Progetto compressore

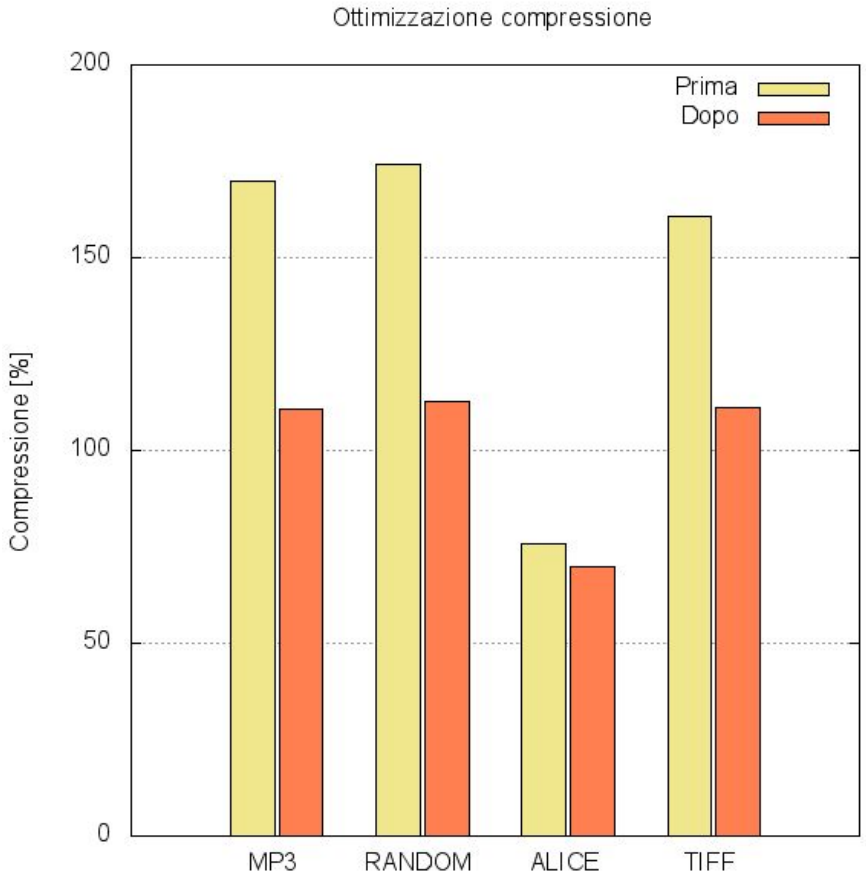
Kuhn - Paoliello

LZ77 - Kuhn

Flag bit

- Dizionario: 8kB
  - Offset: 13bit
- Look Ahead: 512B
  - Length: 9bit
- Totale
  - Match: 13 + 9 + 8 = 30bit
- 3 caratteri: 24bit!
  - Flag ‘0’: solo next char
  - Flag ‘1’: match completo

Flag bit


Ottimizzazione compressione
Prima
Dopo
Compressione [%]
200
150
100
50
0
MP3
RANDOM
ALICE
TIFF

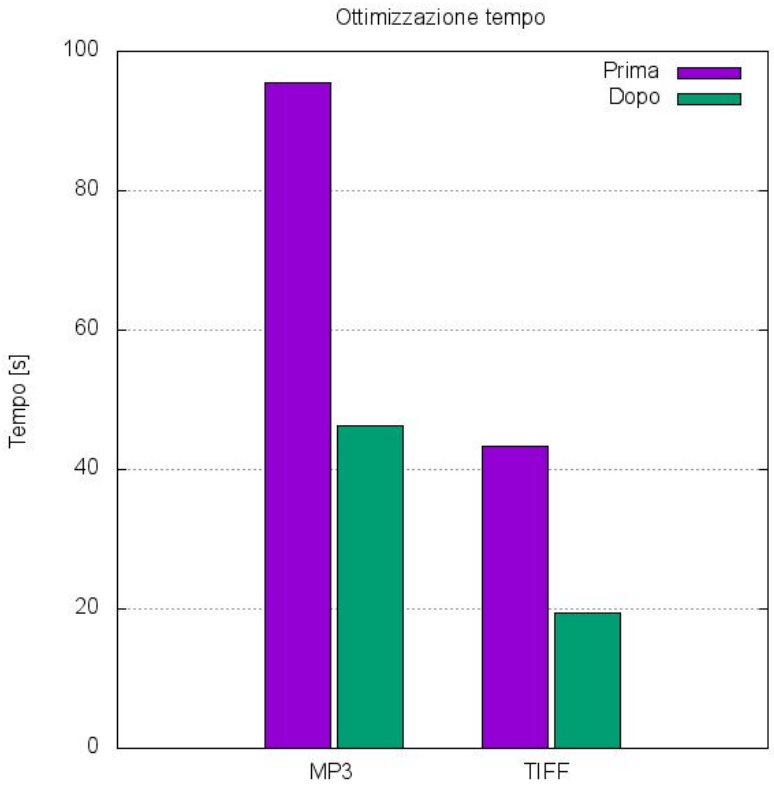
Knuth-Morris-Pratt

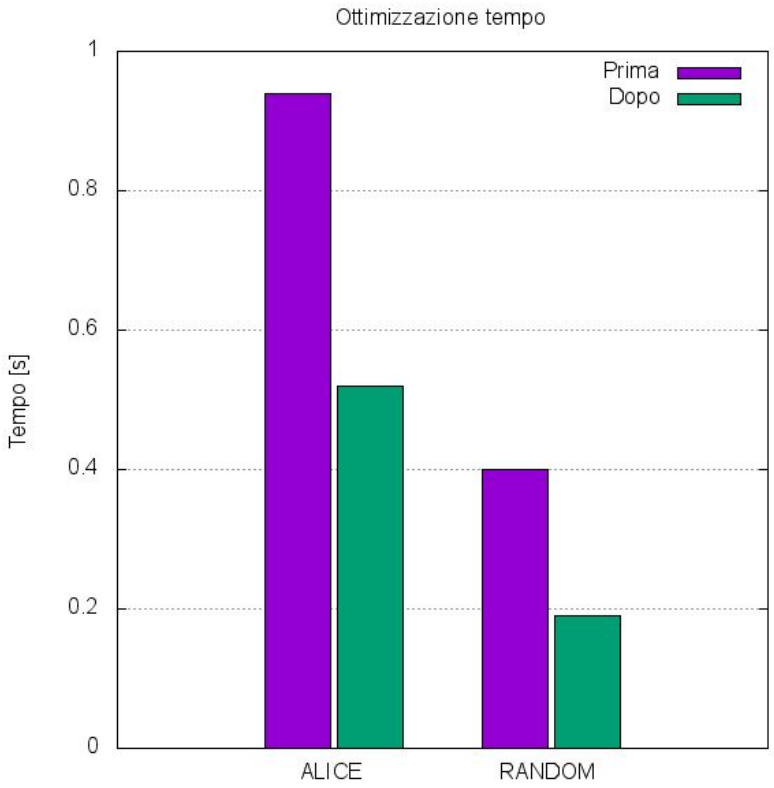
NAÏVE **vs** **KMP**

```
ANDARVIANDANDOALFIUME
ANDANDOALMARE
  ANDANDOALMARE
    ANDANDOALMARE
      ANDANDOALMARE
        ...
```

```
ANDARVIANDANDOALFIUME
ANDANDOALMARE
      ANDANDOALMARE
        ...
```

Knuth-Morris-Pratt

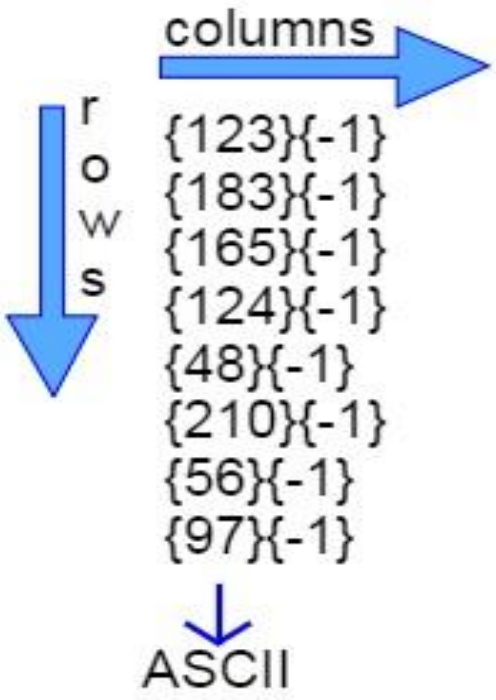




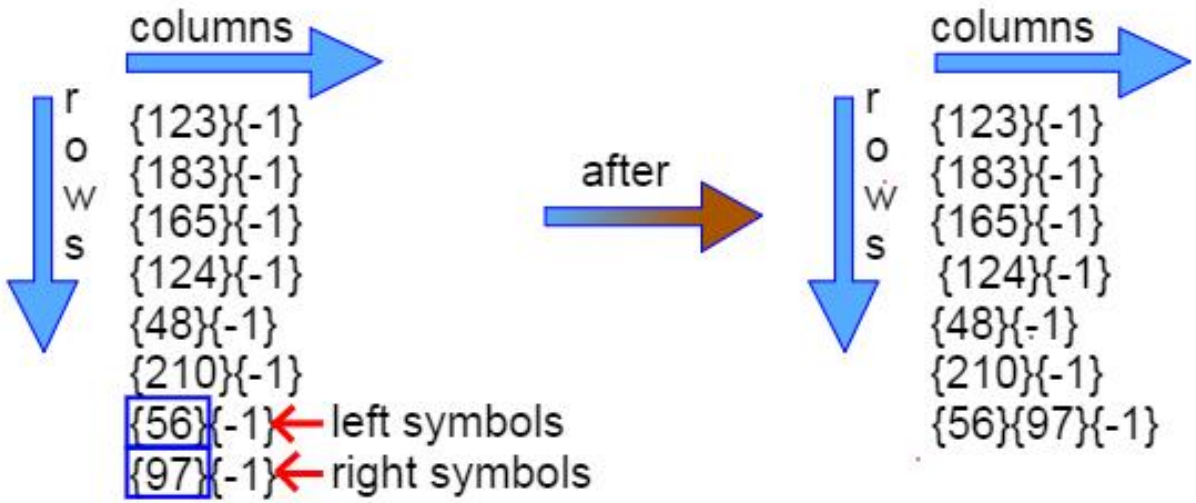
Huffman - Paoliello

Costruzione codifica

```
int **label=malloc(sizeof(int*)*differentChars);
for(int q=0;q<differentChars;q++)
    label[q]=malloc(sizeof(int)*2);
```



Costruzione codifica



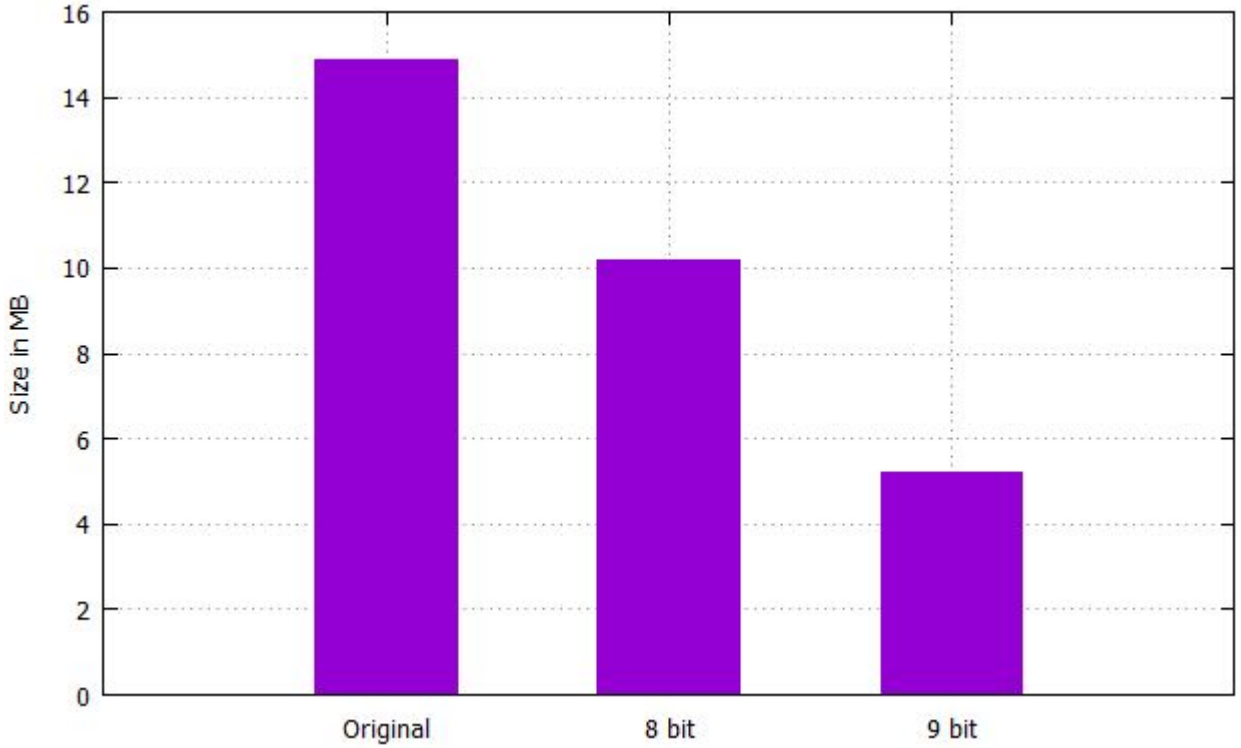

```
strcat(myEncoding[56].code, "0");

strcat(myEncoding[97].code, "1");
```

- Perchè il “bubble sort”?

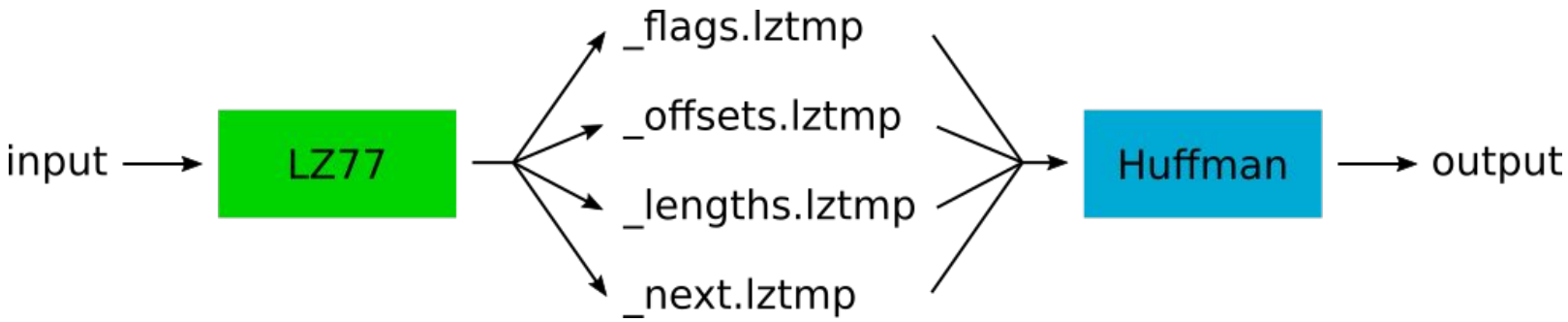
Huffman con lunghezza input generica

- Perchè vantaggiosa?
- Esempio concreto ottenuto comprimendo le lunghezze di pattern matching di LZ77 (9 bit):

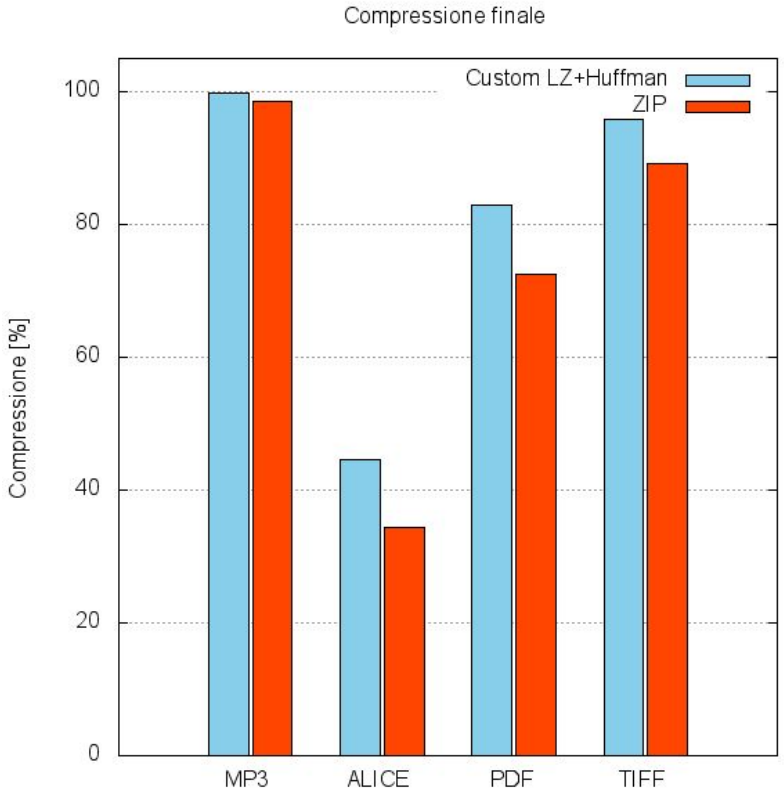


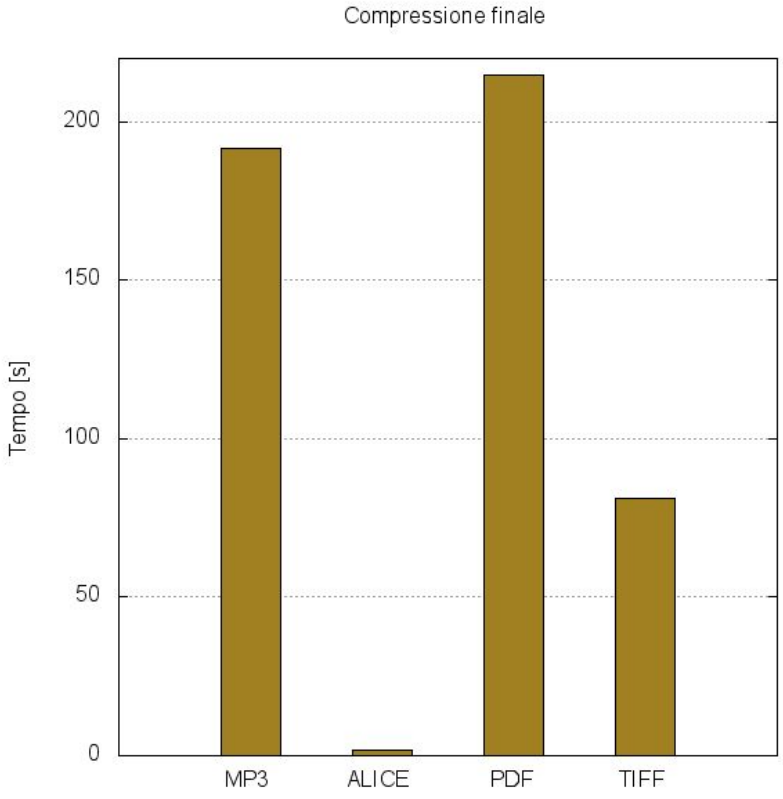
Custom LZ + Huffman

Funzionamento



Risultati


Compressione finale
Custom LZ+Huffman
ZIP
Compressione [%]
100
80
60
40
20
0
MP3
ALICE
PDF
TIFF


Compressione finale
Tempo [s]
200
150
100
50
0
MP3
ALICE
PDF
TIFF

180 CHF

-20% per collaboratori SUPSI!