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]** = TimeIterative/TimeRecursive

Dump i and M[i] in a csv file F (submit this 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

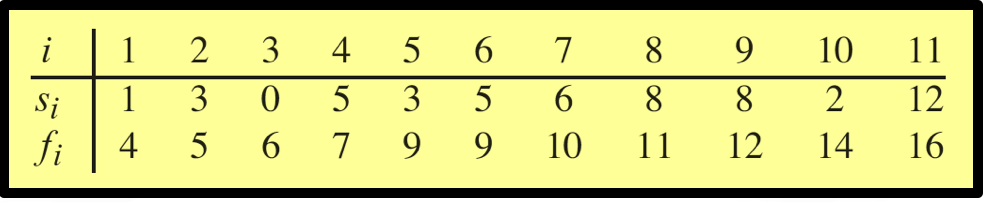
else

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) (15 points) On a Tux machine, compile and execute the algorithm *RecursiveActivitySelector* algorithm on the example below. Take a readable screenshot of date, the compilation directives, the execution, the right output.

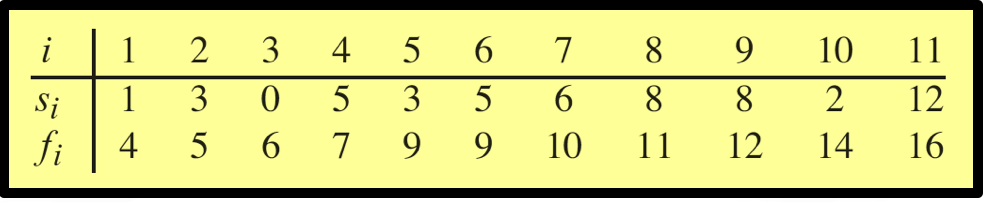


Make sure we see your username and tux machine name as readable as this screenshot template.



Insert here the screenshot here ...

2) (15 points) On a Tux machine, compile and execute the *GreedyActivitySelector* algorithm on the example below. Take a readable screenshot of date, the compilation directives, the execution, the right output.



Make sure we see your username and tux machine name as readable as this screenshot template.



Insert here the screenshot here ...

2) (30 points) Plot M[i] versus i

(5 points) Submit the csv File f (see pseudocode)

(25 points) Insert here the plot ...

A graph with a line

Description automatically generated

3) (40 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. .....

When we pit the iterative greedy activity selector against the recursive one, the plot of M[i] ratios across different runs tells us the story we're looking for. The values are hanging out around 1, which means that, typically, the iterative algorithm isn't beating the recursive one in terms of efficiency, nor is it lagging behind. It's a tie game.

In the context of the activity selection problem, both the iterative and recursive approaches are viable contenders. The iterative version, with its direct loop-through logic, and the recursive version, with its layered problem-solving, are finishing the job in about the same amount of time. This consistent equivalence in performance indicates that the efficiency of one algorithm over the other isn't a factor here. So to answer the question, "Is the iterative algorithm more efficient than the recursive one?" the data says no, they're equally efficient.

**Report**

* Write a report that will contain, explain, and discuss the plot..
* 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 csv file
* Electronic copy of the report (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.

**Grading**

Following the provided points distribution.