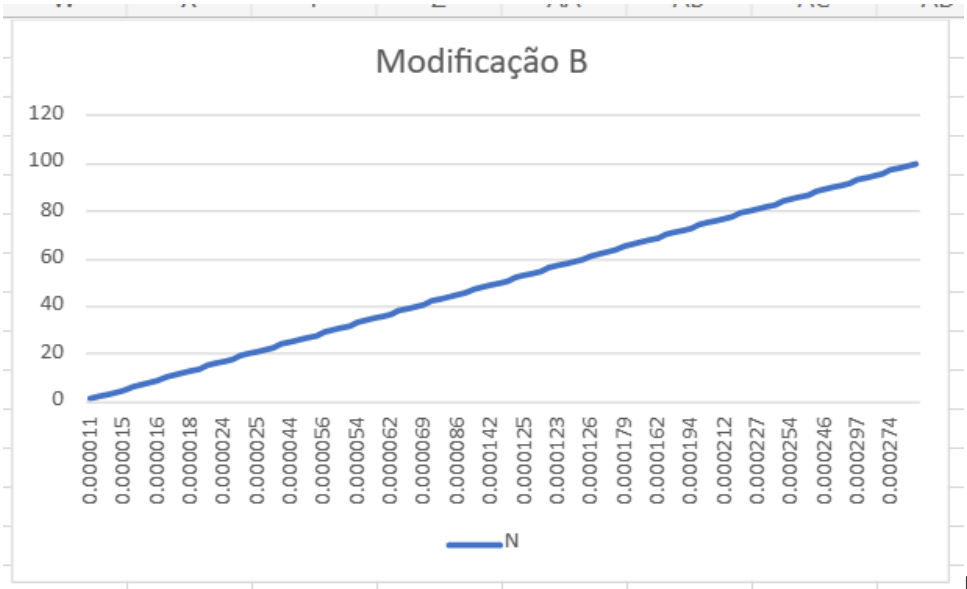
```
for (i = 0; i < cols; i++)
  for (j = 0; j < rows; j++) {
    element = *(M + j * cols + i);
    DetOutput(element, (C[element] > 0) ? log(C[element]) : 0.0, &out_even,
              &out_odd);
    if ((element % 2) == 0)
      *(Q + j * cols + i) = pow(out_even, 2);
    else
      *(Q + j * cols + i) = out_odd;
  }
```

removendo assim o loop de 256 iterações. Com 90 instâncias o speedup é de 0,000282/0,000269 = 1,04 ou 4%.

f)







não se manteve a mesma complexidade

g) A solução A eu considero a melhor custo benefício, pois não aumentou o consumo de memória e aumentou em 13% a velocidade de execução

h) Foi utilizado para teste o replit e todos os códigos foram desenvolvidos nele, caso queira rodar os códigos no mesmo ambiente segue o link para o teste: https://replit.com/join/przprvgauc-marcosanicury