A blue and black logo

Description automatically generated

Instituto Politécnico de Setúbal

Escola Superior de Tecnologia de Setúbal

José Almeida (202100661)

Tiago Silva (202000331)

Licenciatura em Engenharia Informática

Computação Paralela e Distribuída

Contents

[Resultados Obtidos 3](#_Toc165838869)

[Funcionamento 4](#_Toc165838870)

# Resultados Obtidos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  | **1 Core** | **3 Core** | **5 Core** |  |
|  | **5 Segundos** | 32303 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | 32323 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | 31643 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | 32441 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | 33409 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | **Max - 5s** | 33409 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | **Nº de números** | 5 | 20 | 20 |  |
|  | **20 Segundos** | 142327 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | 142151 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | 148817 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | 140069 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | 145687 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | **Max - 20s** | 148817 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | **Nº de números** | 6 | 20 | 20 |  |
|  | **60 Segundos** | 477671 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | 482873 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | 490481 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | 469241 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | 473761 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | **Max - 60s** | 490481 | 805950546409752000000000 | 967140655691703000000000 |  |
|  | **Nº de números** | 6 | 20 | 20 |  |
|  |  |  |  |  |  |

# Funcionamento

**O programa executa a função find\_prime(timeout, num\_processes).**

Na função **find\_prime(timeout, num\_processes)**, são realizadas as seguintes etapas:

* É criado um Manager para gerir a informação partilhada entre processos. É também criado um dicionário "dict()", para que permite que as alterações sejam refletidas em todos os processos.
* O menor valor primo existente (2) é iniciado no dicionário.
* Um objeto de bloqueio é criado para controlar e sincronizar o acesso ao dicionário.
* É definida uma variável **range\_per\_process**, que determina o intervalo máximo de trabalho para cada processo.

Para cada processo passado como parâmetro:

* Calcula-se um número inicial ímpar, uma vez que o único número par é o 2.
* O número máximo do processo é calculado com base no valor de **range\_per\_process**.
* Inicia-se um processo que executará a função **prime\_search()** com os parâmetros definidos anteriormente: número inicial, número máximo do processo, dicionário partilhado e limite de tempo.
* Cada processo criado é adicionado a uma lista de processos.
* Os processos são então iniciados.

**Esta função aguarda o término de todos os processos e retorna o maior valor encontrado.**

Na função **prime\_search(start, end, shared\_dict, timeout):**

* Valida-se se o número inicial é par e, se for, é convertido para ímpar.

Enquanto o número atual for menor que o teto máximo do processo e o tempo estiver dentro do limite:

* É invocado o método miller\_rabin(número atual) para verificar se o número é primo.
* Se o número for primo e maior do que o valor atual armazenado no dicionário, este é substituído.
* O número atual é incrementado por dois para passar ao próximo número ímpar.

Na função **prime\_search(start, end, shared\_dict, timeout):**

* Verifica-se se o número atual é 2 ou 3, retornando verdadeiro se forem, ou falso se forem menores do que ambos ou se forem pares.
* São inicializadas as variáveis "s" e "d". "s" é inicializada a 0 e "d" é inicializada como o número atual menos 1.
* Enquanto "d" não for par, "d" é dividido por 2 e s é incrementado por 1. Este processo descompõe o número atual - 1 em 2^s \* d, onde d é um número par.

O ciclo for realiza k testes:

* A variável "a" recebe um valor aleatório entre 2 e n - 2 (inclusive).
* A variável "x" recebe um valor calculado com a^(d % n), onde d é derivado da decomposição de (n-1) em 2^(s \* d).
* Se "x" for igual a 1 ou "n" - 1, o ciclo continua. Caso contrário, entra num outro ciclo para efetuar verificações adicionais.
* Este ciclo repete-se "s" - 1 vezes, onde "s" representa o número de vezes que "d" foi reduzido pela metade até se tornar ímpar.
  + "x" é elevado ao quadrado e reduzido módulo "n".
  + Se "x" se tornar n - 1, o ciclo é interrompido.
* Se nenhuma das operações anteriores resultar em "n" - 1, então "n" não é primo e é devolvido falso.