%%cu

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <cuda\_runtime.h>

#define BLOCK\_SIZE 256

\_\_global\_\_ void compute\_c(int \*B, int n, int \*C)

{

int tid = blockIdx.x \* blockDim.x + threadIdx.x;

if (tid >= n) return;

int base = B[tid];

int sum = 0;

for (int i = 0; i < base; i++)

sum += C[i];

C[base] = sum + 1;

}

\_\_global\_\_ void compute\_occ(int \*B, int n, int \*C, int \*occ)

{

int tid = blockIdx.x \* blockDim.x + threadIdx.x;

if (tid >= n) return;

int base = B[tid];

int idx = C[base];

atomicAdd(&occ[idx], 1);

}

void compute\_fm\_index(int \*B, int n, int \*C, int \*occ, int \*occ\_sum)

{

// Compute C array

int num\_blocks = (n + BLOCK\_SIZE - 1) / BLOCK\_SIZE;

compute\_c<<<num\_blocks, BLOCK\_SIZE>>>(B, n, C);

// Compute occurrence array

memset(occ, 0, (n + 1) \* sizeof(int));

compute\_occ<<<num\_blocks, BLOCK\_SIZE>>>(B, n, C, occ);

// Compute occurrence sum array

occ\_sum[0] = 0;

for (int i = 1; i <= n; i++)

{

occ\_sum[i] = occ\_sum[i - 1] + occ[i - 1];

}

for(int i=0;i<n;i++)

{

printf("%d",occ\_sum[i]);

}

}

int main()

{

int \*B = (int\*)"ACGT";

int n = strlen((char\*)B);

int \*C, \*occ, \*occ\_sum;

cudaMalloc(&C, 5 \* sizeof(int));

cudaMalloc(&occ, (n + 1) \* sizeof(int));

cudaMalloc(&occ\_sum, (n + 1) \* sizeof(int));

compute\_fm\_index(B, n, C, occ, occ\_sum);

printf("Text:%s",(char\*)B);

cudaFree(C);

cudaFree(occ);

cudaFree(occ\_sum);

return 0;

}