UNIVERSITÀ DEGLI STUDI DI UDINE

Corso di Informatica e Internet of Things, Big Data and Web

Cognome e Nome	Numero Matricola
Mattia Guiotto	147654 147654@spes.uniud.it
Nicola Zerajic De Giorgio	149299 149299@spes.uniud.it
Mattia Novo	109538 109538@spes.uniud.it
Andrea Comisso	147984 147984@spes.uniud.it

Oggetto della prova

L'obiettivo primario di questo progetto è quello di mettere a confronto due algoritmi, entrambi fondati sulla ricerca del periodo frazionario minimo di una stringa, e di analizzare i loro tempi medi di esecuzione al fine di comprendere quale dei due algoritmi risulti il più efficace.

In questo progetto vengono presi in considerazione due algoritmi denominati Period Naive e Period Smart.

Descrizione della prova

Il periodo frazionario di una stringa di lunghezza n è il più piccolo intero p positivo che soddisfa la proprietà $\forall i = 1 \ldots n - p \quad s(i) = s(i + p)$

cioè il carattere in posizione i coincide col carattere in posizione $i + p$.

Esistono due algoritmi che calcolano il periodo frazionario di una stringa: uno con complessità quadratica $O(n^2)$ e l'altro con complessità lineare $O(n)$, quindi più efficiente.

Descrizione dell'algoritmo per la misurazione dei tempi

Per costruire i grafici con i tempi di risposta per i 2 algoritmi, sono state generate 100 stringhe (corrispondo a 100 punti nel grafico) con lunghezza compresa nell'intervallo [1000,500 000] caratteri. Le stringhe sono state salvate in vettore di puntatori a carattere. Per ogni indice del vettore è stato allocato dello spazio in memoria in modo dinamico per contenere esattamente la lunghezza della stringa generata. I valori di risposta per le 100 stringhe sono stati scritti in un file di testo mediante l'uso delle funzioni di libreria del C.

Le stringhe che sono state utilizzate per valutare i tempi di risposta nei 2 algoritmi sono state generate sull'alfabeto {a,b}. Le stringhe che abbiamo generato hanno la probabilità di avere un periodo frazionare vicino ad n. Per esempio, se ho una stringa con n=5000 caratteri, per avere un

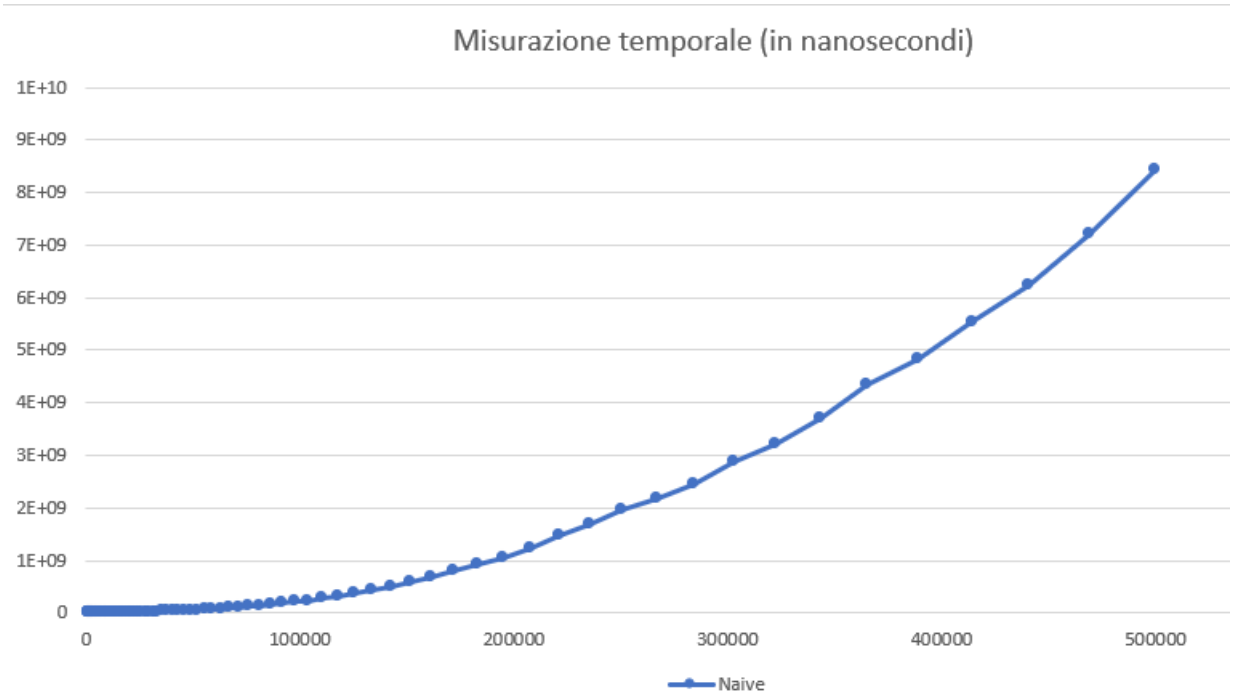
periodo frazionario vicino a n/2, la prima metà della stringa deve essere uguale alla seconda metà. Questo è molto difficile dato che ogni carattere della stringa ha la stessa probabilità di essere il carattere ‘a’ o ‘b’.

- ***ALGORITMO QUADRATICO : Period Naive***

  La procedura consiste nella semplice ricerca per tentativi del periodo p tramite il confronto tra il prefisso (da 1 a n-p) e il suffisso (da n-p+1 a n).

```
int periodNaive(char *s) {
    int len = strlen(s);
    for (int p = 1; p <= len; p++) {
        int dim = len - p;
        char s1[dim];
        strncpy(s1, s, len - p);
        s1[dim] = '\0';
        char s2[dim];
        strncpy(s2, s + p, len - p);
        s2[dim] = '\0';
        if (strcmp(s1, s2) == 0)
            return p;
    }
    return len;
}
```

  In base ai risultati ottenuti attraverso un algoritmo di calcolo dei tempi di elaborazione rispetto all’algoritmo appena descritto si è ottenuto il seguente grafico:



  Sulle ascisse è riportata la dimensione delle stringhe date in input e le ordinate indicano il tempo di elaborazione in nanosecondi.

  Si può notare che al crescere della dimensione delle stringhe il tempo di esecuzione cresce molto rapidamente fino ad arrivare a quasi 9 secondi.

- ***ALGORITMO LINEARE : Period Smart***

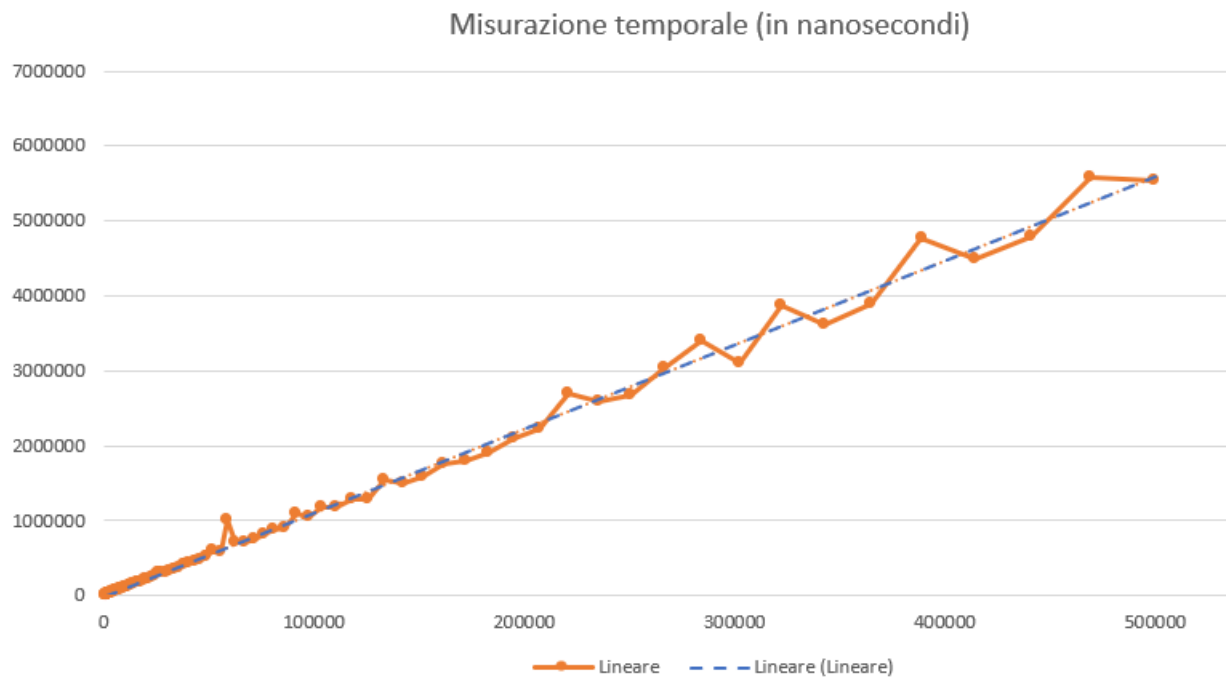
  L’algoritmo lineare si basa sul concetto di bordo di una stringa, più in particolare sul suo calcolo induttivo.

  Un bordo è una sottostringa che è contemporaneamente sia prefisso che suffisso della stringa di partenza.

  Una volta calcolato per induzione sui prefissi della stringa il bordo massimo (di lunghezza massima), il periodo frazionario corrisponde alla differenza fra la lunghezza della stringa e la lunghezza del bordo.

```
int periodLineare (char *s) {
  int n = strlen(s);
  int r[n];
  r[0]=0;
  for (int i=1; i < n; i++) {
      int z = r[i-1];
      while (s[i] != s[z] && z > 0) {
           z = r[z-1];
      }
      if (s[i] == s[z]) {
         r[i] = z + 1;
      } else {
        r[i] = 0;
      }
  }
   return n - r[n-1];
}
```

Il grafico ottenuto dai risultati delle misurazioni dei tempi è il seguente:



Il grafico si presenta come una retta, quindi la durata della procedura aumenta con la dimensione dell'input in modo costante.

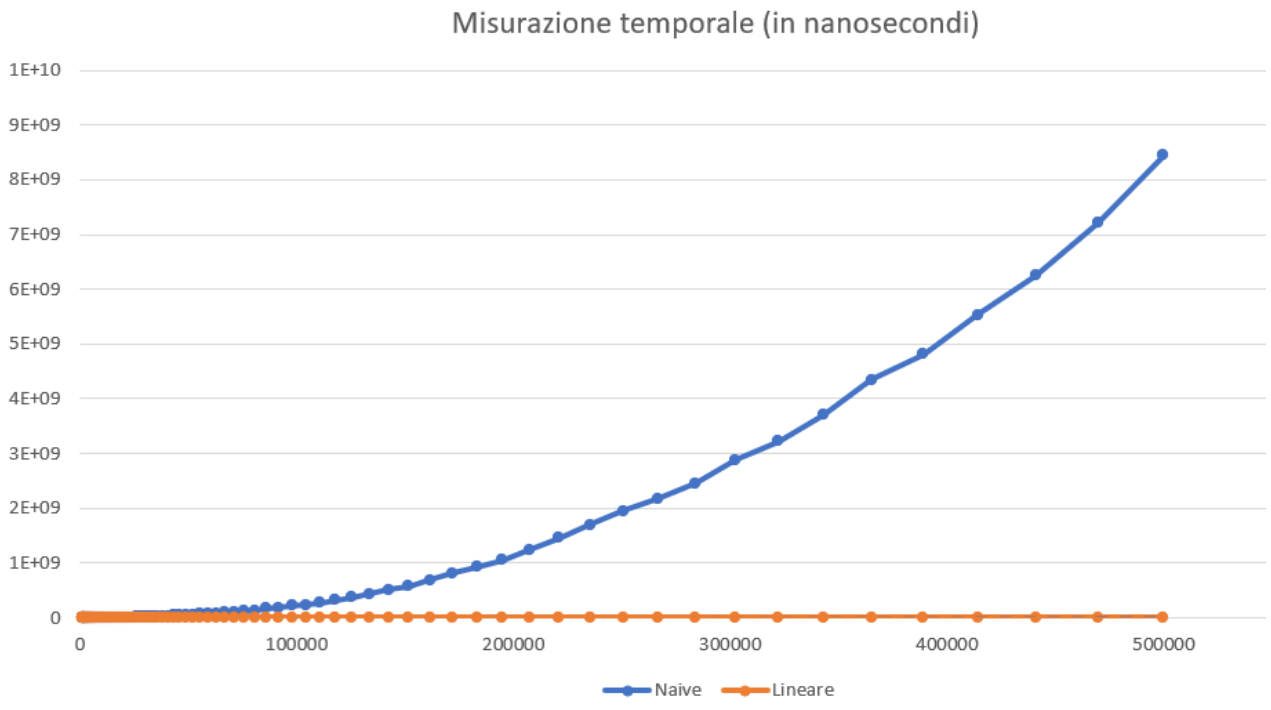
Analizzando i dati si osserva che l'algoritmo lineare è da 4 a 1500 volte più veloce rispetto all'algoritmo quadratico:

Dimensione input	Naive	Lineare	Rapporto
1000	50450	7969	6,33
1065	41866	9400	4,45
1134	46200	9254	4,99
1207	51650	10380	4,98
1285	55600	10830	5,13
1369	61150	12675	4,82

365307	4818022000	4767600	1010,58
388974	5530618200	4491800	1231,27
414174	6248035800	4786100	1305,45
441007	7212840500	5591300	1290,01
469578	8441686000	5542500	1523,08
500000	9478384400	6127400	1546,89

Confronto Period Naive - Period Smart:

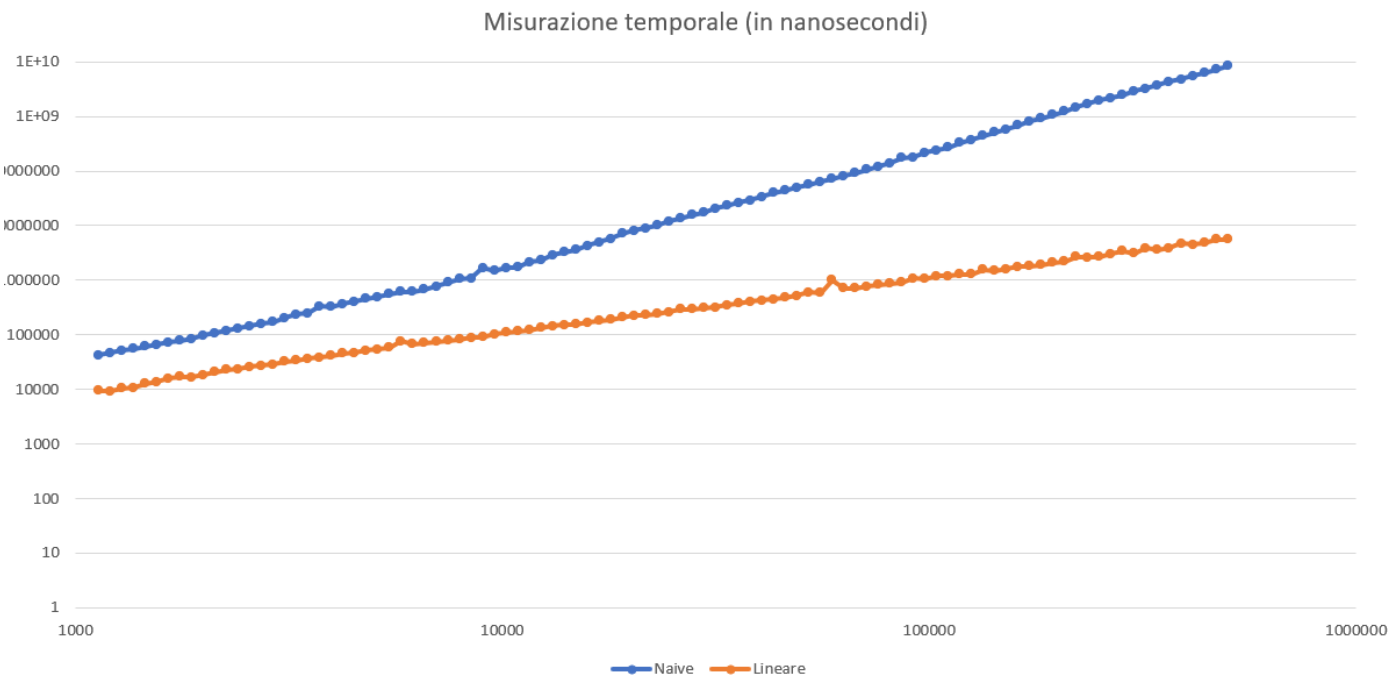
Mettendo insieme i due grafici si ottiene il seguente:



Da questo grafico risulta evidente la differenza, in termini di tempo, tra i due algoritmi.

Infatti il Period Naive, in quanto esponenziale, risulta più lento e inefficace del Period Smart avente caratteristica lineare.

A supporto del risultato ottenuto dal precedente grafico di confronto risulta utile analizzare anche il grafico in scala doppiamente logaritmica:



Si nota come la pendenza assunta dalle due rette sia differente.

Questa caratteristica è dettata dal diverso valore assunto dai coefficienti angolari, che corrispondono alla velocità di risposta del singolo algoritmo al crescere dei caratteri delle stringhe che vengono passate in input.

Il Period Smart risulta avere pendenza inferiore quindi è in grado di fornire l'output con tempi di risposta inferiore rispetto al Period Naive.

Descrizione dell'algoritmo per il calcolo della varianza

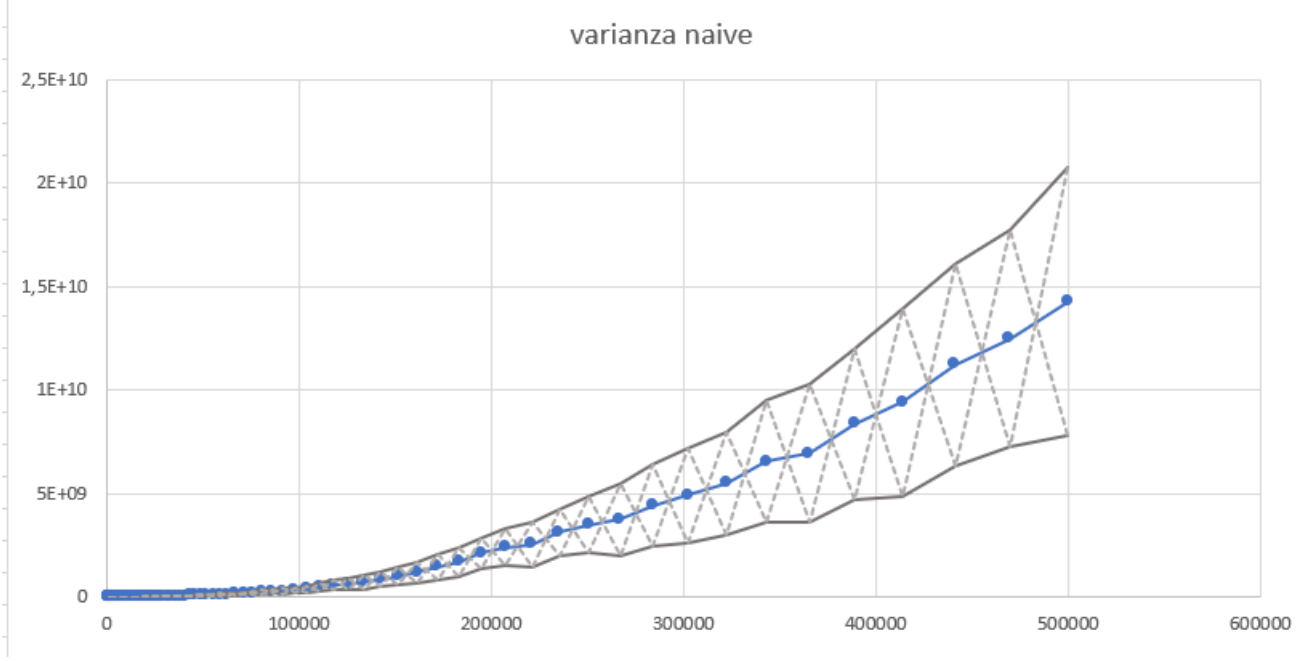
Nell'algoritmo per il calcolo della varianza abbiamo riservato una matrice per contenere 100 tempi risposta per ciascuna stringa generata. Avendo generato 100 stringhe diverse, per ognuna di esse, la misura del tempo di esecuzione nei 2 algoritmi è stato fatta per 100 volte. Queste 100 misure per ogni stringa, ci sono servite per il calcolo del tempo medio (somma delle 100 misure / 100) e per il calcolo della varianza.

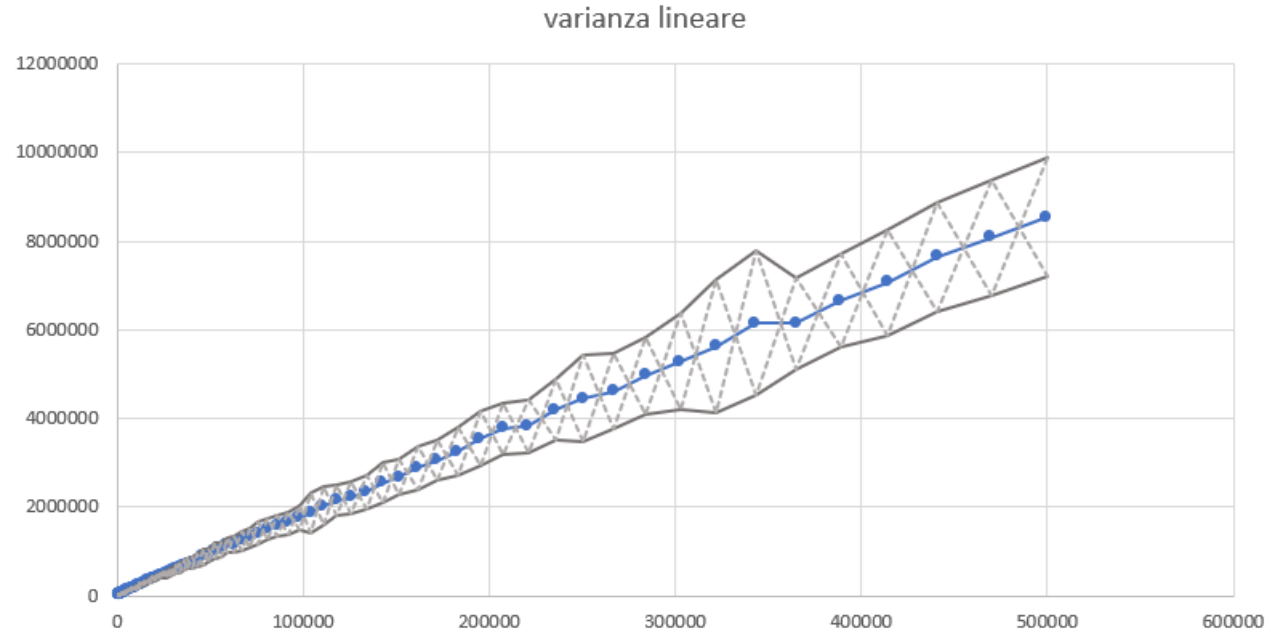
$$\sigma = \sqrt{\frac{1}{(N-1)} \sum_{i=1}^{N} (x_i - \bar{x})^2}$$

questa è la formula per il calcolo dello scarto quadratico medio, per ognuna delle 100 misure di ogni stringa si sottrae il tempo medio e si eleva tutto al quadrato. Si sommano tutti i 100 valori ottenuti e si divide per 99, infine tutto sotto radice. Per memorizzare il valore dello scarto quadratico medio per ogni stringa è stato utilizzato un array di long long.

Le lunghezze delle 100 stringhe, il tempo medio e lo scarto quadratico medio per ogni stringa sono stati salvati in un file di test mediante l'uso delle funzioni di libreria del C.

Lo scarto quadratico medio ci ha fornito la misura di quanto i 100 valori assunti da ciascuna stringa, si discostano rispetto alla media aritmetica dei 100 valori.





Dal grafico della varianza ottenuto con l’algoritmo Period Naive si evidenzia una divergenza dello scarto quadratico rispetto al valore medio sempre più ampia, che segue appunto un andamento quadratico, a conferma dell’estrema inefficienza dell’algoritmo naive per input di grandi dimensioni.

Al contrario la varianza dell’algoritmo Smart segue un andamento costante come la linea dei tempi medi. Ciò significa che col crescere dell’input il tempo di esecuzione non differirà pesantemente dalla media generale.