Análise de Algoritmos

Trabalho Experimental de Algoritmos de Ordenação

Prof. Jaime Cohen

Ensino Remoto - 2020 / 2021

Universidade Estadual de Ponta Grossa

Sumário

Trabalho sobre Algoritmos de Ordenação

Projeto dos Experimentos

Relatório

Avaliação

Compilação do Código

Código Fonte (Linguagem C)

Universidade Estadual de Ponta Grossa

Ideia Geral do Trabalho

- Comparar os tempos de execução de 3 algoritmos de ordenação
- O código base inclui implementações de 2 algoritmos de ordenação
  - Ordenação por Dígitos
  - Quicksort
- Escolher o terceiro algoritmo

Universidade Estadual de Ponta Grossa

Complexidade de Tempo dos Algoritmos

Ordenação por Dígitos

- Tempo: $\Theta(d.n)$

Quicksort

- Pior Caso: $\Theta(n^2)$
- Caso médio: $\Theta(n \log n)$

Escolher um terceiro algoritmo de controle

Universidade Estadual de Ponta Grossa

Objetivos

- Executar experimentos para diferentes valores de *n* e *d*
- Determinar os valores para os quais:
  - Ordenação por Dígitos é mais rápido
  - QuickSort é mais rápido
  - São equivalentes

Universidade Estadual de Ponta Grossa

Objetivos

Automatizar os experimentos

- Execução dos experimentos
  - acrescente código em C para automatizar os experimentos
- Geração de tabelas e gráficos
  - pyplot, gnuplot, R, etc.

Projeto dos Experimentos

O projeto do experimento deve determinar

- valores de *n*
- valores de *d* (máximo 18)
- número de amostras
- estatísticas a serem usadas
- tipos de gráficos e tabelas a serem produzidos

Universidade Estadual de Ponta Grossa

Relatório

- descrição do terceiro algoritmo
- descrição do projeto dos experimentos
- resultados: tabelas, gráficos e discussão
- conclusões
- anexo: todo código fonte utilizado (C, scripts, etc.)

- aguarde a publicação do modelo de relatório para fazê-lo
- o relatório será feito em Latex

Avaliação

- Vale nota com peso de 3 atividades regulares
- Serão avaliados:
  - Projeto dos experimentos
  - Automatização dos experimentos e produção dos resultados
  - Relatório: estrutura, redação, clareza na apresentação dos resultados, discussão e conclusão
  - Cumprimento do prazo de entrega

```
$  gcc -O3 -o ordenacao ordenacao.c
```

- `gcc` : compilador C
- `-O3` : otimizar o código
- `-g` : não utilizar a opção `-g` para depuração
- `-o` : definir o nome do arquivo executável

```
./ordenacao 10000 5
```

```
Tempo do qsort():              0.001708 seg.
Tempo da ordenacaoDigital(): 0.000907 seg.
```

Código Fonte (Linguagem C)

main()

1. processa os argumentos de linha de comando
2. cria um vetor `v[]` de *n* strings de *m* dígitos aleatórios cada
3. cria um vetor `vi[]` de inteiros (long long) a partir de `v[]`
4. Chama `quicksort(vi, n, m)`
5. Chama `ordena_por_digitos(v, n, m)`

Código Fonte (Linguagem C)

`quicksort(vi, n, m)`

- executa a função `qsort()` da biblioteca padrão
- mede e imprime o tempo de execução em segundos

`ordena_por_digitos(v, n, m)`

- executa a função `ordenacaoDigital (v, n, m)` que implementa a ordenação por dígitos
- mede e imprime o tempo de execução em segundos

`void ordenacaoDigital (char **v, int n, int W)`

- implementa a ordenação por dígitos

Código Fonte (Linguagem C)

```
int teste()
```

- testa as funções de ordenação em vetores com 10 números de 7 dígitos cada
- imprime o vetor não ordenado e ordenado

Código Fonte (Linguagem C)

```
int comparador(const void * a, const void
* b)
```

- função usado pelo `qsort()` para comparar 2 números

Código Fonte (Linguagem C)

```
void strings_aleatorias(char **v, int n,
int m)
```

- gera `n` strings com `m` dígitos de 0 a 9 cada
- armazena o resultado em `v` (previamente alocado)

```
int randomInteger (int low, int high)
```

- gera um número aleatório entre `low` e `high`

Código Fonte (Linguagem C)

`print_v(mensagem, v, n)` e
`print_vi(mensagem, v, n)`

- imprime os vetores (para testes e depuração)