**21/12/2020**

**Substring matching with application to genomics/proteomics**

**Report by:**

**Mladen Jurkic – Chiara Balduzzi**

**Preliminary analysis of the algorithm:**

The aim of the developed algorithm is to find a group of characters (called ‘pattern’) among a larger set (indicated as ‘sequence’). Both of the sets are only composed out of 4 elementary characters which refer to a nitrogenous base that could be found inside a nucleic acid chain (i.e.: DNA): adenine (A), thymine T, guanosine G, cytosine C (controllare sta cazzata che ho scritto).

In our program, the comparison between pattern and sequence can be made in both forward and backward directions. Assuming the length of the sequence and of the pattern to be, respectively, ‘slen’ and ‘plen’, we can have a total or partial overlapping. In this latter case, a match is found when the sub-sequence and pattern are equal up to a limited number of contiguous errors (denoted as ‘holes’ – at most 10% of plen) having a maximum allowed width, which in our implementations is set to 0.1% of the length plen.   
In either case it is possible to get the positions of the sequence at which the correspondence was found. Obviously, by adding the possibility to take holes into account we increase complexity and overall computational time of the program.

A first idea to solve the given problem, could be to use the built-in C function strcmp: int strcmp(const char \*sub\_sequence, const char \*pattern).   
Assume the length of the sequence and of the pattern to be, respectively, ‘slen’ and ‘plen’. This function can compare the pattern with a sub-string of the sequence having length ‘plen’. By selecting with proper indices the sub-string of the sequence and ending the loop when we reach ‘slen-plen’, it’s possible to obtain the number of perfect correspondences. However, we abandon this solution favouring a “manual” approach for two main reasons. First of all, by using the built-in function we cannot appreciate the effect of the parallelization as the serial code is already optimized and very fast. Furthermore, the need to include partial overlaps makes this solution not useful for us.

To implement the aforementioned solution we had to compare single characters of the sequence, one at a time: in case of a match of the first character, we proceed in the comparison of the successive ones. Otherwise, we simply increment a counter variable, shifting the beginning of the sequence and de facto comparing the pattern with a new sub-sequence.

**A-priori study of available parallelism:**

**…**

**OpenMP implementation:**

**…**

**Preliminary analysis of the algorithm:**

**…**

***Testing and debugging:***

***24 cores***

***10000 – 100***

Parallel\_v1\_break.c MATCH TIME: 0.01158 TOTAL TIME: 0.01175

Parallel\_v2\_break.c MATCH TIME: 0.00313 TOTAL TIME: 0.00332

Parallel\_v1\_task.c MATCH TIME: 0.02067 TOTAL TIME: 0.02101

Parallel\_v2\_task.c MATCH TIME: 0.00464TOTAL TIME: 0.00482

***100000 – 1000***

Parallel\_v1\_break.c MATCH TIME: 0.00374 TOTAL TIME: 0.00490

Parallel\_v2\_break.c MATCH TIME: 0.03256 TOTAL TIME: 0.03361

Parallel\_v1\_task.c MATCH TIME: 0.00367 TOTAL TIME: 0.00476

Parallel\_v2\_task.c MATCH TIME: 0.03259 TOTAL TIME: 0.03374

***1000000 – 1000***

Parallel\_v1\_break.c MATCH TIME: 0.00440 TOTAL TIME: 0.01441

Parallel\_v2\_break.c MATCH TIME: 0.02786 TOTAL TIME: 0.03734

Parallel\_v1\_task.c MATCH TIME: 0.00433 TOTAL TIME: 0.01405

Parallel\_v2\_task.c MATCH TIME: 0.02836TOTAL TIME: 0.03809

***10000000 – 1000***

Parallel\_v1\_break.c MATCH TIME: 0.03672 TOTAL TIME: 0.13258

Parallel\_v2\_break.c MATCH TIME: 0.27386 TOTAL TIME: 0.36450

Parallel\_v1\_task.c MATCH TIME: 0.03651 TOTAL TIME: 0.12728

Parallel\_v2\_task.c MATCH TIME: 0.28042 TOTAL TIME: 0.37020

***10000000 – 10000***

Parallel\_v1\_break.c MATCH TIME: 0.88490 TOTAL TIME: 0.97405

Parallel\_v2\_break.c MATCH TIME: 5.31968 TOTAL TIME: 5.40984

Parallel\_v1\_task.c MATCH TIME: 0.88530 TOTAL TIME: 0.97600

Parallel\_v2\_task.c MATCH TIME: 5.34456TOTAL TIME: 5.43625

***100000000 – 10000***

Parallel\_v1\_break.c MATCH TIME: 9.20338 TOTAL TIME: 10.13168

Parallel\_v2\_break.c MATCH TIME: 76.10272 TOTAL TIME: 77.01095

Parallel\_v1\_task.c MATCH TIME: 9.20192 TOTAL TIME: 10.11473

Parallel\_v2\_task.c MATCH TIME: 75.49195 TOTAL TIME: 76.40364

10^8 – 10^5 24 cores nienteeee

***16 cores***

***10000 – 100***

Parallel\_v1\_break.c MATCH TIME: 0.00377 TOTAL TIME: 0.00399

Parallel\_v2\_break.c MATCH TIME: 0.00342 TOTAL TIME: 0.00375

Parallel\_v1\_task.c MATCH TIME: 0.00360 TOTAL TIME: 0.00384

Parallel\_v2\_task.c MATCH TIME: 0.00151 TOTAL TIME: 0.00184

***100000 – 1000***

Parallel\_v1\_break.c MATCH TIME: 0.00160 TOTAL TIME: 0.00288

Parallel\_v2\_break.c MATCH TIME: 0.00525 TOTAL TIME: 0.00660

Parallel\_v1\_task.c MATCH TIME: 0.00218 TOTAL TIME: 0.00339

Parallel\_v2\_task.c MATCH TIME: 0.00618 TOTAL TIME: 0.00751

***1000000 – 1000***

Parallel\_v1\_break.c MATCH TIME: 0.02231 TOTAL TIME: 0.03225

Parallel\_v2\_break.c MATCH TIME: 0.03559 TOTAL TIME: 0.04559

Parallel\_v1\_task.c MATCH TIME: 0.00558 TOTAL TIME: 0.01545

Parallel\_v2\_task.c MATCH TIME: 0.03377 TOTAL TIME: 0.04387

***10000000 – 1000***

Parallel\_v1\_break.c MATCH TIME: 0.05404 TOTAL TIME: 0.14792

Parallel\_v2\_break.c MATCH TIME: 0.36891 TOTAL TIME: 0.46191

Parallel\_v1\_task.c MATCH TIME: 0.05279 TOTAL TIME: 0.14558

Parallel\_v2\_task.c MATCH TIME: 0.34006 TOTAL TIME: 0.43185

***10000000 – 10000***

Parallel\_v1\_break.c MATCH TIME: 1.35811 TOTAL TIME: 1.45170

Parallel\_v2\_break.c MATCH TIME: 7.54932 TOTAL TIME: 7.65257

Parallel\_v1\_task.c MATCH TIME: 1.35024 TOTAL TIME: 1.44993

Parallel\_v2\_task.c MATCH TIME: 7.60344 TOTAL TIME: 7.69529

***100000000 – 10000***

Parallel\_v1\_break.c MATCH TIME: 13.60532 TOTAL TIME: 14.53676

Parallel\_v2\_break.c MATCH TIME: 65.85800 TOTAL TIME: 66.77868

Parallel\_v1\_task.c MATCH TIME: 13.62230 TOTAL TIME: 14.55495

Parallel\_v2\_task.c MATCH TIME: 70.89387 TOTAL TIME: 71.81061

***EMPTY TABLE***

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Core** | **Match time V1** | **Total time V1** | **Expected Speedup** | **Real S** | **Match Time V1 task** | **Total Time V1 task** | **Expected Speedup** | **Real**  **S** |
|  | 2 |  |  |  |  |  |  |  |  |
| **S 10^4** | 4 |  |  |  |  |  |  |  |  |
| **P 10^2** | 8 |  |  |  |  |  |  |  |  |
|  | 16 |  |  |  |  |  |  |  |  |
|  | 24 |  |  |  |  |  |  |  |  |
|  | **Core** | **Match time V1** | **Total time V1** | **Expected Speedup** | **Real S** | **Match Time V1 task** | **Total Time V1 task** | **Expected Speedup** |  |
|  | 2 |  |  |  |  |  |  |  |  |
| **S 10^5** | 4 |  |  |  |  |  |  |  |  |
| **P 10^3** | 8 |  |  |  |  |  |  |  |  |
|  | 16 |  |  |  |  |  |  |  |  |
|  | 24 |  |  |  |  |  |  |  |  |
|  | **Core** | **Match time V1** | **Total time V1** | **Expected Speedup** | **Real S** | **Match Time V1 task** | **Total Time V1 task** | **Expected Speedup** |  |
|  | 2 |  |  |  |  |  |  |  |  |
| **S 10^6** | 4 |  |  |  |  |  |  |  |  |
| **P 10^3** | 8 |  |  |  |  |  |  |  |  |
|  | 16 |  |  |  |  |  |  |  |  |
|  | 24 |  |  |  |  |  |  |  |  |
|  | **Core** | **Match time V1** | **Total time V1** | **Expected Speedup** | **Real S** | **Match Time V1 task** | **Total Time V1 task** | **Expected Speedup** |  |
|  | 2 |  |  |  |  |  |  |  |  |
| **S 10^7** | 4 |  |  |  |  |  |  |  |  |
| **P 10^3** | 8 |  |  |  |  |  |  |  |  |
|  | 16 |  |  |  |  |  |  |  |  |
|  | 24 |  |  |  |  |  |  |  |  |
|  | **Core** | **Match time V1** | **Total time V1** | **Expected Speedup** | **Real S** | **Match Time V1 task** | **Total Time V1 task** | **Expected Speedup** |  |
|  | 2 |  |  |  |  |  |  |  |  |
| **S 10^7** | 4 |  |  |  |  |  |  |  |  |
| **P 10^4** | 8 |  |  |  |  |  |  |  |  |
|  | 16 |  |  |  |  |  |  |  |  |
|  | 24 |  |  |  |  |  |  |  |  |
|  | **Core** | **Match time V1** | **Total time V1** | **Expected Speedup** | **Real S** | **Match Time V1 task** | **Total Time V1 task** | **Expected Speedup** |  |
|  | 2 |  |  |  |  |  |  |  |  |
| **S 10^8** | 4 |  |  |  |  |  |  |  |  |
| **P 10^4** | 8 |  |  |  |  |  |  |  |  |
|  | 16 |  |  |  |  |  |  |  |  |
|  | 24 |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Core** | **Match time V1** | **Total time V1** | **Expected Speedup** | **Real S** | **Match Time V1 task** | **Total Time V1 task** | **Expected Speedup** | **Real**  **S** |
|  | 2 | 0.00394 | 0.00436 |  |  | 0.00076 | 0.00127 |  |  |
| **S 10^4** | 4 | 0.00044 | 0.00074 |  |  | 0.00244 | 0.00275 |  |  |
| **P 10^2** | 8 | 0.00154 | 0.00190 |  |  | 0.00271 | 0.00314 |  |  |
|  | 16 | 0.00377 | 0.00399 |  |  | 0.00360 | 0.00384 |  |  |
|  | 24 | 0.01158 | 0.01175 |  |  | 0.02067 | 0.02101 |  |  |
|  | **Core** | **Match time V1** | **Total time V1** | **Expected Speedup** | **Real S** | **Match Time V1 task** | **Total Time V1 task** | **Expected Speedup** |  |
|  | 2 | 0.00379 | 0.00515 |  |  | 0.00833 | 0.00986 |  |  |
| **S 10^5** | 4 | 0.00224 | 0.00371 |  |  | 0.00963 | 0.01111 |  |  |
| **P 10^3** | 8 | 0.01183 | 0.01338 |  |  | 0.01042 | 0.01188 |  |  |
|  | 16 | 0.00160 | 0.00288 |  |  | 0.00218 | 0.00339 |  |  |
|  | 24 | 0.00374 | 0.00490 |  |  | 0.00367 | 0.00476 |  |  |
|  | **Core** | **Match time V1** | **Total time V1** | **Expected Speedup** | **Real S** | **Match Time V1 task** | **Total Time V1 task** | **Expected Speedup** |  |
|  | 2 | 0.03619 | 0.04642 |  |  | 0.04885 | 0.06056 |  |  |
| **S 10^6** | 4 | 0.02125 | 0.03421 |  |  | 0.06790 | 0.07921 |  |  |
| **P 10^3** | 8 | 0.01153 | 0.02391 |  |  | 0.06556 | 0.07670 |  |  |
|  | 16 | 0.02231 | 0.03225 |  |  | 0.00558 | 0.01545 |  |  |
|  | 24 | 0.00440 | 0.01441 |  |  | 0.00433 | 0.01405 |  |  |
|  | **Core** | **Match time V1** | **Total time V1** | **Expected Speedup** | **Real S** | **Match Time V1 task** | **Total Time V1 task** | **Expected Speedup** |  |
|  | 2 | 0.37222 | 0.46656 |  |  | 0.50192 | 0.61212 |  |  |
| **S 10^7** | 4 | 0.20925 | 0.32978 |  |  | 0.64497 | 0.75030 |  |  |
| **P 10^3** | 8 | 0.10835 | 0.23258 |  |  | 0.62880 | 0.73476 |  |  |
|  | 16 | 0.05404 | 0.14792 |  |  | 0.05279 | 0.14558 |  |  |
|  | 24 | 0.03672 | 0.13258 |  |  | 0.03651 | 0.12728 |  |  |
|  | **Core** | **Match time V1** | **Total time V1** | **Expected Speedup** | **Real S** | **Match Time V1 task** | **Total Time V1 task** | **Expected Speedup** |  |
|  | 2 | 10.15439 | 10.24702 |  |  | 14.22391 | 14.33278 |  |  |
| **S 10^7** | 4 | 5.64013 | 5.75996 |  |  | 17.81030 | 17.91615 |  |  |
| **P 10^4** | 8 | 2.87570 | 3.01167 |  |  | 17.76892 | 17.87497 |  |  |
|  | 16 | 1.35811 | 1.45170 |  |  | 1.35024 | 1.44993 |  |  |
|  | 24 | 0.88490 | 0.97405 |  |  | 0.88530 | 0.97600 |  |  |
|  | **Core** | **Match time V1** | **Total time V1** | **Expected Speedup** | **Real S** | **Match Time V1 task** | **Total Time V1 task** | **Expected Speedup** |  |
|  | 2 | 101.74417 | 102.76884 |  |  |  |  |  |  |
| **S 10^8** | 4 | 56.45102 | 57.65937 |  |  | 179.07820 | 180.13454 |  |  |
| **P 10^4** | 8 | 29.04837 | 30.23910 |  |  | 177.61403 | 178.70088 |  |  |
|  | 16 | 13.60532 | 14.53676 |  |  | 13.62230 | 14.55495 |  |  |
|  | 24 | 9.20338 | 10.13168 |  |  | 9.20192 | 10.11473 |  |  |