Universidade Estadual de Mato Grosso do Sul
Curso de Ciência da Computação
Disciplina de Programação Paralela e Distribuída
Trabalho: Implementa do QuickSort Paralelo
Trabalho Individual
Data de entrega: 27/06/2025

INTRODUÇÃO

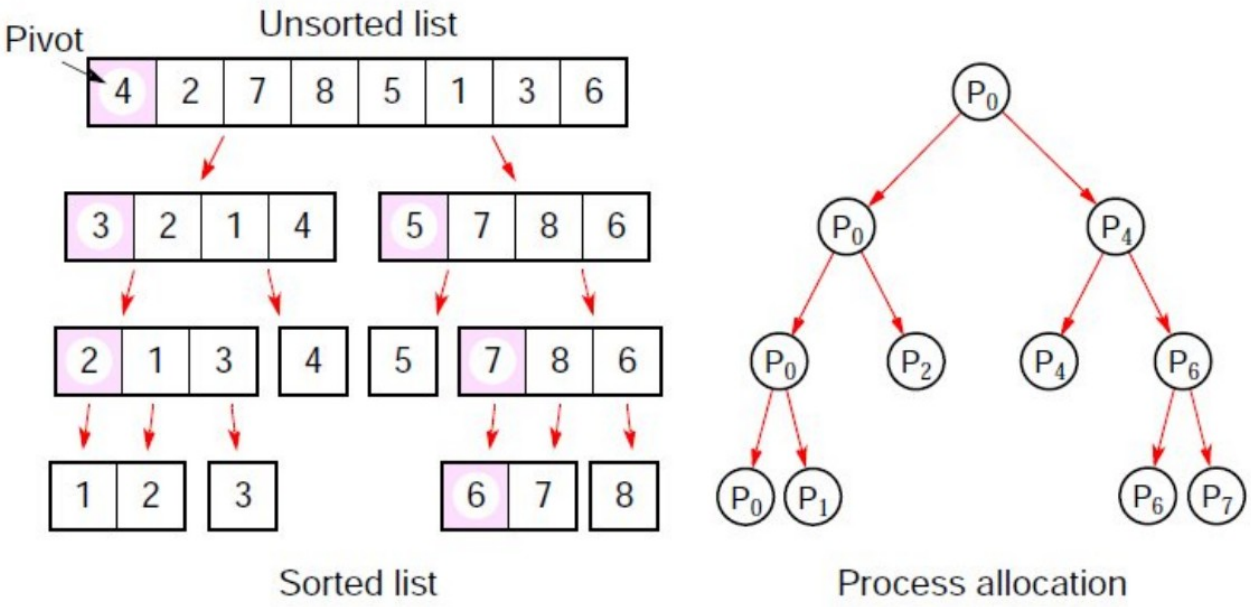
O processo de **ordenação** é um dos problemas mais estudados em Ciência da Computação. Há estudos que mostram que boa parte do tempo de processamento de "grandes" computadores são gastos com ordenação. Desta forma, ordenar continua sendo um problema teórico interessante e desafiador no campo prático.

O algoritmo QuickSort é amplamente aceito como uma das técnicas de ordenação mais eficientes. Esta técnica ordena uma lista de chaves recursivamente por meio da escolha de uma chave **p** como pivot, em torno da qual são rearranjadas as outras chaves da lista. O algoritmo Quicksort possui um desempenho satisfatório em uma variedade de situações com complexidade de tempo de O(N log N). Entretanto, dependendo da situação, pode-se ter o pior caso com custo de $O(N^2)$ dependendo do tipo de ordenação da lista inicial.

QUICKSORT PARALELO

- O setup com os números desordenados estão armazenados em 8 arquivos separados. Vocês precisarão juntar todos em um só. Este arquivo único será lido pela máquina mestre (rank 0).
- A máquina rank 0 aplicará o pivot para dividir a lista de números. O pivot a ser usado será calculado da seguinte forma: Soma(primeiro elemento + elemento central + último elemento)/3;
- Ao fazer a divisão, a máquina rank 0 enviará para duas máquina abaixo (estamos trabalhando com uma árvore) as partes da lista dividida. Ou seja, o nó da esquerda ficará com o elementos menores ou igual ao pivot, e a máquina da direita ficará com os elementos maiores que o pivot. Cada uma dessas duas máquinas fará novamente a divisão de suas listas e expandirá a árvore em mais 4 nós. Ou seja, os processos se dividem em dois grupos e o algoritmo se torna recursivo.
- Após log(P) recursões, cada processo tem uma lista não classificada de valores completamente distintos dos valores mantidos pelos outros processos.
- O maior valor no processo i será menor que o menor valor mantido pelo processo i + 1. Cada processo finalmente classifica sua lista usando a classificação rápida sequencial.

A seguir segue uma figura ilustrativa:



Parâmetros de testes:

1. Testar com o código sequencial;
2. Testar com 2, 4, 6 e 8 máquinas;
3. Gerar o gráfico de tempoXmáquinas. Isto é, um gráfico de colunas mostrando quanto tempo cada conjunto de máquinas gastou.
4. Traçar o gráfico do Speedup e neste gráfico colocar o Speedup Linear.
5. Traçar o gráfico de Eficiência.

- **Código** – O código e a documentação devem ser entregues em um arquivo no formato zip. Dentro deste arquivo zip devem conter um **readme.txt** com o nome do integrante e o comando para a execução do código e, um arquivo **pdf** da documentação. Inclua todos os arquivos fontes (.c, .h, makefile, **não inclua executáveis ou arquivos objetos**).
- **Documentação** – O texto da documentação deve ser breve, de forma que o professor possa entender o que foi feito no código sem ter que entender linha a linha dos arquivos. Implementações modularizadas deverão mencionar quais funções são implementadas em cada módulo ou classe. A documentação deverá conter os seguintes itens:
  - O sumário do problema a ser tratado;
  - Uma descrição sucinta do algoritmo e dos tipos abstratos de dados, das principais funções, procedimentos e das decisões de implementação;
  - Decisões de implementação que porventura estejam omissos na especificação;
  - Como foi tratada a transmissão das mensagens entre os nós (ponto a ponto ou coletivo);
  - Testes, mostrando que o programa está funcionando de acordo com as especificações, seguidos de sua análise;
  - Print screens mostrando o correto funcionamento do programa e exemplos de testes executados;
  - Conclusão e referências bibliográficas.

Avaliação:

A avaliação do trabalho será composta pela execução do programa desenvolvido e pela análise da documentação.

Os seguintes itens serão avaliados:

- A qualidade do código (código bem organizado, estruturado, com comentários explicativos, variáveis com nomes intuitivos, modularidade, etc);
- Execução correta do código com entrada de testes a serem definidas no momento da avaliação. As entradas de testes irão exercitar a funcionalidade completa do código e, testar casos especiais ou de maior dificuldade de implementação, mas que devem ser tratados por um programa correto;
- Conteúdo da documentação que deve conter os itens mencionados anteriormente;
- Coerência e coesão da documentação (apresentação visual e organização, uso correto da língua portuguesa, qualidade textual e facilidade de compreensão);
- CASOS DE CÓPIA NÃO SERÃO TOLERADOS;
- Qualquer elemento que não esteja especificado neste documento, mas que tenha que ser inserido para que o protocolo funcione, deve ser descrito na documentação de forma explícita.

BOM TRABALHO!