Desafio de Programação Paralela

ERAD-SP 2017

Regras

Leia atentamente todos os problemas apresentados. Cada problema possui uma breve descrição, e exemplos de entradas e saídas (que podem estar nos anexos). Você pode criar sua própria versão paralela e/ou distribuída para o problema ou modificar uma implementação disponibilizada pelos juízes de acordo com sua estratégia, a não ser que o enunciado do problema diga o contrário.

Cada equipe deve criar um arquivo *Makefile* (veja os exemplos nos anexos) com as instruções de compilação. O arquivo binário principal gerado ou um script de execução para (e.g. MPI) deve ser igual a letra que representa o enunciado do problema, em maiúsculo: por exemplo, se o problema chama-se ***A***, o executável/script deve-se chamar ***A***.

Para a submissão, cada equipe deve compactar o código fonte e o *Makefile* em um arquivo .zip (obrigatoriamente) e enviá-lo no sistema BOCA. Esse arquivo pode ter, no máximo, 32 Kb de tamanho. Submissões maiores do que este tamanho serão desconsideradas.

O tempo de execução de um problema será medido utilizando a ferramenta *time* no ambiente de testes dos juízes. Será considerada apenas o tempo real de CPU (*real CPU*). Cada submissão será executada três vezes com as mesmas entradas e o tempo médio será utilizado como a métrica de tempo de execução. O tempo de execução das soluções sequenciais dos problemas, produzido pelos juízes, também será medido da mesma forma. A pontuação se dará pelo *speedup*, ou seja, a razão entre o tempo de execução da solução sequencial pelo tempo de execução da submissão. As equipes ganharão pontos em cada problema conforme o valor do *speedup*, sendo este acrescentado no placar.

O time que tiver a maior pontuação será considerada a campeã do Desafio!

Problema A

A fórmula BBP do π

Ah, o π! Não tem como fugir desse lindo número cabalístico! ☺

Desde a antiguidade, seja no Egito, na China, na Babilônia, esse número vem sendo estudado pelos principais matemáticos.

Atualmente, existem diversas formas de se encontrar esse número. E uma dessas maneiras é a partir da fórmula *Bailey–Borwein–Plouffe*:

$$\pi = \sum_{k=0}^{n} \left[\frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right) \right]$$

Aplicando-se algumas propriedades conhecidas de somatório, é possível chegar na seguinte fórmula:

$$\pi = 4\sum_{k=0}^{n} \frac{1}{(16^k)(8k+1)} - 2\sum_{k=0}^{n} \frac{1}{(16^k)(8k+4)} - \sum_{k=0}^{n} \frac{1}{(16^k)(8k+5)} - 4\sum_{k=0}^{n} \frac{1}{(16^k)(8k+6)}$$

Modifique o código-fonte disponibilizado para calcular o π utilizando técnicas paralelas e/ou distribuídas.

Entrada

A entrada contém apenas um caso de teste. A primeira linha contém dois valores: o primeiro é o número de dígitos *D* de casas decimais do π; o segundo é o índice final *N* da somatória. Considere ($1 \leq D \leq 10^5$) e ($1 \leq N \leq 10^8$).

A leitura dos dados é feita a partir da entrada padrão.

Saída

A saída contém apenas uma única linha com o valor do π calculado com *D* dígitos de precisão.

A escrita dos dados é feita na saída padrão.

Exemplo

Entrada	Saída respectiva a entrada
`12 7`	`3.141592653587`

Problema B

Distância Euclidiana

A distância euclidiana é utilizada para cálculo de similaridades e agrupamento. É um cálculo simples, mas que, se aplicado em grandes massas de dados, como em *BigData*, a computação se torna realmente intensiva!

Dados dois pontos $P = (p_1, p_2, \dots, p_n)$ e $Q = (q_1, q_2, \dots, q_n)$ num espaço euclidiana n-dimensional, a distância euclidiana entre esses pontos é definida como:

$$d(P, Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots (p_n - q_n)^2}$$

Construa uma solução paralela e/ou distribuída para calcular a distância euclidiana entre um ponto *P* e os pontos de um conjunto *Q*, apresentando o índice do ponto Q_i que teve a menor distante com o ponto *P*.

Por questões de reprodutibilidade, deve ser utilizado o código-fonte do gerador de números pseudoaleatórios disponibilizado.

Por questões práticas, a geração dos pontos deve ser de responsabilidade da solução construída/apresentada, seguindo obrigatoriamente esta ordem:

1. Primeiro, gera-se todos os pontos de um conjunto *Q*, começando pelo ponto Q_0 até o último ponto. Para cada ponto, gera-se aleatoriamente os valores de cada coordenada do ponto em questão, num total de *N* coordenadas por ponto.
2. Em seguida, gera-se o ponto *P* e os valores de suas coordenadas são geradas aleatoriamente.
3. Por fim, encontra-se o Q_i que possui a menor distancia euclidiana com P e imprime o valor de *i*.

Entrada

A entrada contém apenas um caso de teste. A primeira linha informa o valor *N* que define o espaço euclidiano de n-dimensões, e o valor de *Q* que indica a quantidade de pontos do conjunto.

A leitura dos dados é feita a partir da entrada padrão.

Saída

A saída contém apenas uma única linha com o índice i ($1 \leq i \leq Q$) do ponto Q_i que possui a menor distância euclidiana com o ponto P.

A escrita dos dados é feita na saída padrão.

Exemplo

Entrada	Saída respectiva a entrada
`2 2`	`1`