Relazione del Progetto Laboratorio di ASD

Prima Parte

M. Giunta[1] S. Anzolin[2] F. Casani[3] G. De Nardi [4]

Giugno 2021

[1]Marco Giunta 147852 giunta.marco@spes.uniud.it
[2]Samuele Anzolin 142766 anzolin.samuele@spes.uniud.it
[3]Federico Casani 141212 casani.federico@spes.uniud.it
[4]Gianluca Giuseppe Maria De Nardi 142733 142733@spes.uniud.it

Indice

1 Introduzione

In questo progetto abbiamo implementato e analizzato due algoritmi per il calcolo del periodo frazionario minimo di una stringa. I due algoritmi implementati sono:

- *PeriodNaive*
- *PeriodSmart*

Il linguaggio di programmazione che abbiamo utilizzato per questo progetto è C, in quanto è un linguaggio veloce ed efficiente e qualitativamente migliore per un'analisi temporale.

2 Ipotesi

Essendo *periodNaive* un algoritmo con complessità asintotica nel caso peggiore pari a $O(n^2)$ mentre il secondo raggiunge una complessità lineare, possiamo dedurre che *periodSmart* sarà notevolmente migliore nel caso peggiore.

3 Implementazione

3.1 periodNaive

L'implementazione di questo algoritmo come dice il suo nome, è abbastanza intuitiva.
Utilizziamo quindi un ciclo che scandisca l'intera stringa di input e internamente controlliamo la congruenza tra il prefisso e il suffisso precedentemente calcolato aumentando la lunghezza del bordo minimo man mano che il ciclo e il controllo avanzano.

3.2 periodSmart

Questo algoritmo invece, sfrutta il concetto di bordo, cioè una qualsiasi stringa che sia, prefisso e suffisso proprio della stringa principale. Come osservato quindi **p** è un periodo frazionario di **s** solo se $p = |s| - r$ dove **r** è la lunghezza di un bordo di **s**, quindi il problema si riduce al solo calcolo del bordo massimo della stringa **s**.
Andiamo quindi ad analizzare l'implementazione vera e propria dell'algoritmo in questione, avremo quindi un algoritmo *searchLongestBorder* che calcola il bordo di lunghezza massima e un algoritmo *periodSmart* che sottrae semplicemente la lunghezza del bordo massimo alla lunghezza della stringa di input per ottenere il periodo frazionario minimo che stiamo cercando.

3.3 Algoritmo per il calcolo dei tempi medi

L'algoritmo per il calcolo dei tempi medi richiede di generare delle stringhe di lunghezza compresa tra 1000 e 500000 caratteri e appartenente ad un alfabeto ternario. Abbiamo utilizzato due metodi per la generazione delle stringhe:

- Nel primo caso i caratteri della stringa vengono generati in modo casuale
- Il secondo caso, invece, genera una porzione della stringa e la ripete per tutta la lunghezza totale

Abbiamo utilizzato la funzione *clock_ gettime* per ottenere il clock necessario al calcolo. La funzione richiede due parametri, il primo indica il tipo di clock, mentre il secondo contiene il puntatore di una *struct* contenente il tempo all'istante della chiamata a funzione. Il tipo di clock utilizzato è CLOCK_MONOTONIC che appunto come suggerisce il nome, è di tipo monotonico. Dovendo avere un errore relativo massimo di 0,001, dovremo reiterare fino a quando l'equazione:

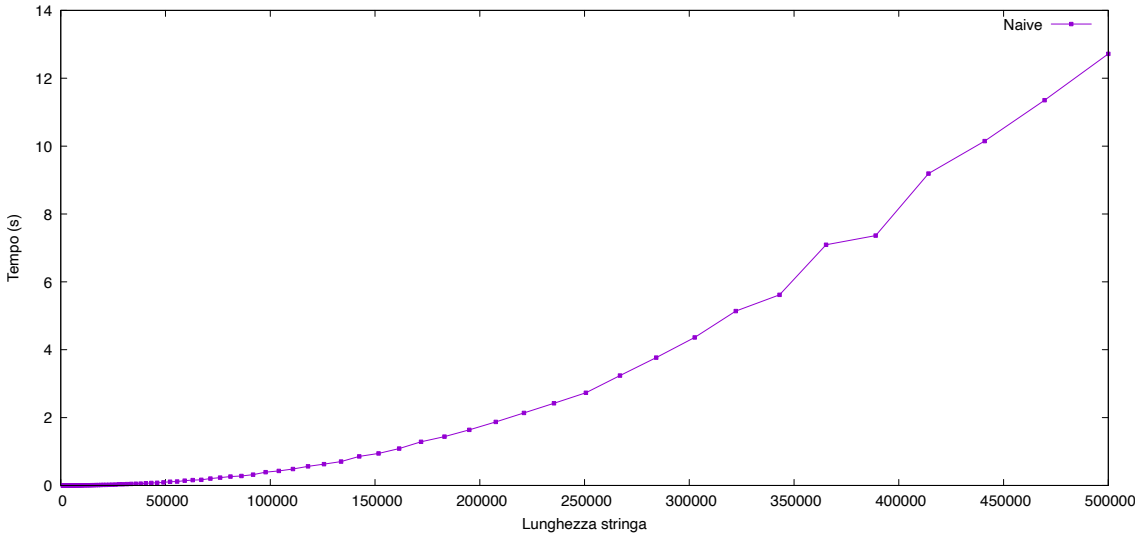
$$diff(Start, end) < Resolution * (\frac{1}{0,001} + 1)$$

non è più soddisfatta ed inseriremo i dati ottenuti all'interno dei grafici.

4 Analisi dei dati ottenuti

4.1 Grafico dei tempi di periodNaive

(a) Periodo non distribuito



(b) Periodo distribuito

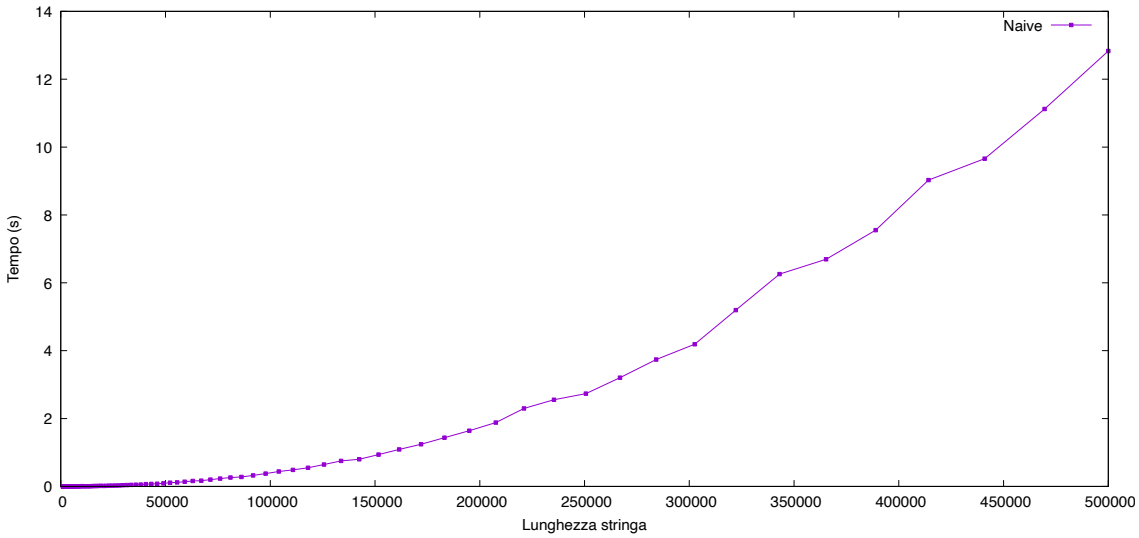
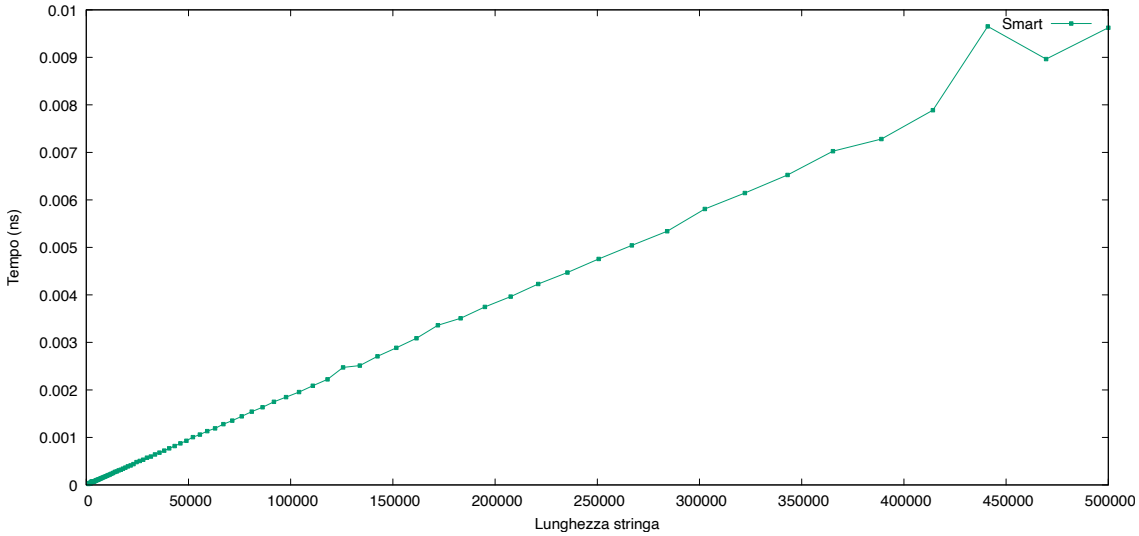


Figura 1: Tempi di calcolo del periodo con algoritmo naive

Come possiamo osservare dai due grafici (Figura 1a e Figura 1b), il tempo di esecuzione dell'algoritmo *periodNaive* impiega più di dieci secondi per stringhe con più di 450000 caratteri.

4.2 Grafico dei tempi di periodSmart

(a) Periodo non distribuito



(b) Periodo distribuito

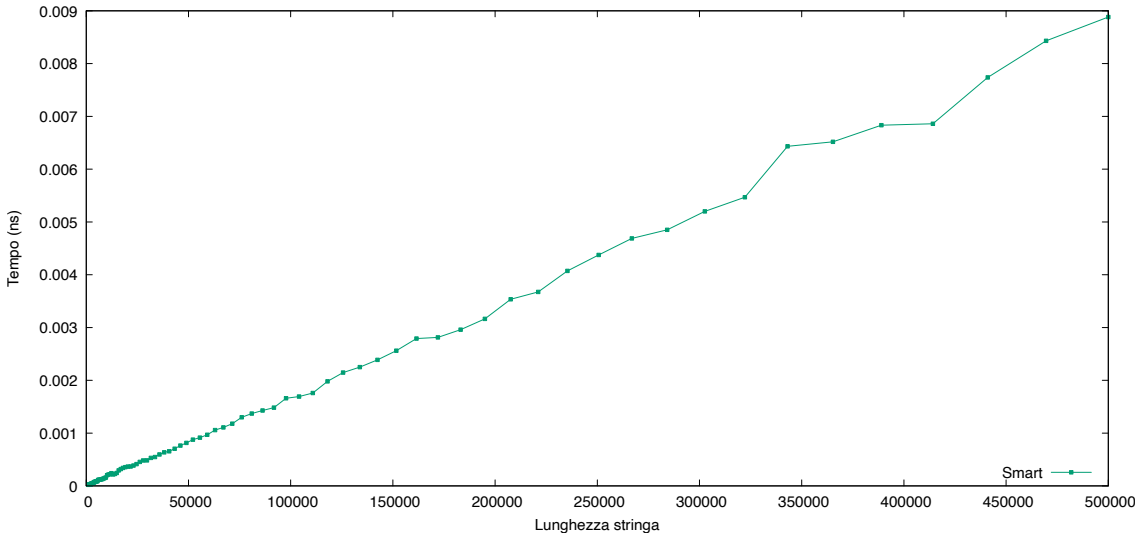
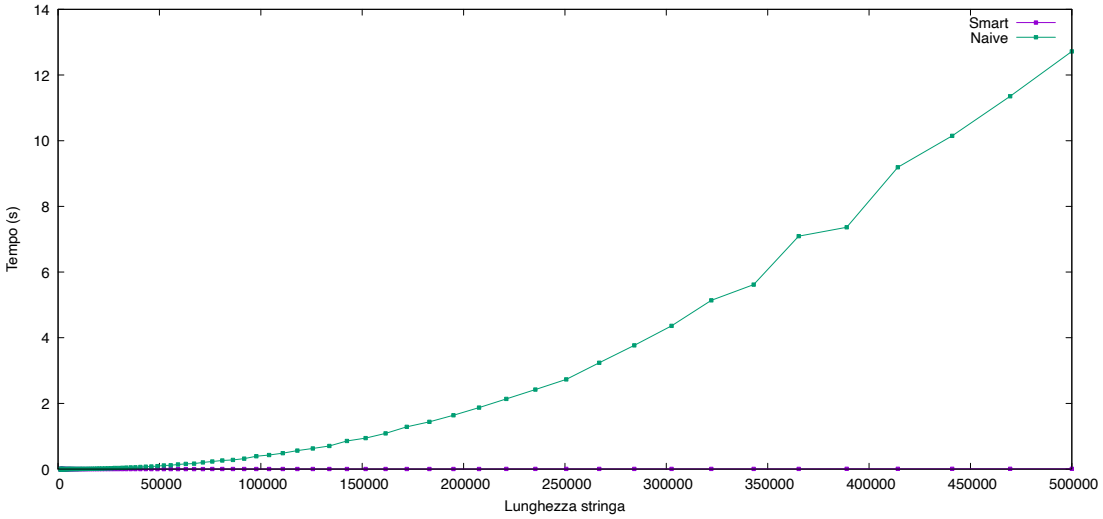


Figura 2: Tempi di calcolo del periodo con algoritmo smart

periodSmart, invece, impiega un tempo nettamente inferiore per stringhe con una grande quantità di caratteri, parliamo di nanosecondi.

4.3 Analisi logaritmica dei due algoritmi

(a) Periodo non distribuito



(b) Periodo distribuito

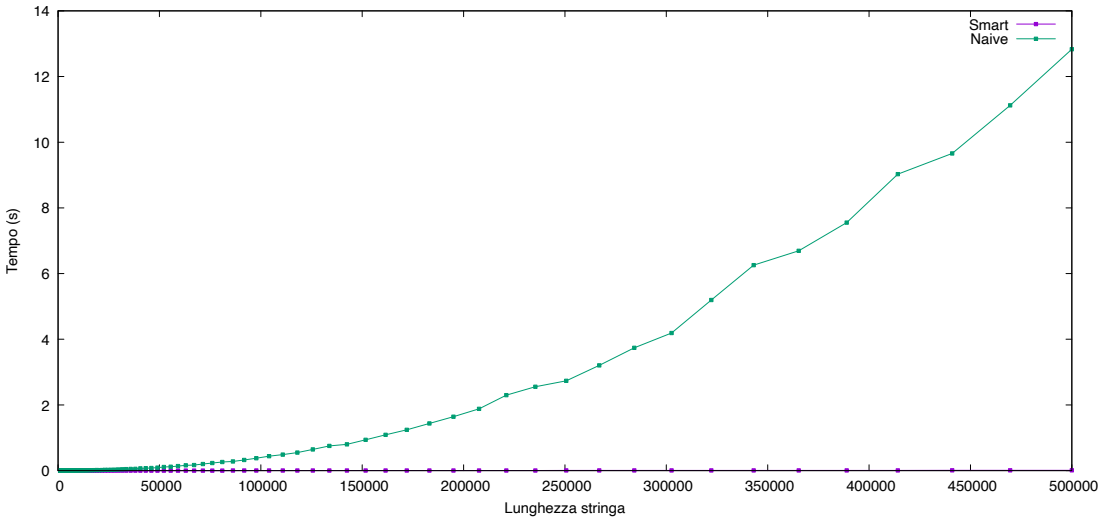
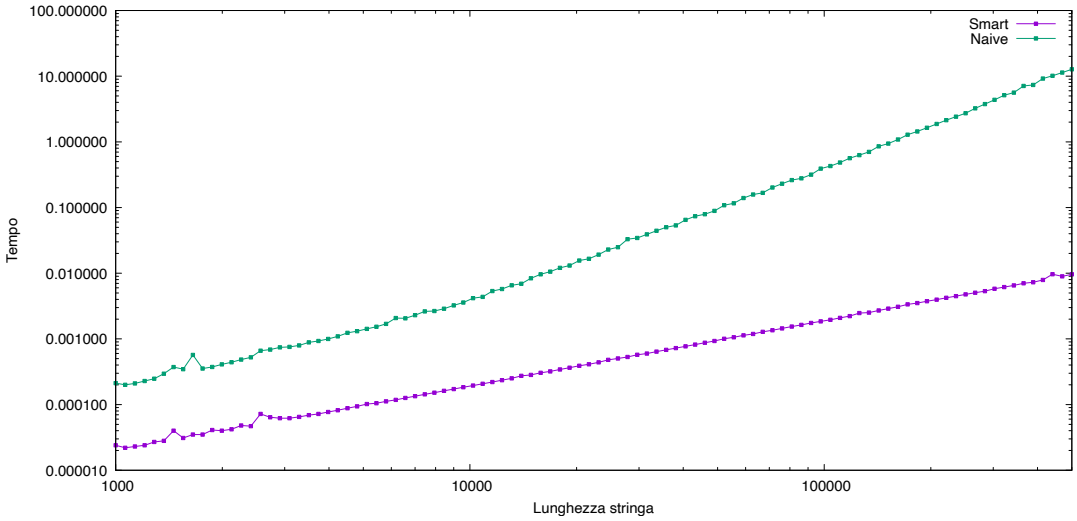


Figura 3: Confronto tempi di calcolo del periodo tra algoritmo smart e algoritmo naive

Non potendo analizzare i due algoritmi, visto che sono due scale temporali completamente differenti, dobbiamo quindi analizzarli in una scala logaritmica.

(a) Periodo non distribuito



(b) Periodo distribuito

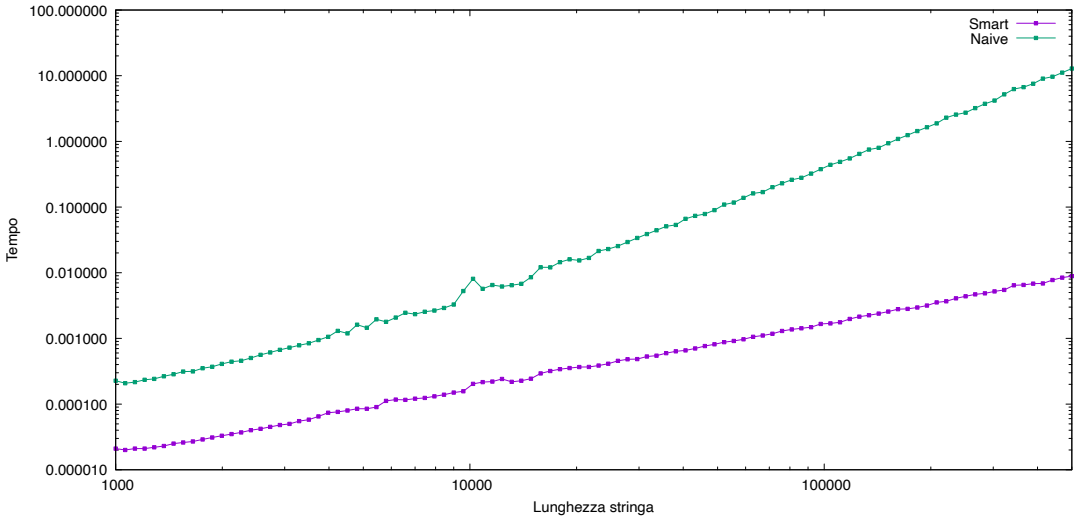
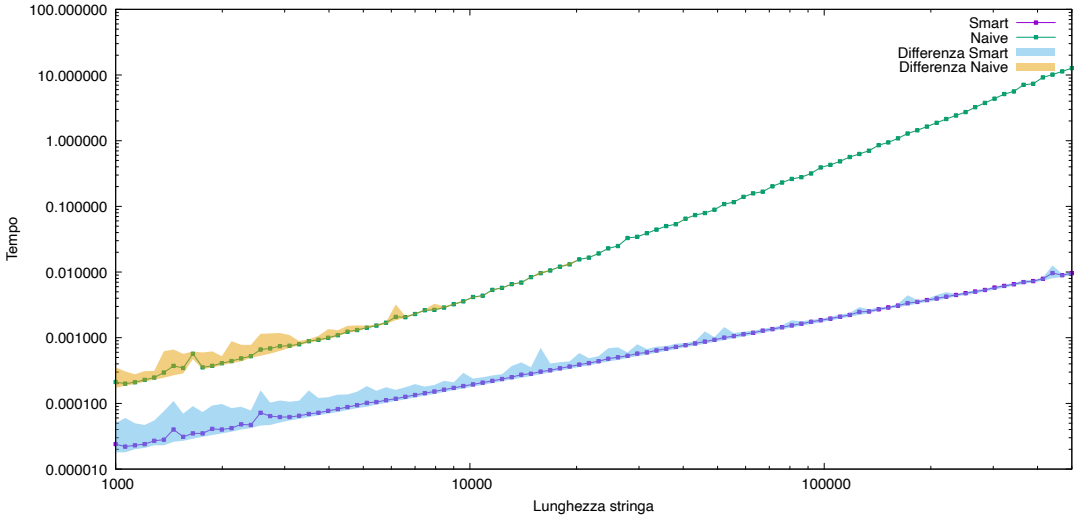


Figura 4: Confronto tempi di calcolo del periodo tra algoritmo smart e algoritmo naive

Non vi sono grossissime differenze con stringhe molto piccole, tuttavia più la stringa di input cresce, più la differenza di tempo tra i due algoritmi aumenta.

4.4 Analisi comparativa tra i due algoritmi

(a) Periodo non distribuito



(b) Periodo distribuito

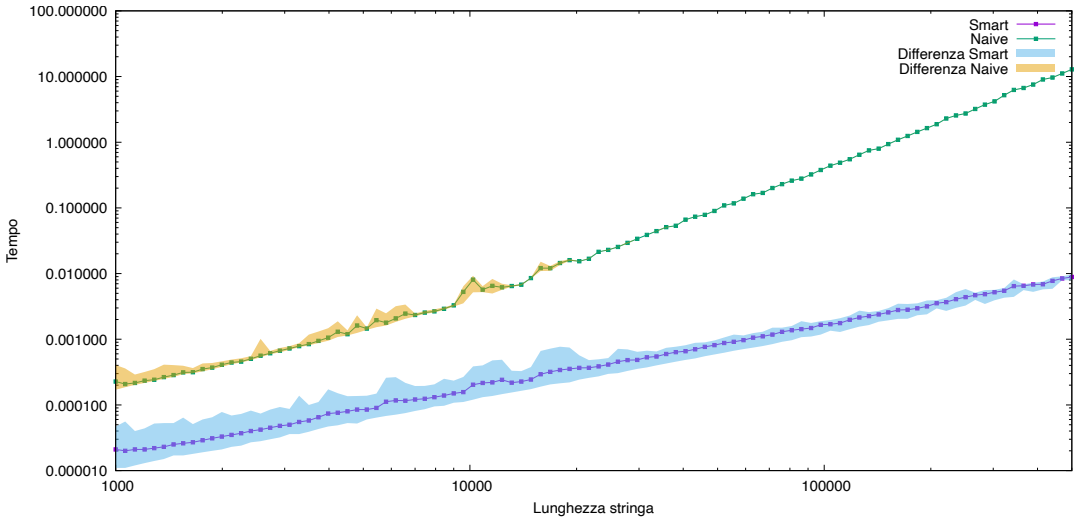


Figura 5: Confronto tempi di calcolo del periodo tra algoritmo smart e algoritmo naive

Dai grafici qui sopra possiamo notare come, l'errore relativo massimo, influenzi il numero di rilevazioni che andiamo a fare. Infatti, nel caso di *periodNaive* il numero di rilevazioni raccolte scende a uno dopo una certa soglia, mentre *periodSmart* continua a effettuare più di una rilevazione anche quando l'input raggiunge dimensioni considerevoli. Questo ad indicare che i tempi di *periodSmart* sono molto ridotti.

5 Conclusioni

Come osservato dai grafici, abbiamo sicuramente costatato che tra i due algoritmi c'è una notevole differenza in termini di tempo. L'argoritmo *periodSmart* impiega molto meno tempo dell'algoritmo *periodNaive* ed è quindi più consigliato il suo utilizzo.