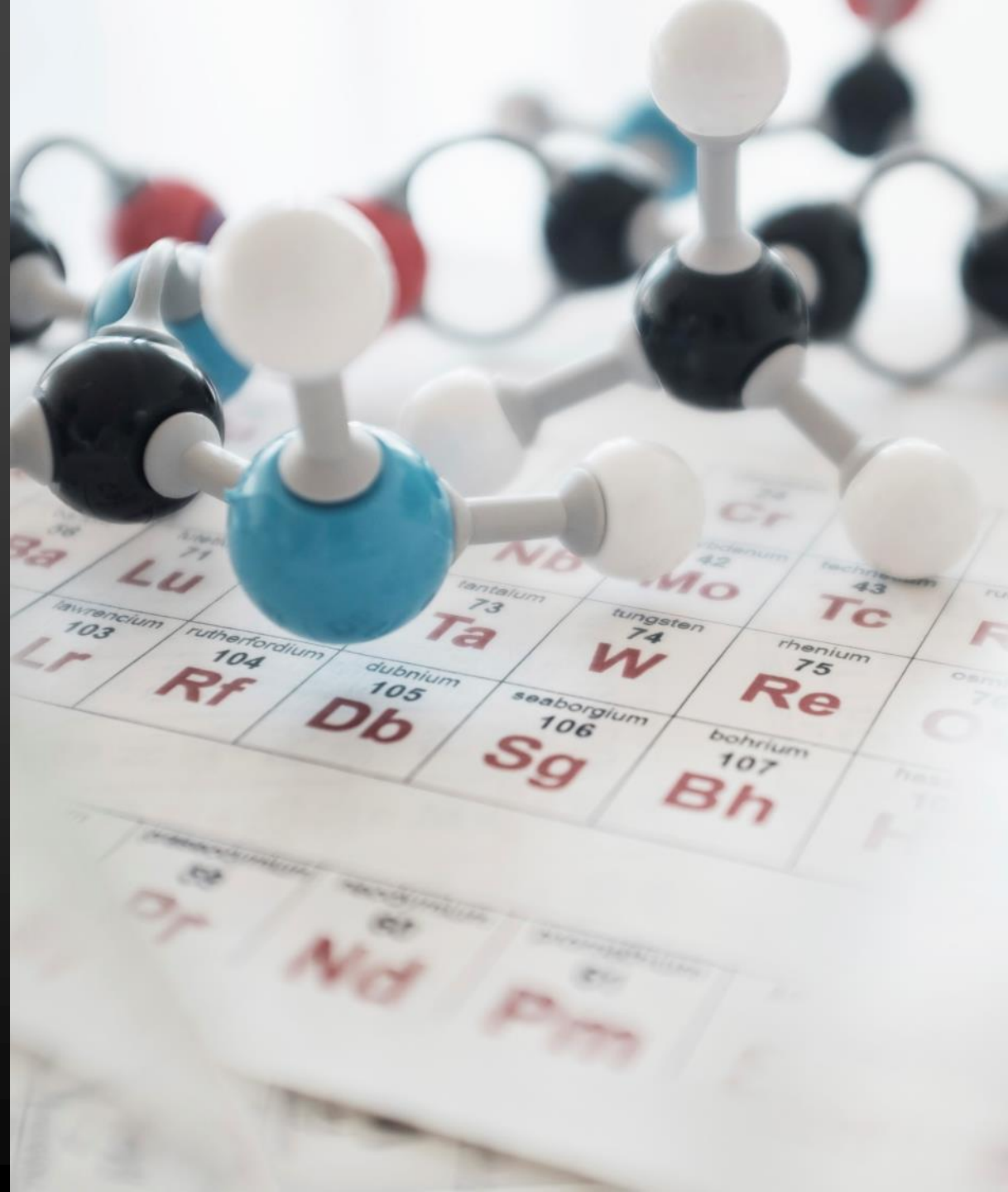


Implementazione algoritmo SMITH- WATERMAN

PRESENTAZIONE PROGETTO



		T	G	T	T	A	C	G	G
		0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

3	6	9	7	10	13
G	T	T	-	A	C
I	I	I		I	I
G	T	T	G	A	C

L'ALGORITMO

Qual è il suo scopo



Nel main prima del lancio del kernel

Cosa mi serve per l'implementazione

- Allocazione dinamica degli arrays.
 - Inserimento random dell'input
 - Dopo il `get_time()` di `start_cpu` e `end_cpu`:
`cudaMalloc` e `cudaMemcpy`
 - Quanti threads ho usato e considerazioni sul confronto tra le tempistiche della CPU e della GPU

```
// randomly generate sequences
for (int i = 0; i < N; i++)
{
    for (int j = 0; j < S_LEN; j++)
    {
        query[i][j] = alphabet[rand() % 5];


        Q[i*S_LEN+j]=query[i][j];
        reference[i][j] = alphabet[rand() % 5];
        R[i*S_LEN+j]=reference[i][j];
    }
}
```



Funzione inplinGpu

Parallelismo sulle antidiagonali

	0	C	A	G	C	C	U	C	G	C	U	U	A	G
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	5	0	0	0	0	0	0	0	?			
A	0	0	5	2	0	0	0	0	0	?				
U	0	0	0	2	0	0	5	0	?					
G	0	0	0	5	0	0	0	?						
C	0	5	0	0	10	5	?							
C	0	5	2	0	5	?								
A	0	0	10	1	?									
U	0	0	1	?										
U	0	0	?											
G	0	?												
C	0													
C	0													
G	0													
G	0													



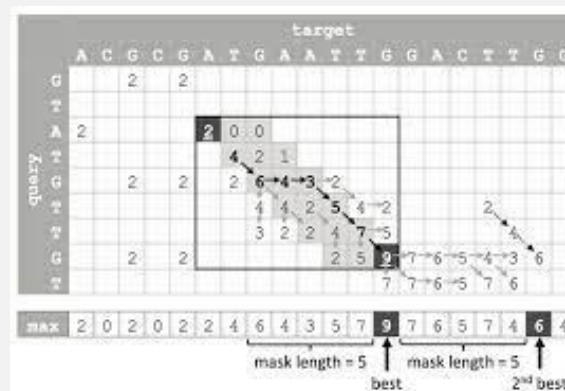
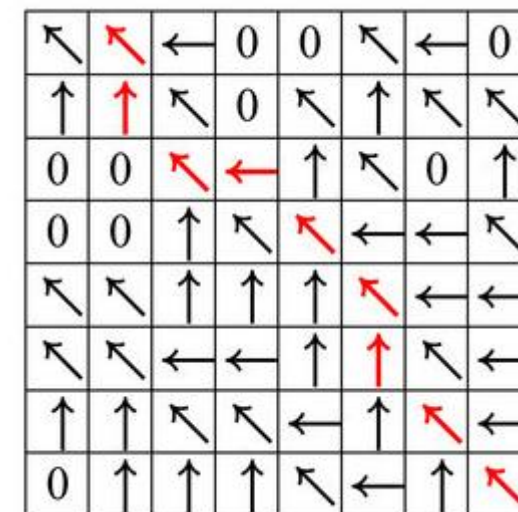
- Un blocco per ogni riga delle matrici di input
 - sc e dir inizializzate a 0
 - sc e dir una antidiagonale dopo l'altra iniziando dall'angolo superiore sinistro .
 - Gli arrays shared max e posizioni.
 - Il massimo di max e l'array RES

Funzione backtraceGpu

Riduzione a calcolo di sottomatrici

- L'array shared next
 - Il calcolo della sottomatrice
 - Il calcolo delle direzioni
 - L'iteratività
 - La compilazione di *gsimple_rev_cigar*

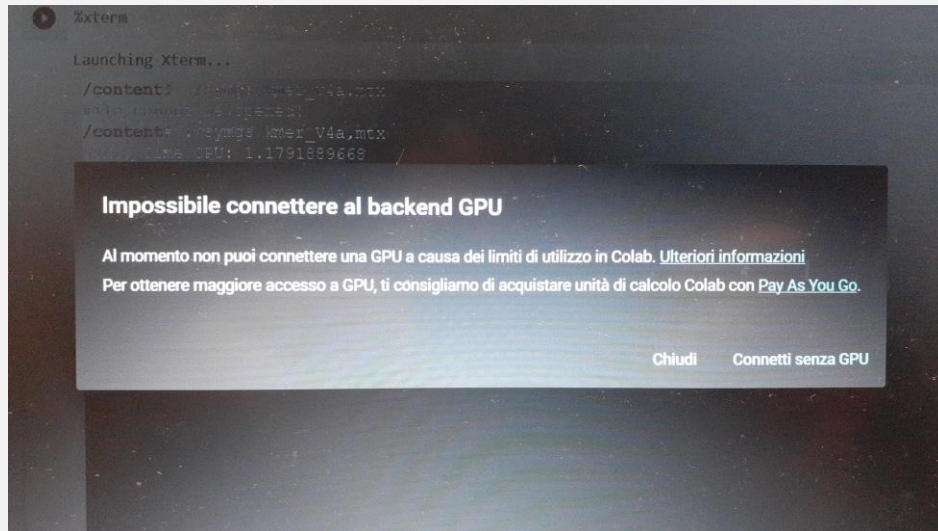
	0	A	C	G	T	A	T	G	C
0	0	0	0	0	0	0	0	0	0
A	0	2	0	0	0	2	0	0	0
C	0	0	4	2	1	0	1	0	2
G	0	0	2	6	4	3	2	3	1
A	0	2	1	4	5	6	4	3	2
A	0	2	1	3	3	7	5	4	3
C	0	2	4	2	2	5	6	4	6
C	0	0	2	3	1	4	4	5	6
C	0	0	2	1	2	3	3	3	7
T	0	0	0	1	3	2	5	3	5
T	0	0	0	0	3	2	4	4	4
G	0	0	0	2	1	2	2	6	4
C	0	0	2	0	1	0	1	4	8



Problemi affrontati e come li ho risolti

Colab I

- Limite di tempo per l'utilizzo della GPU



Colab 2

- Risultati dei calcoli

```
__shared__ int partenza;  
partenza=((maxind+1)%(513*513))-1;
```

Limite contenuti shared

- Ho risolto passando le matrici per indirizzo

Cose che non capivo: GPU edition

- Risolte come per i corsi del piano di studi

Verifica risultati

```
printf("SW Time GPU: %.10lf\n", end_gpu - start_gpu);

CHECK(cudaMemcpy(RES, gres, N * sizeof(int), cudaMemcpyDeviceToHost));

CHECK(cudaMemcpy(SRC, gsimple_rev_cigar, N*2*S_LEN* sizeof(char), cudaMemcpyDeviceToHost));
int t;
for( t=0;t<N && res[t]==RES[t];t++);
    if(t==N)
        printf("VERIFICA!\n");

for( int l=0;l<N;l++){
    for( t=0;SRC[512*2*l+t]!=0;t++){
        if(SRC[512*2*l+t]!=simple_rev_cigar[l][t])
            printf("LLL%d,%d,%d,%d\n",SRC[512*2*l+t],simple_rev_cigar[l][t],l,t);
    }
}
```





Chiara Fossà

✉ chiara.fossa@mail.polimi.it