NumCSE exercise sheet 4
Convolution and FFT, filtering, Lagrange interpolation

alexander.dabrowski@sam.math.ethz.ch
oliver.rietmann@sam.math.ethz.ch

November 13, 2018

Exercise 4.1. *(Long exercise) Sparse vectors, their convolution and FFT*

Given a vector `x` of numbers, we want to represent it with a new data structure which should be suitable in case most of its entries are negligible. We also want to implement convolution and fast Fourier transform for this new data structure.

(a) Consider the following two `structs`:

- `template<class T> duplet`, which has members
  - `int ind`, the index of an element in `x`;
  - `T val`, the value of the element at position `ind`;
- `template<class T> sparse_vec`, which has members
  - `double tol`. We consider negligible any number with absolute value smaller than `tol`;
  - `duplets`, a `vector` of objects `duplet<T>` which stores the indices and values of the non-negligible elements of `x`;
  - `len`, an integer which records the length of `x`.

Implement in `sparse_vec<T>` the following utility methods:

- `void append(int ind, T val)`, which appends to `duplets` a new duplet with index `ind` and value `val`, if `abs(val)>=tol`;
- `void cleanup()`, which sorts `duplets` with respect to `ind`, eliminates repetitions, with the convention that if two elements have the same index their values should be added up, and eliminates any element with norm smaller than `tol` or with index larger than `len-1`. If `x.cleanup()` leaves `x` as it is, we say that `x` is **clean**;
- `T get_val(int ind)`, which returns the element of `x` in position `ind`. It should work in $O(\log$ `duplets.size()`$)$ time and assumes that `x` is clean.

Example: A representation of $x = (0, 0, 1, 0, 7 + i, 0)$ can be obtained with a `sparse_vec` with `duplets = {(ind=4,val=6-I), (ind=1,val=1e-7), (ind=22,val=9), (ind=4,val=1+2I), (ind=2,val=1)}`, `len=6`, `tol=1e-6`. Notice that `duplets.size()!=len`. Running `cleanup()` on such an object, `duplets` would become `{(ind=2,val=1), (ind=4,val=7+I))}`.

Important notes: In all the subproblems below, you can assume all vectors of type `sparse_vec` which are passed as inputs are already clean. Do not convert the `sparse_vec`s to/from dense vectors.

(b) In `struct sparse_vec` implement a method `cwise_mult` which returns the component-wise multiplication between `sparse_vec a` and `sparse_vec b` in $O(sz_a + sz_b)$ time, where sz_a, sz_b are respectively the number of non-negligible elements of `a` and `b`.

(c) Implement a method `conv` which returns the discrete convolution of `sparse_vec a` and `sparse_vec b` in $O(sz_a\, sz_b)$ time.

(d) Implement a method `fft` which returns the discrete Fourier transform of `sparse_vec x` in $O(n(\log n)^2)$ time, where $n =$ `x.len`. You can assume that n is a power of 2. Your function should return a `sparse_vec` which is clean.

(*Optional*) Improve your code so that it runs in $O(n \log n)$. Even if you don't implement this improvement, you can suppose in the following subproblems that the runtime of `fft` is $O(n \log n)$.

(e) Implement a method `ifft` which returns the inverse discrete Fourier transform of `sparse_vec x` in $O(n \log n)$ time. You can assume that n is a power of 2.

(f) Let `a` and `b` be two complex vectors of length $n + 1$ and n respectively. Assume that n is a power of 2. Implement a method `conv_fft` which returns the discrete convolution of `a` and `b` in $O(n \log n)$ time.

(g) Can `conv_fft(x, y)` be asymptotically slower than `conv(x, y)` for a particular choice of `x` and `y`? Briefly motivate your answer.

Exercise 4.2. *2D convolution, FFT2 and Laplace filter*

We review the two dimensional convolution, its relation with the discrete Fourier transform, and implement a discrete Laplacian filter.
Consider the 2-dimensional infinite arrays $X = (X_{k_1,k_2})_{k_1,k_2\in\mathbb{Z}}$ and $Y = (Y_{k_1,k_2})_{k_1,k_2\in\mathbb{Z}}$. We can define their convolution as the 2-dimensional infinite array $X * Y$ such that

$$(X * Y)_{k_1,k_2} = \sum_{j_1=-\infty}^{\infty} \sum_{j_2=-\infty}^{\infty} X_{j_1,j_2} Y_{k_1-j_1,k_2-j_2}.$$

In the same way as in the 1-dimensional case, given two matrices $X \in \mathbb{R}^{m_1,m_2}$ and $Y \in \mathbb{R}^{n_1,n_2}$ we can define their *discrete convolution* as the convolution between the zero extensions of X and Y trimmed of unnecessary zeros, and their *circular convolution* as the smallest period of the convolution between the periodic extension of X and the zero extension of Y.
Consider two matrices $A \in \mathbb{R}^{n,m}$ and $F \in \mathbb{R}^{k,k}$ with $k < n, m$.

(a) How should we extend A and F to larger matrices $\tilde{A}$ and $\tilde{F}$ so that the discrete convolution of A with F is equal to the circular convolution of $\tilde{A}$ with $\tilde{F}$?

(b) By recalling to the 1-dimensional fast fourier transform (see the `Eigen::FFT` module), implement a function `fft2` which returns the 2-dimensional fast fourier transform of a matrix.

(c) Implement a function `ifft2` which returns the 2-dimensional inverse fast fourier transform of a matrix.

(d) Implement an efficient function with arguments A and F which computes their discrete convolution.

Hint: use the result derived in the previous steps and the two dimensional circular convolution theorem.

(e) The discrete Laplacian filter is defined as

$$F = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}.$$

It is often used to detect edges in images. Test your filter on the black and white "picture" provided in the template.

Exercise 4.3. *Lagrange interpolation.*

Fix $n \in \mathbb{N}_0$ and let $t_0, \ldots, t_n \in \mathbb{R}$ be distinct nodes, i.e. $t_i \neq t_j$ if $i \neq j$. Then

$$L_i(t) := \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{t - t_j}{t_i - t_j}$$

is called the i-th *Lagrange polynomial* for these given nodes. For $y_0, \ldots, y_n \in \mathbb{R}$ we call

$$p(t) := \sum_{i=0}^{n} y_i L_i(t)$$

the *Lagrange interpolant* through $(t_i, y_i)_{i=0}^n$.

(a) Prove that

$$\sum_{i=0}^{n} L_i(t) = 1$$

for all $t \in \mathbb{R}$.

Hint: Choose $y_i = 1$ for all $i \in \{0, \ldots, n\}$ and use uniqueness of Lagrange interpolants.

(b) For $t \in \mathbb{R}$ define $\omega(t) := \prod_{j=0}^{n}(t - t_j)$ and fix $i \in \{0, \ldots, n\}$. Prove that $\omega'(t_i) \neq 0$ and

$$L_i(t) = \omega(t) \frac{\lambda_i}{t - t_i}$$

for all $t \in \mathbb{R}$, where

$$\lambda_i := \frac{1}{\omega'(t_i)}.$$

(c) Compute the Lagrange interpolant $p(t)$ corresponding to the data given in Table 1.

i	t_i	y_i
0	-1	2
1	0	-4
2	1	6

Table 1: data

(d) Use the Newton basis approach to compute the interpolating polynomial $\tilde{p}(t)$ for the data in Table 1.

(e) Is $\tilde{p}(t)$ different from $p(t)$? Explain your answer.

(f) What are the advantages of using the Newton basis compared to the Lagrange polynomials?