PROGRAMMING ASSIGNMENT #2 - Exhaustive Search

Student Name: **Phuc Le**
Course: CPSC 335
Section: 1

INPUT: a positive integer n and a list of n elements
OUTPUT: a longest non-decreasing subsequence of the initial sequence

End-to-Beginning Algorithm:

```
# Start the chronograph to time the execution of the algorithm
```

Time Unit

```
# populate the array H with 0 values
Arrays.fill(H, 0);
```

Subtotal: **1**

```
# loop to calculate the values of array H
for (i = n - 2; i >= 0; i--):
    for (j = n - 1; j > i; j--):
        if (A[i] < A[j] && H[i] <= H[j])
            H[i] = H[j] + 1
        end if
    end for
end for
```

n-1
n-i-1
3
2
Subtotal: $\sum_{i=1}^{n-1} 5(n-i-1)$

```
# calculate in max the length of the longest subsequence
max = H[0];
for (i = 1; i < n; i++):
    if (H[i] > max)
        max = H[i]
    end if
end for
```

1
n-1
1
1
Subtotal: **1 + 2(n-1)**

```
# allocate space for the subsequence R
R = new int[max + 1]
```

Subtotal: **2**

```
# store in j the index of the element appended to R
index = 0;
for (i = 0; i < n; i++):
    if (H[i] == max) {
        R[index] = A[i]
        index++
        max--
    end if
end for
# End the chronograph to time the loop
```

1
n
1
1
1
1
Subtotal: **1 + 4n**

* Total Running Time of End-to-Beginning Algorithm:

$$6n + 3 + \sum_{i=1}^{n-1} 5(n-i-1) = 6n + 3 + 5n\sum_{i=1}^{n-1} - 5\sum_{i=1}^{n-1} i - 5\sum_{i=1}^{n-1}$$

$$= 6n + 3 + 5n(n-1) - \frac{5\,n(n-1)}{2} - 5(n-1) = \frac{5\,n^2}{2} + \frac{5n}{2} + 8$$

PowerSet Algorithm:

```
# Start the chronograph to time the execution of the algorithm
                                                  Time Unit
bestSet = new IntegerObject[n + 1];                2
bestSize = new IntegerObject(0);                   1
printPowerset(n, bestSize, bestSet, A);            4 + 2^n(9n-5)  -- calculated below
R = new int[bestSize];                             1

#ASSIGN VALUES FOR RESULT SET R
for (i = 0; i < bestSize; i++):                    n
    R[i] = A[bestSet[i]]                           1
End for
# End the chronograph to time the loop             Subtotal: 8 + n + 2^n(9n-5)
```

* Total Running Time of PowerSet Algorithm:

$$8 + n + 2^n(9n-5)$$

Function printPowerset:

```
private static void printPowerset(int n, int bestSize, int[] bestSet, int[] A):
    #allocate space for the set                            Time Unit
    int[] stack = new int[n + 1]                            2
    stack[0] = 0                                            1
    int k = 0                                               1
    while (true):                                           2^n
        if (stack[k] < n):                                  1
            stack[k + 1] = stack[k] + 1                     3
            k++                                             1
        else:
            stack[k - 1]++;                                 2
            k--                                             1
        end if
        if (k == 0):                                        1
            break;
        end if
        checkSet(stack, k, bestSet, bestSize, A)            9n-11   -- calculated below
    end while
    return;                                                 Subtotal: 4 + 2^n(6 + 9n-11)
}
```

* Running Time of *printPowerset* Function:

$= 4 + 2^n(6 + 9n-11) = 4 + 2^n(9n-5)$

Function checkSet:

```
void checkSet(int[] stack, int k, int[] bestSet, int bestSize, int[] A):
    # function to check the currently generated set stack of size k
    # against the current bestSet of size bestSize                      Time Unit
    int i = 0;                                                          1
    if (k < 2):                                                         1
        if (k > bestSize.value) {                                       1
            bestSet[0] = new IntegerObject(stack[0]);                   1
            bestSize.value = k;                                         1
            return;
        end if
    else:
        for (i = 0; i < k - 1; i++):                                    n-2
            if (A[stack[i + 1] - 1] > A[stack[i + 2] - 1])              5
                return;
            end if                                                      Subtotal:2 + 5(n-2)
    end if
   #we have an non-decreasing
   #so we compare it against the current best set
    if (k > bestSize.value):                                            1
      # we found a better set,
       # STORE stack into bestSet and UPDATE bestSize to k
        for (i = 0; i < k; i++):                                        n-1
            bestSet[i] = new IntegerObject(stack[i + 1] - 1)            3
            bestSize.value = k                                          1
        end for
        return;                                                         Subtotal:1 + 4(n-1)
     else
        return;
     end if
```

* Running Time of *checkSet* Function:

$= 2 + 5(n-2) + 1 + 4(n-1) = 9n-11$

TIME EFFICIENCY:

*** End-to-Beginning Algorithm:**

$$\frac{5n^2}{2} + \frac{5n}{2} + 8$$

*** PowerSet Algorithm:**

$$8 + n + 2^n(9n-5)$$

End-to-Beginning Algorithm

$f(n) = \frac{5}{2}n^2 + \frac{5}{2}n + 8$

$f(n) \in \Theta(n^2)$

* Prove Time complexity using Limit:

$$L = \lim_{n \to \infty} \frac{\frac{5}{2}n^2 + \frac{5}{2}n + 8}{n^2} = \frac{5}{2}$$, a non-negative constant.

Thus, the relationship is TRUE

* Prove Time complexity using Definition

$$\frac{5}{2}n^2 + \frac{5}{2}n + 8 \leq c \cdot n^2 \quad , \quad c = ?, \; n_0 = ?$$

divide both sides by n^2

$$\frac{5}{2} + \frac{5}{2n} + \frac{8}{n} \leq c$$

$$\frac{5}{2} \xrightarrow{n \to \infty} \frac{5}{2} \qquad \frac{5}{2n} \xrightarrow{n \to \infty} 0 \qquad \frac{8}{n} \xrightarrow{n \to \infty} 0$$

$5/2 \quad 0 \quad 0 \leq c$ is TRUE

End-to-Beginning Algorithm

$f(n) = \frac{5}{2}n^2 + \frac{5}{2}n + 8$

$f(n) \in \Theta(n^2)$

* Prove Time complexity using Limit:

$$L = \lim_{n \to \infty} \frac{\frac{5}{2}n^2 + \frac{5}{2}n + 8}{n^2} = \frac{5}{2}$$, a non-negative constant.

Thus, the relationship is TRUE

* Prove Time complexity using Definition

$\frac{5}{2}n^2 + \frac{5}{2}n + 8 \leq c \cdot n^2$, $c = ?$, $n_0 = ?$

divide both sides by n^2

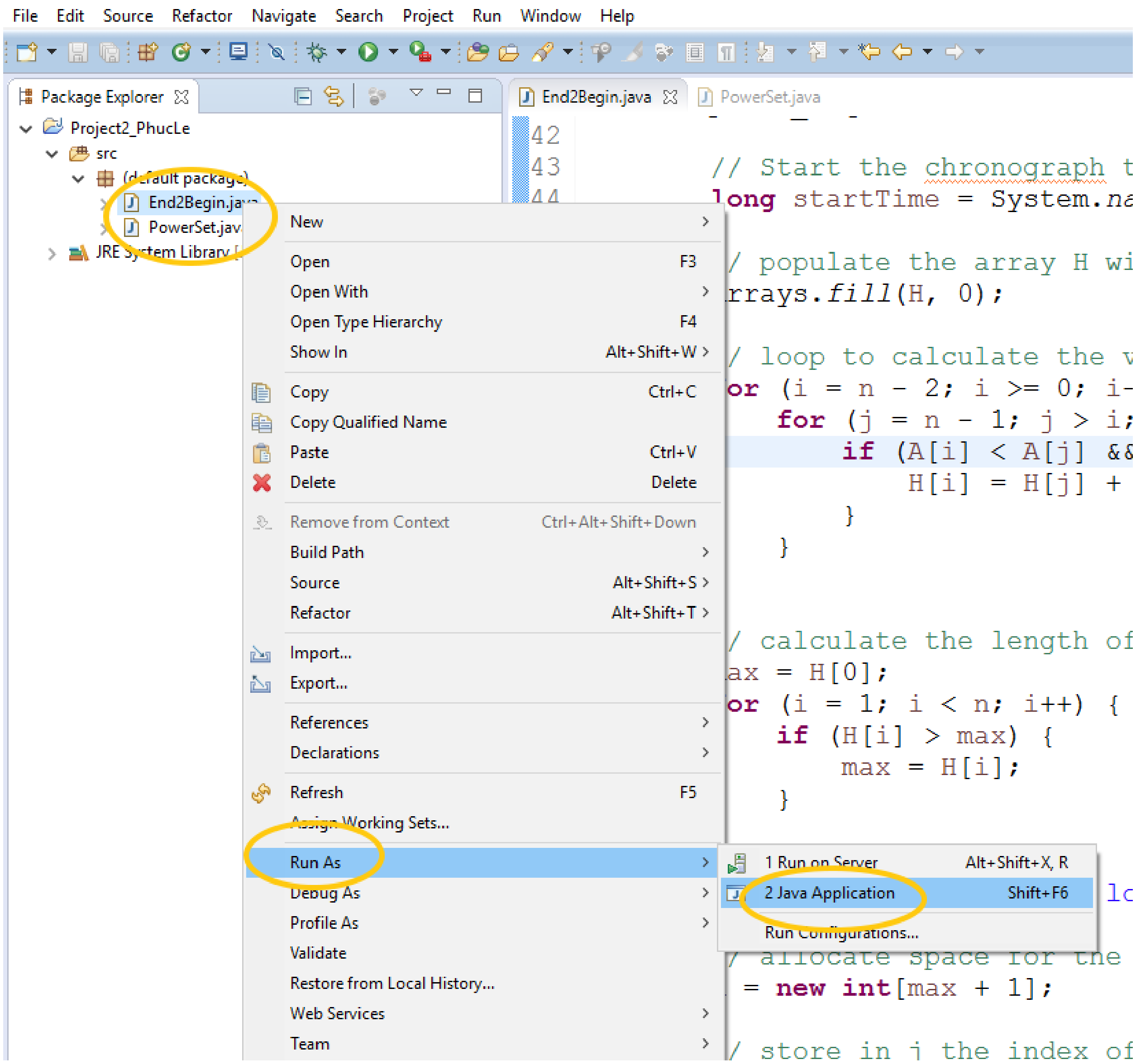
$$\frac{5}{2} + \frac{5}{2n} + \frac{8}{n} \leq c$$

$n \to \infty$ $n \to \infty$ $n \to \infty$

$5/2$ 0 $0 \leq c$ is TRUE

How to Run the Source Code - Method 1:

import project "Project2_PhucLe" into Eclipse Mar (version 4.5), open the appropriate algorithm file, right click on the source code, click "Run As", and click "Java Application"



How to Run the Source Code - Method 2:

copy 2 files: "***end2begin.jar***" and "***powerset.jar***" inside folder "***Executable Files***" into ***C:*** drive, open the console windows of commands and type the commands as the demonstration below:

`java -jar end2begin.jar`

```
C:\WINDOWS\system32\cmd.exe

C:\>java -jar end2begin.jar
CPSC 335-1 - Programming Assignment #2
Longest non-decreasing subsequence problem, end-to-beginning algorithm
Enter the number of elements in the sequence
5
Enter the elements in the sequence
0 8 4 12 2
Input Sequence
0 8 4 12 2
The longest non-decreasing subsequence has length
3
The longest non-decreasing subsequence is
0 8 12
Elapse time: 0.000139 seconds

C:\>
```

`java -jar powerset.jar`

```
C:\WINDOWS\system32\cmd.exe

C:\>java -jar powerset.jar
CPSC 335-1 - Programming Assignment #2
Longest non-decreasing subsequence problem, powerset  algorithm
Enter the number of elements in the sequence
16
Enter the elements in the sequence
0 8 4 12 2 10 6 14 1 9 5 13 3 11 7 15
Input Sequence
0 8 4 12 2 10 6 14 1 9 5 13 3 11 7 15
The longest non-decreasing subsequence has length
6
The longest non-decreasing subsequence is
0 4 6 9 13 15
Elapse time: 0.005435 seconds

C:\>
```

The Output Examples:

Example 1:

```
CPSC 335-1 - Programming Assignment #2
Longest non-decreasing subsequence problem, end-to-beginning algorithm
Enter the number of elements in the sequence
5
Enter the elements in the sequence
0 8 4 12 2
Input Sequence
0 8 4 12 2
The longest non-decreasing subsequence has length
3
The longest non-decreasing subsequence is
0 8 12
Elapse time: 0.000139 seconds
```

Example 2:

```
CPSC 335-1 - Programming Assignment #2
Longest non-decreasing subsequence problem, end-to-beginning algorithm
Enter the number of elements in the sequence
16
Enter the elements in the sequence
0 8 4 12 2 10 6 14 1 9 5 13 3 11 7 15
Input Sequence
0 8 4 12 2 10 6 14 1 9 5 13 3 11 7 15
The longest non-decreasing subsequence has length
6
The longest non-decreasing subsequence is
0 4 6 9 13 15
Elapse time: 0.000164 seconds
```

Example 3:

```
CPSC 335-1 - Programming Assignment #2
Longest non-decreasing subsequence problem, powerset algorithm
Enter the number of elements in the sequence
5
Enter the elements in the sequence
0 8 4 12 2
Input Sequence
0 8 4 12 2
The longest non-decreasing subsequence has length
3
The longest non-decreasing subsequence is
0 8 12
Elapse time: 0.000765 seconds
```

Example 4:

```
CPSC 335-1 - Programming Assignment #2
Longest non-decreasing subsequence problem, powerset algorithm
Enter the number of elements in the sequence
16
Enter the elements in the sequence
0 8 4 12 2 10 6 14 1 9 5 13 3 11 7 15
Input Sequence
0 8 4 12 2 10 6 14 1 9 5 13 3 11 7 15
The longest non-decreasing subsequence has length
6
The longest non-decreasing subsequence is
0 4 6 9 13 15
Elapse time: 0.005435 seconds
```