Instructions

Your submission is graded (pass/fail) based on what you submit on learnit alone. The CodeJudge is hopefully running during the exam and is meant only as a debugging aid and to help you meet the specification of the input and output. We expect two files, one pdf and one archive (tar, tgz or zip) with all implementations, with file names that indicate what question is addressed. State the compiler and command line options of the compiler that you are using. If your code compiles on CodeJudge, the shortname there is enough.

You are allowed to use any external source. If you do so, you have to give a reference. I.e.: You can copy paste small pieces of code from publicly available sources as much as you want as long as it is clear which part of your implementation (or text) is literally copied, or based on a copy. If we find that you used a source without referencing it, we will consider this a case of cheating. Of course we base our grading only on your contribution, so it is important that it becomes clear what goes beyond the copied pieces. Before the deadline for handing in, you are not supposed to discuss this exam with anybody (by whatever medium) and you are not supposed to share your solutions or receive outside advice or solutions. If we find indications (in the check directly after the exam , while grading or in any other way) that you received such help, we will report it as a suspected case of cheating.

To pass the exam, it is sufficient to answer 2 out of the 3 questions in a satisfactory manner.

Don't forget to answer the questions that go beyond submitting your source code. (That is what the pdf is for). Do not include your complete implementation into the pdf. If you feel the need to comment on your code, try to restrict yourself and include only a few lines, or reference to the code. A comment in the code might also be a good idea in such situations. The code is part of your submission anyway.

1 Sum of Degrees Squared

Given an undirected graph, compute the sum of the squares of the degree of all vertices. The input is to be read from `stdin`. It starts with a line with the two integers n and m denoting that $V = \{0, \dots, n-1\}$ and that $|E| = m$. Then follow m lines with two numbers u and v (separated by space), each encoding an undirected edge (u, v) of G. Note that parallel edges and loops are allowed, loops count as two edges.

Use an array of size n to compute the degrees of the vertices while reading the input file.

Input:

```
3 3
0 1
1 2
2 0
```

Output:

```
12
```

Input:

```
4 3
0 1
0 2
0 3
```

Output:

```
12
```

Input:

```
5 4
0 1
1 2
2 3
2 4
```

Output:

```
16
```

1.1 Implement

[**30** min]

Write a program in C, C++, java, python (2 and 3), bash, pascal, C#, or F# that implements this task. Submit your source code. You can try this on CodeJudge as "Question 1".

1.2 Questions to discuss

1. [**5** min] Can you avoid a final scan over the array of degrees to compute the sum of the squares?

2. [**5** min] Assume you have a good implementation of the above algorithm that operates on the same huge graph where all edges are represented by (u, v) with $u < v$. You observe a 40% time difference between the following two representations:

   (a) the edges appear in random order in the input file.

   (b) the edges appear sorted according to u in the input file.

   Give an explanation for this behavior.

3. [**5** min] Which model of computation could help quantify the phenomenon?

4. [**5** min] Assume that your code is supposed to be moved to a new computing platform, for which you want to quantify the above effect. Design a simple experiment in terms of its performance metric, parameters and factors.

2 Quality of Iterated Hashing

This question is accompanied by three tasks on CodeJudge. Only the last one corresponds to a program that you are required to hand in. The first and the second task are meant as building up to the third in the sense of a debugging aid.

Consider the hash function h defined by h(`x`)= `0x7fffffff &( ((x*32)^(x/2)) + 1 )` (Here `0x` starts a constant in hexadecimal notation, `&` is bitwise and, `^` bitwise xor; We also use $a\%b$ to denote the remainder of the integer division a/b.) We will consider iterated application of h, i.e. the family of hash function $h_i(x) = h(h_{i-1}(x))$ and $h_0(x) = x$. To clarify this definition, observe that we can express $h(i, x) = h_i(x)$ in pseudocode in the following way:

```
integer h(integer i, integer x):
   r = x
   for(int j = 0; j < i; j++)
       x = h(x)
   return r
```

Assume in your application the following set of keys is generated:

$$S = \{x_j = ((2^{j\%5}+j*32)*16 \mid j = 0 \ldots 1023\} = \{16, (2+32)*16, (4+64)*16, \ldots, (2^{1023\%5}+1023*32)*16\}$$

Your task is to assess the quality of $h_i(x)\%8$ as a hash function on S for different i. To this end compute and plot the maximal number of items that hash to the same value. More formally, compute $m(i) = \max_{k=0,\ldots,7} |\{x \in S \mid k = h_i(x)\%8\}|$.

Your program creates an array $a[]$ of size 1024 and sets $a[j] = x_j$. Note that $h_0(x_j) = x_j$, i.e., after initialization the array $a[]$ stores what we could call $h_0(S)$. Then it loops and increases the value i: If $a[]$ represents $h_i(S)$, setting $a[j] = h(a[j])$ for all $j = 0, \ldots, 1023$ makes $a[]$ represent $h_{i+1}(S)$. This part of your program you can check on CodeJudge: Your program for "Question 2a" should expect the integer i on the command line and print the array $a[]$ (representing $h_i(x_j)$) to stdout, for increasing j, one entry per line.

Next, you are supposed to count the load balancing achieved with $h_i(x)\%8$ on S: First, compute how many items of S fall into the 8 different buckets, i.e., count for each $k = 0 \ldots 7$ the number $f_i(k)$ of positions j in the table with $a[j]\%8 = k$. This part of the program you can check on CodeJudge as "Question 2b". It expects a number i on the command line For $i = 0$ you see the following counts:

```
0: 1024(0)    0(1)    0(2)    0(3)    0(4)    0(5)    0(6)    0(7)
```

and for $i = 2$:

```
2:    0(0)  384(1)    0(2)    0(3)    0(4)  640(5)    0(6)    0(7)
```

The above format is that each line of the table starts with i in decimal, followed directly by a colon, then the table for $f_i(k)$ with k in parenthesis following directly. This second program should expect a single integer n on the command line and then produce the compute the first n lines of the above table, i.e. for $i = 0$ to $i = n - 1$.

Note that $m(i)$ is the maximum $\max_{k=0,\ldots,7} f_i(k)$.

2.1 Implement

[**60** min] Write a program in C, C++, java, python (2 and 3), bash, pascal, C#, or F# enabling you to compute $m(i)$. Submit the source code of this implementation. This task corresponds to the “Question 2c” on CodeJudge. Your program is expected to produce for $i = 0, \dots, 999$, one line for each i with i and $m(i)$, separated by a space, to `stdout`. It will get 1000 as argument on the command line, but you are welcome to ignore this.

Remember that the other two tasks on CodeJudge are not mandatory, they are rather meant as a debugging aid.

2.2 Plot

[**10** min] Discuss your table/plot addressing the following questions: How many items would be in each of the eight buckets if the hash function would distribute them evenly? How close to that minimum does $m(i)$ get? What happens for large values of i, say $i > 300$?

2.3 Implementation Question to discuss

[**5** min]

Now consider a generalization of the program where the array $a[]$ is really large. Can the above program to create the statistics be sped up by multi-core parallelization? What about using vectorized instructions?

3 Representatives

In this task you are given a set of entities, and you need to find for each entity a number representing it, which means that no two entities share the same number as representative. Each entity comes with an upper and a lower bound on the value of its representative. The computational task is, given a list of such upper and lower bounds, find admissible and unique representatives.

Input

The input starts with the number of numbers n. The following n lines each contains two floating point numbers l_i and u_i, understood as a lower and upper bound respectively.

Output

A list of distinct integers $m_1, \ldots, m_n$ such that $l_i \leq m_i \leq u_i$ for all i.
If this cannot be done, print "`impossible`."

Input:

```
1
1.4 1.7
```

Output:

```
impossible
```

Input:

```
4
-.5 3.5
-.5 3.5
-2 9
1.9 2.1
```

Output:

```
0 1 -2 2
```

Input:

```
3
.5 2.5
.5 2.5
.5 2.5
```

Output:

```
impossible
```

This problem can be solved with a greedy algorithm. It first transforms the floating point numbers into a list of intervals on the integers that represent the admissible integers.

The task can now be understood as a scheduling problem, where we have unit time tasks that can only be executed in a time window. The algorithm sorts the intervals by starting time (left endpoint of the interval), it keeps and advances a current time and always schedules the most urgent available task first. It is formulated in the following pseudocode:

```
Data: List of pairs $(l_i, u_i)$ of floating point numbers
Result: List of distinct integers $m_i$ with $l_i \leq m_i \leq u_i$
Create the list $L$ of integer intervals $l_i = [a_i, b_i]$ with $a_i = \lceil l_i \rceil$ and $b_i = \lfloor u_i \rfloor$
Sort the intervals $L$ by increasing start time $a_i$
Create empty priority queue $PQ$ of intervals, smallest end time first
while $L$ not empty do
    move the first (remaining) interval $[a_i, b_i]$ of $L$ into $PQ$
    set current time to $a_i$
    while $PQ$ not empty do
        move all intervals with current time as start time from $L$ into $PQ$
        Let $e$ be the first element of $PQ$ and remove it (most urgent job)
           (break ties such that earlier input is scheduled first)
        Assign the current time as representative to $e$
        Fail if current time is not inside the interval $e$
        increase current time by one
    end
end
Report the time assigned to interval $l_i$ as representative $m_i$
```

3.1 Implement

[**60** min]

Write a program in C, C++, java, python (2 and 3), bash, pascal, C#, or F# that implements this task. Use an appropriate library function for sorting and the priority queue. If you are not using the C++ `std`, the java `java.util` or the python `heapq` library, give a pointer to the library you are using. Submit your source code. You can try this on CodeJudge as “Question 3”.