Trabalho Prático 03 – AEDS 1 – Em Dupla

Professora: Thais R. M. Braga Silva

Valor: 15 pontos

Data de Entrega: 28/11/2016 (PVANet) e 29/11/2016 (entrevistas)

Forma de Entrega: PVANet (formato .zip ou .tar.gz)

Este trabalho consiste em analisar o desempenho de diferentes algoritmos de ordenação em diferentes cenários, descritos a seguir. Esta análise consistirá em comparar os algoritmos considerando três métricas de desempenho: número de comparações de chaves, o número de cópias de registros realizadas, e o tempo total gasto para ordenação (veja como ao final).

Você deverá comparar o desempenho de diferentes variações do Quicksort para ordenar um conjunto de N inteiros armazenados em um vetor. As variações do Quicksort a serem implementadas e avaliadas são:

- **Quicksort Recursivo:** este é o Quicksort recursivo apresentado em sala de aula.
- **Quicksort Mediana(k):** esta variação do Quicksort recursivo escolhe o pivô para partição como sendo a mediana de k elementos do vetor, aleatoriamente escolhidos. Experimente com k = 3 e k = 5.
- **Quicksort Insercao(m):** esta variação modifica o Quicksort Recursivo para utilizar o algoritmo de inserção para ordenar partições (isto é, pedaços do vetor) com tamanho menor ou igual a m. Experimente com m = 10 e m = 100.
- **Quicksort Empilha Inteligente():** esta variação otimizada do QuicksortRecursivo processa primeiro o lado menor da partição.
- **Quicksort Iterativo:** esta variação escolhe o pivô como o elemento do meio (como apresentado em sala de aula), mas não é recursiva. Em outras palavras, esta é uma versão iterativa do Quicksort apresentado em sala de aula.
- **Quicksort Empilha Inteligente():** esta variação otimizada do Quicksort Iterativo processa primeiro o lado menor da partição.

Realize experimentos com as seis variações considerando vetores aleatoriamente gerados com tamanho N = 1000, 5000, 10000, 50000, 100000, 500000 e 1000000, no mínimo. Para cada valor de N , realize experimentos com 5 sementes diferentes e avalie os valores médios do tempo de execução, do número de comparações de chaves e do número de cópias de registros.

Apresente gráficos e/ou tabelas com os resultados obtidos. Discuta os resultados e conclusões obtidos. Qual variação tem melhor desempenho, considerando as diferentes métricas. Por quê? Qual o impacto das variações nos valores de k e de m nas versões Quicksort Mediana(k) e Quicksort Insercao(m)?

O seu programa principal deve ser organizado da seguinte maneira:

• Recebe a semente do gerador de números aleatórios bem como os nomes dos arquivos de entrada e de saída. Estes parâmetros devem ser passadospela linha de comando (argc e argv em C). Por exemplo:

quicksort 10 entrada.txt saida10.txt

onde *entrada.txt* tem o formato:

7 -> número de valores de N que se seguem, um por linha

1000

5000

10000

50000

100000

500000

1000000

• Para cada valor de N , lido do arquivo de entrada:

– Gera cada um dos conjuntos de elementos, ordena, contabiliza estatísticas de desempenho

– Armazena estatísticas de desempenho em arquivo de saída

Ao final, basta processar os arquivos de saída referentes a cada uma das sementes, calculando as médias de cada estatística, para cada valor de N.

Medindo o tempo de execução de uma função em C:

O comando getrusage() é parte da biblioteca padrão de C da maioria dossistemas Unix. Ele retorna os recursos correntemente utilizados pelo processo,em particular os tempos de processamento (tempo de CPU) em modo de usuário e em modo sistema, fornecendo valores com granularidades de segundos e microssegundos. Um exemplo que calcula o tempo total gasto na execução de uma tarefa é mostrado abaixo:

```
#include <stdio.h>
#include <sys/resource.h>
void main () {
      struct rusage resources;
      int rc;
      double utime, stime, total_time;
      /* do some work here */
      if((rc = getrusage(RUSAGE_SELF, &resources)) != 0)
            perror("getrusage failed");
      utime = (double) resources.ru_utime.tv_sec
            + 1.e-6 * (double) resources.ru_utime.tv_usec;
      stime = (double) resources.ru_stime.tv_sec
            + 1.e-6 * (double) resources.ru_stime.tv_usec;
      total_time = utime+stime;
      printf("User time %.3f, System time %.3f, Total Time %.3f\n",
            utime, stime, total_time);
}
```