COP-3530 Data Structures

Programming Assignment 2:
Effect of Picking Pivot in Quick Sort Running Time

Due Date: June 17 at 11:59 PM

This assignment has two parts: 1) Implementing multiple pivot picking methods for the quicksort routine given below, 2) comparing the running times of the implemented pivot picking methods in two given scenarios.

```
/**
* Internal quicksort method that makes recursive calls.
* Uses median-of-three partitioning and a cutoff of 10.
* @param a an array of Comparable items.
* @param left the left-most index of the subarray.
* @param right the right-most index of the subarray.
*/
private static void quicksort( double [ ] a, int left, int right )
{
    if( left + CUTOFF <= right )
    {
        double pivot = pivotPick( a, left, right );
        // Begin partitioning
        int i = left - 1, j = right + 1;
        for( ; ; )
        {
            while( a[ ++i ] < pivot ) { }
            while( a[ --j ] > pivot ) { }
            if( i < j )
                swapReferences( a, i, j );
            else
                break;
        }
        quicksort( a, left, i - 1 ); // Sort small elements
        quicksort( a, j+1 , right ); // Sort large elements
    }
    else // Do an insertion sort on the subarray
        insertionSort( a, left, right );
}
```

Figure 1: Quicksort routine using pivotPick method to find a pivot dividing the array into two subarrays and insertion sort for sorting fairly small arrays. Array a can be sorted by calling quicksort(a, 0, a.length − 1)

1 Implementing Multiple Pivot Picking Methods

Consider the quicksort routine shown in Figure 1. In this section, you need to implement 5 pivotPick methods with the signature *private static double pivotPick (double [ ] a, int left, int right)* used in the quicksort routine:

- *pivotPick1*: returns middle element of array at index $\left\lfloor \frac{\text{left+right}}{2} \right\rfloor$
- *pivotPick2*: returns the median of three elements of array at indices left, $\left\lfloor \frac{\text{left+right}}{2} \right\rfloor$, and right.
- *pivotPick3*: returns the median of five elements of array at indices left, $\left\lfloor \frac{3\cdot\text{left+right}}{4} \right\rfloor$, $\left\lfloor \frac{\text{left+right}}{2} \right\rfloor$, $\left\lfloor \frac{\text{left}+3\cdot\text{right}}{4} \right\rfloor$, and right.
- *pivotPick4*: returns the median of five values: μ and the four elements of array at indices left, $\left\lfloor \frac{2\cdot\text{left+right}}{3} \right\rfloor$, $\left\lfloor \frac{\text{left}+2\cdot\text{right}}{3} \right\rfloor$, and right. Value μ is the estimated statistical median of the array assuming that it contains uniformly distributed numbers in the given interval of [min, max]:

$$\mu = \min + \frac{\text{left} + \text{right}}{2(a.\text{length} - 1)}(\max - \min)$$

- *pivotPick5*: returns the median of five values: γ and the four elements of array at indices left, $\left\lfloor \frac{2\cdot\text{left+right}}{3} \right\rfloor$, $\left\lfloor \frac{\text{left}+2\cdot\text{right}}{3} \right\rfloor$, and right. Value γ is the estimated statistical median of the array assuming that it contains normally distributed values with mean m and standard deviation s:

$$\gamma = m + 4\Big(\frac{\text{left} + \text{right}}{a.\text{length} - 1} - 1\Big) \cdot s$$

2 Efficiency Comparison of the Implemented Pivot Picking Methods

In this part, you need to write a program that compares the efficiency of the pivot picking methods implemented in part one. To do so, you need to find and compare the running times of the following two scenarios for different pivotPick implementations:

2.1 Scenario 1: Sorting an Array of Uniformly Distributed Long Integers

The array is given by file "uniformlyDistributedFrom-12BTo+12B.csv" on Canvas. Also, the quicksort method uses the following method for picking the pivot: pivotPick1, pivotPick2, pivotPick3, and pivotPick4.

1. Read the file uniformlyDistributedFrom-12BTo+12B.csv and store it in array a.
2. For every $n \in \{100, 1000, 4000, 160000, 32000, 64000, 128000, 256000, 512000\}$, do the following:
   (a) long startTime = System.nanoTime();

(b) call quicksort $(a, 0, n-1)$

(c) long endTime = System.nanoTime();

(d) long totalTime = endTime - startTime;

(e) System.out.println(n + ", " + totalTime);

2.2 Scenario 2: Sorting an Array of Normally Distributed Long Integers

The array is given by file "normallyDistributedTruncatedFrom-12BTo+12B.csv" on Canvas. The numbers are normally distributed with mean $m = 0$ and standard deviation $s = 3,000,000,000$. Also, the quicksort method uses the following method for picking the pivot: pivotPick1, pivotPick2, pivotPick3, and pivotPick5.

1. Read the file normallyDistributedTruncatedFrom-12BTo+12B.csv and store it in array a.

2. For every $n \in \{100, 1000, 4000, 160000, 32000, 64000, 128000, 256000, 512000\}$, do the following:

   (a) long startTime = System.nanoTime();

   (b) call quicksort $(a, 0, n-1)$

   (c) long endTime = System.nanoTime();

   (d) long totalTime = endTime - startTime;

   (e) System.out.println(n + ", " + totalTime);

3 20% Bonus Part**

In the bonus part of the assignment, you are asked to write the following pivotPick method evaluating the exact statistical median of a normally distributed array. You need to compare its sorting performance with other methods used for Scenario 1:

pivotPick6: returns the median of five values: γ^* and the four elements of array at indices left, $\left\lfloor \frac{2 \cdot \text{left} + \text{right}}{3} \right\rfloor$, $\left\lfloor \frac{\text{left} + 2 \cdot \text{right}}{3} \right\rfloor$, and right. Value γ^* is the statistical median of the array assuming that it contains normally distributed values with mean m and standard deviation s:

$$\gamma^* = m + s \times \text{quantile}\Big(\frac{\text{left} + \text{right}}{2(a.\text{length} - 1)}\Big)$$

where quantile(p) is the quantile function of normal distribution defined in "quantile.csv" (available on Canvas).

4 Submissions

You need to submit a *.zip* file compressing the following folder:

- all the Java source file(s).
- An image showing the plot of running times (in milliseconds) of scenario 1 vs. $\log n$ when pivotPick1 is applied.
- An image showing the plot of running times (in milliseconds) of scenario 1 vs. $\log n$ when pivotPick2 is applied.
- An image showing the plot of running times (in milliseconds) of scenario 1 vs. $\log n$ when pivotPick3 is applied.
- An image showing the plot of running times (in milliseconds) of scenario 1 vs. $\log n$ when pivotPick4 is applied.
- An image showing the plot of running times (in milliseconds) of scenario 2 vs. $\log n$ when pivotPick1 is applied.
- An image showing the plot of running times (in milliseconds) of scenario 2 vs. $\log n$ when pivotPick2 is applied.
- An image showing the plot of running times (in milliseconds) of scenario 2 vs. $\log n$ when pivotPick3 is applied.
- An image showing the plot of running times (in milliseconds) of scenario 2 vs. $\log n$ when pivotPick5 is applied.

Please note that you can place all the four plots pertaining to one scenario in one coordinate system for the sake of easier comparison.