Algorithms and Parallel Computing

Course 052496
Prof. Danilo Ardagna

Date: 14-06-2022

Last Name: .....
First Name: .....
Student ID: .....
Signature: .....

Exam duration: 2 hours and 15 minutes

Students can use a pen or a pencil for answering questions.

Students are NOT permitted to use books, course notes, calculators, mobile phones, and similar connected devices.

Students are NOT permitted to copy anyone else's answers, pass notes amongst themselves, or engage in other forms of misconduct at any time during the exam.

Writing on the cheat sheet is NOT allowed.

Exercise 1: _____ Exercise 2: _____ Exercise 3: _____

Exercise 1 (14 points)

A tensor is a data container which can store data in N dimensions. You have to develop a class implementing tensors up to $N = 3$ dimensions for **double**s. You have three main goals: i) your implementation should optimize the **worst case complexity**, ii) you need to support **random access**, iii) your data structure will be **sparse**.

Considering that any index starts from 0, you have to:

1. implement the constructor which receives as input parameter the dimensions of your `Tensor` class
2. provide the definition of the method:

   ```
   void set(double value, unsigned i, int j=-1, int k= -1)
   ```

   k has default value -1 to denote the scenarios where the tensor has one or two dimensions, while j default value -1 characterizes one dimensional tensors. This method will add (for an unseen index) or update (in case the index already exists) values in your tensor objects.
3. implement the method:
   `double get(unsigned i, int j=-1, int k= -1) const`
   k has default value -1 to denote the scenarios where the tensor has one or two dimensions, while j default value -1 characterizes one dimensional tensors. In other words, if t1 is a 1-dim tensor, t2 is a 2-dim tensor and t3 is a 3-dimensional tensor, the user of your class can write the following code raising in some scenarios error conditions:

   ```
   t1.set(1,0); //OK t1[0]=1
   t2.set(-2,0,1); //OK t2[0][1]=2
   t2.set(1,0,0); //OK t2[0][0]=1
   t2.set(2,0,0); //OK t2[0][0]=2
   t3.set(3,2,2,2); //OK t3[2][2][2]=3
   std::cout << t1.get(0) << std::endl; //OK
   std::cout << t2.get(0) << std::endl; // Index not appropriate
   std::cout << t2.get(0,0) << std::endl; // OK
   std::cout << t3.get(2,2,2) << std::endl; // OK
   std::cout << t3.get(3,3,3) << std::endl; // Index not appropriate
   std::cout << t1.get(1) << std::endl; // Index not appropriate
   std::cout << t2.get(1,0) << std::endl; // Index not appropriate
   ```

In your implementation **you have to cope with error conditions** and check if indexes are appropriate. In case of error `get` will return a NaN while `set` will not have effect in your data structure.

Moreover, provide the implementation and complete the declarations of the following methods **(you have to consider also error conditions and add any missing `const` qualifiers)**:

4. `Tensor operator*(double alpha)`: the product of the tensor with a double
5. `double norm(const std::string& norm_type))`: the method computes entry-wise tensor norms (i.e., each tensor element is treated as a vector element). `norm_type` can be "inf", "euclidean", "2" (same as euclidean), "fro" (i.e., Frobenius norm). Note that "fro" can be applied only if the tensor is a bi-dimensional matrix.
6. `void reshape(.... new_shape)`: **discuss (providing high level ideas, no code is required)** how you would implement this method that can change the tensor size (for example, a two-dimensional tensor t2 can be transformed into a one-dimensional tensor as you do in matlab through t2(:)).

Additionally, provide:

7. the **worst complexity** of all the methods you have implemented.