%%cu

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <cuda.h>

#define MAX\_SEQ\_LENGTH 100000

\_\_global\_\_ void build\_fm\_index\_kernel(int\* fm\_index, char\* sequence, int seq\_length) {

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

    if (tid < seq\_length) {

        if (tid == 0) {

            fm\_index[0] = 0;

        } else {

            if (sequence[tid] != sequence[tid-1]) {

                fm\_index[tid] = fm\_index[tid-1] + 1;

            } else {

                fm\_index[tid] = fm\_index[tid-1];

            }

        }

    }

}

void build\_fm\_index(int\* fm\_index, char\* sequence, int seq\_length) {

    int\* dev\_fm\_index;

    char\* dev\_sequence;

    cudaMalloc((void\*\*) &dev\_fm\_index, sizeof(int) \* seq\_length);

    cudaMalloc((void\*\*) &dev\_sequence, sizeof(char) \* seq\_length);

    cudaMemcpy(dev\_sequence, sequence, sizeof(char) \* seq\_length, cudaMemcpyHostToDevice);

    int threads\_per\_block = 256;

    int num\_blocks = (seq\_length + threads\_per\_block - 1) / threads\_per\_block;

    build\_fm\_index\_kernel<<<num\_blocks, threads\_per\_block>>>(dev\_fm\_index, dev\_sequence, seq\_length);

    cudaMemcpy(fm\_index, dev\_fm\_index, sizeof(int) \* seq\_length, cudaMemcpyDeviceToHost);

    cudaFree(dev\_fm\_index);

    cudaFree(dev\_sequence);

}

int main() {

    char sequence[MAX\_SEQ\_LENGTH];

    fgets(sequence, MAX\_SEQ\_LENGTH, stdin);

    int seq\_length = strlen(sequence) - 1;

    int\* fm\_index = (int\*) malloc(sizeof(int) \* seq\_length);

    build\_fm\_index(fm\_index, sequence, seq\_length);

    for (int i = 0; i < seq\_length; i++) {

        printf("%d ", fm\_index[i]);

    }

    printf("\n");

    free(fm\_index);

    return 0;

}