Bilkent University
[cs481 | cs583] - Bioinformatics Algorithms

Spring 2025

Homework Assignment #0

GENERAL INSTRUCTIONS

FAILURE TO FULFILL ANY OF THE FOLLOWING ITEMS WILL RESULT IN A GRADE OF 0 (zero), WITHOUT ANY CHANCE OF REDEMPTION.

- You must **write your code yourself**. Sufficient evidence of plagiarism will be treated the same as for plagiarism or cheating.
- **C / C++, Rust, Python or Java** will be used as programming language. STL for C++, and APR for C are allowed. The assignments are created with C / C++ in mind, so using other languages would require some tweaking.
- Your code must **compile**.
- Your code must **be complete**.
- Your code must **run on dijkstra.cs.bilkent.edu.tr** server.
- Your code must make use of **argument parsing**.
  Refer to: https://docs.google.com/presentation/d/1rU0DhBg6yVXbfEtNuK73pjc1yWSPG90YnGNH-b-kklQ
- Submit your answers **ONLY** through the Moodle page.
- **Zip** your files and send them in only one zipped file. File name format `surname_name_hw#.zip`.
- The zip file **must contain** the following items:
  - All the source files.
  - `Makefile` to compile the source code and produce the binary. Even if you use Python, include this Makefile.
  - A `README.txt` file that briefly describes how your program works.
- All submissions must be made **strictly before the stipulated deadline**.
- A bonus will be given for the fastest code that solves the assignment successfully.

---

1) ENVIRONMENT SETUP

Aim: In this assignment, you will be familiarized with the logistics of the following assignments. Every assignment MUST have the following features:

- Running on DIJKSTRA server: in order to have a standardized platform for development and testing, your assignments must run on dijsktra.cs.bilkent.edu.tr.
- A README file: containing information on HOW to run the program.
- A MAKEFILE file: which will be used to compile and test-run your program. Makefiles must be included even if you are using Python or Java.
- Make use of COMMAND-LINE ARGUMENTS: to be able to run with different inputs without recompiling the code.

1.1 DIJKSTRA

To obtain a user account on the server, email Mr. Ahmet Hakan Göktaş goktas@bilkent.edu.tr. After that you can follow the link http://cs.bilkent.edu.tr/~oalbayrak/cs202/dijkstra_upload_guide.pdf for a guide on how to use the server.

1.2 README

README files belong to the GNU Coding Standards. You can find out about them in the link https://www.gnu.org/prep/standards/standards.html#index-README-file. Please include the following your information in it: full name, Bilkent ID, email, Course, Assignment number and the instructions on how to run your program.

1.3 MAKEFILE

MAKEFILE is a tool for compiling and running programs in *unix environments. A crash-course on how to use them can be found in the link https://sites.google.com/view/rrb-codebits/makefile. Your Makefile must include at the very least three targets: COMPILE, RUN, CLEAN, for compiling, running an example, cleaning the directory. For the RUN target, choose any parameters to do a test run of your program.

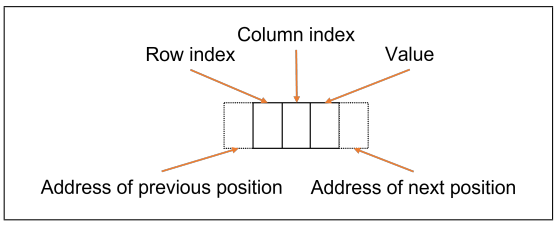
1.4 COMMAND-LINE ARGUMENTS

Most of the programs you will use in *unix can and must take arguments that can tweak its execution. Depending on the programming language you are using, their implementation will differ. In the links below, you can find examples for C, C++, Java and Python.

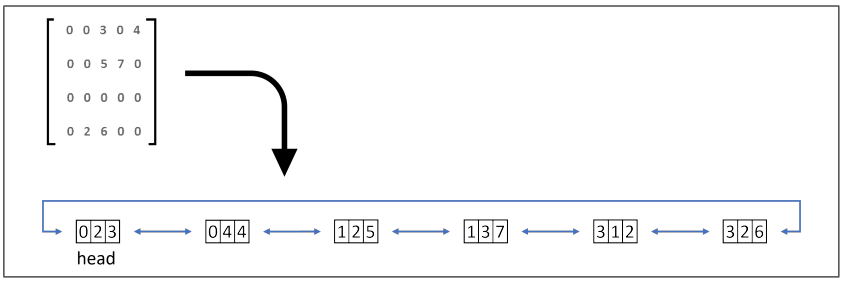
- https://sites.google.com/view/rrb-codebits/command-line-arguments
- https://docs.google.com/presentation/d/1rU0DhBg6yVXbfEtNuK73pjc1yWSPG9OYnGNH-b-kklQ/edit?usp=sharing

2) BASIC MATRIX OPERATIONS

To ensure the understanding of the previous tools and concepts, you will develop a small program that implements the basic matrix operations of addition, scalar multiplication, and transposition in sparse matrices. For this you will use a circular double linked list that stores the indices of the non-zero values along with the value itself. You can use the following diagrams to visualize the data structure.



Node structure



Matrix example

First, your program will compute the addition of two random square matrices and print both the matrices and the result on the standard output. The size of these matrices will be given on the command-line with the switch `-n`.

Secondly, your program will compute the scalar multiplication of a random square matrix and scalar given on the command-line with with switch `-s`. The random matrix will have the same size as the one on the first step. Print both the scalar, the random matrix and the resulting matrix on the standard output.

Finally, your program will compute the transpose of a random matrix and print the original and the result on the standard output. The random matrix will have the same size as the one on the first step.
Your program will execute the 3 operations in sequence on each run, as shown in the example.

3) EXAMPLE

```
user@dijkstra$ ./hw0 -n 3 -s 10

# addition of two 3x3 random matrices
A:
  1   2   3
  4   5   6
  7   8   9
B:
  11  12  13
  14  15  16
  17  18  19
C:
  12  14  16
  18  20  22
  24  26  28

# scalar multiplication of 3x3 matrix by 10
s: 10
A:
  9   8   7
  6   5   4
  3   2   1
C:
  90  80  70
  60  50  40
  30  20  10

# transposition of a random matrix
A:
  1   11  111
  2   22  222
  3   33  333
A.T:
  1   2   3
  11  22  33
  111 222 333
```

```
user@dijkstra$ ./hw0 -n 2 -s 2

# addition of two 2x2 random matrices
A:
  10   20
  3   7
B:
  1 4
  14  15
C:
  11    24
  17    22

# scalar multiplication of 2x2 matrix by 2
s: 2
A:
  2   3
  5   7
C:
  4  6
  10  14

# transposition of a random matrix
A:
  -4   -2
  4   2
A.T:
  -4   4
  -2  2
```

.