2017/18 Advanced Algorithms Project

Alexandre Francisco and Luís Russo

Last modified: March 5, 2018

1. This project considers a simplified version of the problem of inferring phylogenies. Students should deliver working implementations, along with a report including the experimental setup, time and space analysis, both theoretical and experimental, and appropriate references, for the algorithms proposed and implemented. Students should not use existing code for the algorithms described in the project, either from software libraries or other electronic sources. The deadline for the submission of both project code and report is May 18th, at 17:00. It is important to read the full description of the project before starting to design and implement solutions.

2. Let S be a set of $n > 0$ distinct binary strings, each one with length $m > 0$, representing SNP (single-nucleotide polymorphism) profiles and k an integer such that $0 < k \leq m$. The objective is to find the optimal phylogeny as follows:

i. Construct the graph $G = (V, E)$ where each node corresponds to a string, i.e., $V = S$, and where there is a link between any pair of strings $u, v \in S$ if $d(u, v) \leq k$, i.e., $(u, v) \in E$ if $d(u, v) \leq k$, where d is the Hamming distance.

ii. Assuming that each string $u \in S$ is identified by its occurring position i_u in the input, define the total order $\leq$ on E such that $(u_1, v_1) \leq (u_2, v_2)$ if and only if:

$$\begin{aligned} &d(u_1, v_1) < d(u_2, v_2) \vee \\ &\Big(d(u_1, v_1) = d(u_2, v_2) \wedge \Big(\min\{i_{u_1}, i_{v_1}\} < \min\{i_{u_2}, i_{v_2}\} \vee \\ &\qquad \Big(\min\{i_{u_1}, i_{v_1}\} = \min\{i_{u_2}, i_{v_2}\} \ \wedge \ \max\{i_{u_1}, i_{v_1}\} \leq \max\{i_{u_2}, i_{v_2}\} \Big) \Big) \Big) \end{aligned}$$

for $(u_1, v_1), (u_2, v_2) \in E$. Note that for real phylogenetic analysis the total order relies on more complex criteria, this is a simple version just for this project.

iii. Find the spanning forest on G that minimizes $\leq$.

3. The last step is equivalent to find the optimal basis of a graphic matroid, for which we can use several algorithms such as Prim's algorithm, Kruskal's algorithm or Borůvka's algorithm. We recommend the book "Combinatorial Optimization: Algorithms and Complexity", by C. H. Papadimitriou and K. Steiglitz, published by Dover in 1998, for more details on graphic matroids and related optimization problems.

4. A simple implementation for the approach described in 2 is to compute the distance among all pairs of strings (u, v), filter and keep those such that $d(u, v) \leq k$, and use Prim's algorithm (with a priority queue based on binary heaps) or Kruskal's algorithm. Note that an implementation relying on Kruskal's algorithm may require more space than one relying on Prim's algorithm. Such an implementation, including its complexity analysis and experimental evaluation, will be graded at most 12 points out of 20, with 10 points being assigned automatically on project submission. Higher grades will be given as follows:

i. The use of indexing and approximate string matching techniques to avoid the computation of the distance among all pairs of strings for small values of k, including the comparison with the naive approach and the analysis of the benefit of such approach depending on the value of k, will be graded with up to 4 more points.

ii. The use of Prim's algorithm using a priority queue based on more advanced heap data structures, including its analysis and comparison with binary heaps, will be graded with up to 4 more points.

5. To automatically validate the implementation we use the following conventions. Your implementation should generate a binary that is executed with the following command:

```
./project < in > out
```

The file `in` contains the input and the output is stored in the file named `out`. The input and output must respect the specification in 6 and 7 precisely. The output file will be validated against an expected result, stored in a file named `check`, with the following command:

```
diff out check
```

This command should produce no output, thus indicating that both files are identical.

6. The format of the input file is the following. The first line contains a single integer, n, that indicates the number of binary strings that follow. Each of the following n lines contains then a binary string, i.e., a string over the alphabet {`0`, `1`}. A final line contains a single integer, k, that indicates the distance threshold to be considered. The input contains no more data.

7. The output should consist of the list of links in the found optimal forest, one link per line. Each link (u, v) should be written as two integers, separated by a single space, where the first integer is given by $\min(i_u, i_v)$ and the second integer is given by $\max(i_u, i_v)$. The lines must be sorted numerically by the first integer and, then, by the second integer.

8. The following example shows the expected output for the given input. Input:

```
6
0000000
0001000
1000000
1011000
0000111
1000111
2
```

Output:

```
1 2
1 3
2 4
5 6
```

9. As stated in 4, up to 10 points will be assigned automatically on project submission by the mooshak system. The mooshak system accepts implementations in both C or Python. Projects that do not compile or run in the mooshak system will be graded 0. Only the code that compiles in the mooshak system will be considered; commented code, or including code in the report will not be considered for evaluation. Submissions to the mooshak system should consist of a `.zip` archive, which must include a `Makefile`. We will compile the submitted code by extracting the archive and running

```
make -s all
```

The compilation process should produce absolutely no errors or warnings. An executable named `project` must be generated and it should behave exactly as explained in the specification above. Be mindful that `diff` will produce output even if a single character is different, such as a space or a newline.

10. The report should be delivered in the fenix web page, in PDF format with at most 10 pages A4, fonts of at least 11pt and margins of at least 3cm. The report should contain a brief introduction, any relevant implementation decisions, extensive experimental results for the algorithms, including performance graphs for all the algorithms and comparison to the projected theoretical bounds, using several families of inputs, measuring both time and space, and also some brief conclusions. Reports that are not presented in PDF, will be graded 0. Reports that are not delivered in fenix by the designated deadline will be graded 0. Notice that you can submit the report to fenix several times, you are strongly advised to submit several times and as early as possible. The same happens in the mooshak system, with the same advice. Only the last version is considered for grading purposes, all other submissions are ignored. There will be **no** deadline extensions. Submissions by email will **not** be accepted.