Objective of this assignment:

* To explore the impact of the *function calls* overhead

What you need to do:

1. Implement the greedy **recursive** algorithm to solve the activity-selection problem
2. Implement the greedy **iterative** algorithm to solve the activity-selection problem
3. Repeatedly execute both algorithms on the **same** problem and measure the running time of each algorithm
4. Plot results, compare, analyze and conclude.

**Objective**:

The objective is to study the overhead of the function *calls*. Recursive algorithms call themselves to solve problems. Iterative algorithms do not. Throughout this course (and the textbook), we read that while recursive algorithms may have the same asymptotic running times as iterative homologous algorithms, they are in general less efficient than iterative algorithms (i.e., running time differ by the coefficients of the growth functions). This makes sense because function calls are not free: they take CPU time (just refer to your assembly course of what the cost of the *CALL* and *RET* instructions), left alone the management of the parameters on the stack. This lab aims to check this empirically.

**Programming**

1. Implement RecursiveActivitySelector(k,n), the greedy **recursive** algorithm to solve the activity-selection problem.
2. Implement GreedyActivitySelector(n), the greedy **iterative** algorithm to solve the activity-selection problem.
3. Implement the following program to collect data to plot and analyze. (submit this program with your assignment)

**StudyOverhead(NumberPoints)**

Initialize Array\_s[n] // start times

Initialize Array f[n] // finish times

for i = 1 to NumberPoints

TimeRecursive = 0

TimeIterative = 0

for j = 1 to NumberRuns

Initialize set A //Use an array to represent a set A[i] = 0 if

RecursiveActivitySelector(0, **i-1**)

Collect running time for recursive and add it to TimeRecursive

GreedyActivitySelector(**i-1**)

Collect running time for iterative and add it to TimeIterative

**Collect M[i]** = TimeRecursive/TimeIterative

Dump M[i] in a file

**InitializeArrays(n)** // Create about n/2mutually compatible activities

s[0] = 0

f[0] = 0

for i = 1 to n-1

if (i is even)

s[i] = f[i-2]

f[i] = s[i] + 2

else

s[i] = f[i-1] - 1 // s[1] will be negative, but that is fine.

f[i] = f[i-1]+1

**Data collection and analysis**

1) (25 points) Plot M[i] versus i

Insert here the plot ...

**I trimmed the first 500 data points off so the initial overhead execution time wouldn’t affect the plot.**

Chart, scatter chart

Description automatically generated

2) (45 points) Analyze your results and answer the question we asked at the beginning of this programming assignment. Is the iterative algorithm more efficient than the recursive one? You should set the variable NumberPoints and NumberRuns such that they are not too large or too small. If these variables are too large, you will wait too long to collect data (depends on the machine you are using). If the values are too small, you may not see much difference between the two algorithms.

*Compare the two algorithms, discuss and analyze* based on the plot of M[i] versus i. .....

**My program works correctly.**

**Theoretically, the recursive algorithm should be slightly less efficient than the iterative algorithm. This is because in the recursive method you have to make calls to the function multiple times, in comparison to the iterative function that does not have to make such calls. This is of course under the assumption that the two algorithms have the same basic time complexity of O(nlogn). Because of this, we would leave the iterative model as represented by the time complexity O(nlogn), but would change the recursive model to be represented by O(Anlogn), where the A is some constant. We add this constant in to represent the time taken by calling the recursive function. We can confirm this by examining our plot attached in the previous problem. In the plot, we are showing the results of dividing the running time of the recursive algorithm by the iterative algorithm as the activity problem they analyze grows. If we divided these time complexities, it would be Anlogn/nlogn. This result of this division would be just the constant A. As we can see, the graph above overall shows a horizontal line with minor variation. This horizontal line is what we would expect out of a representation of a constant. Also, since this number is fairly consistently above the value of 1, that confirms our theory that the recursive algorithm takes more time than the iterative algorithm. So, through our testing and analysis of our plot, we can confirm that the recursive algorithm is less efficient than the iterative algorithm.**

**How to compile: Navigate to the directory (I was on bmm0066@tux251) The file is just in that home directory. StudyOverhead.java should be there. Type in “javac StudyOverhead.java”. After compilation finishes, type in “java StudyOverhead”. The program should run and it will output to the system how many iterations are done, it will run to 2000. Once done, there should be a file called “Output.csv” with all the output data.Report**

* Write a report that will contain, explain, and discuss the plot. The report should not exceed one page.
* In addition, your report must contain the following information:
  + whether the program works or not (this must be just ONE sentence)
  + the directions to compile and execute your program
* Good writing is expected.
* Recall that answers must be well written, documented, justified, and presented to get full credit.

**What you need to turn in:**

* Electronic copy of your source program (standalone/separately attached to this assignment)
* Electronic copy of the report (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.

**Grading**

* Program is worth 30% if it works and provides data to analyze
* Quality of the report is worth 70% distributed as follows: good plot (25%), explanations of plot (10%), discussion and conclusion (35%).