RELAZIONE ALGORITMO LVQUICKSORT

Lavoro svolto da:
-Magno Alessandro : 4478234

Descrizione

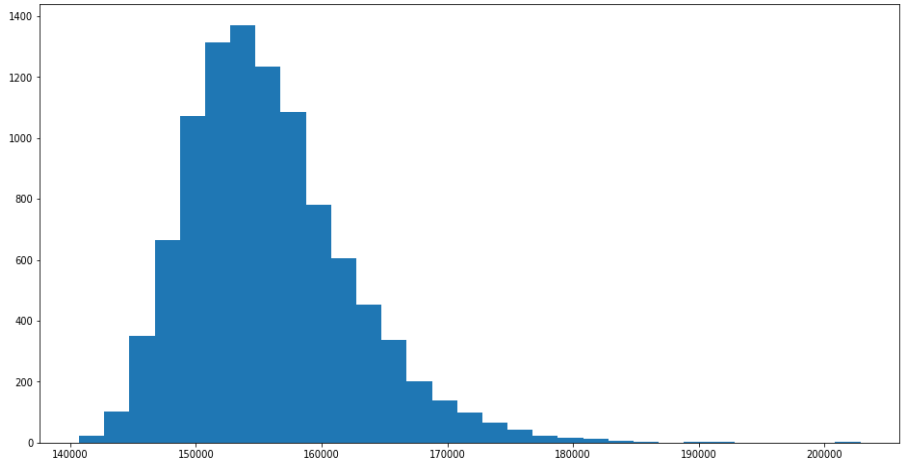
Data una sequenza di numeri, ad ogni passo si effettua un ordinamento parziale. Si sceglie un elemento in maniera casuale come pivot, si confrontano con esso gli altri elementi e si posizionano alla sua sinistra i minori e alla sua destra i maggiori, senza tener conto del loro ordine. Dopo questo passo, si ha che il pivot è ordinato in maniera corretta. In seguito, si ordinano ricorsivamente la parte sinistra e la parte destra di elementi rimasti non ordinati, fino al loro esaurimento.

Implementazione

```
1.  import random
2.  import matplotlib.pyplot as plt
3.  import math
4.
5.  #generazione casuale 10000 numeri
6.  """
7.  l = random.sample(range(10000), 10000)
8.  with open('numbers.txt', 'w') as f:
9.      for item in l:
10.         f.write("%s\n" % item)
11. """
12.
13. #confronti
14. comparisons = 0
15.
16. def Partition(A,left,right):
17.     global comparisons
18.     rindex = random.randrange(left,right+1)
19.     A[rindex],A[right] = A[right],A[rindex]
20.     pivot = A[right]
21.     i = left-1
22.     j = left
23.     while j < right:
24.         if A[j] <= pivot:
25.             i+=1
26.             A[i],A[j]=A[j],A[i]
27.         j+=1
28.         comparisons+=1
29.     A[i+1],A[right]=A[right],A[i+1]
30.     return i+1
31.
32. def LVQuickSort(A,left,right):
33.     if left < right:
34.         q = Partition(A,left,right)
35.         LVQuickSort(A,left,q-1)
36.         LVQuickSort(A,q+1,right)
37.
38. def main():
39.
40.   global comparisons
41.   #lista confronti ad ogni iterazione
42.   list_of_comparisons = []
43.   #totale di tutti i confronti effettuati
44.   totale_comparisons = 0
45.   #numeri run di iterazioni
46.   R = 10000
47.   #dimensioni lunghezza sequenza
48.   n = 10000
```

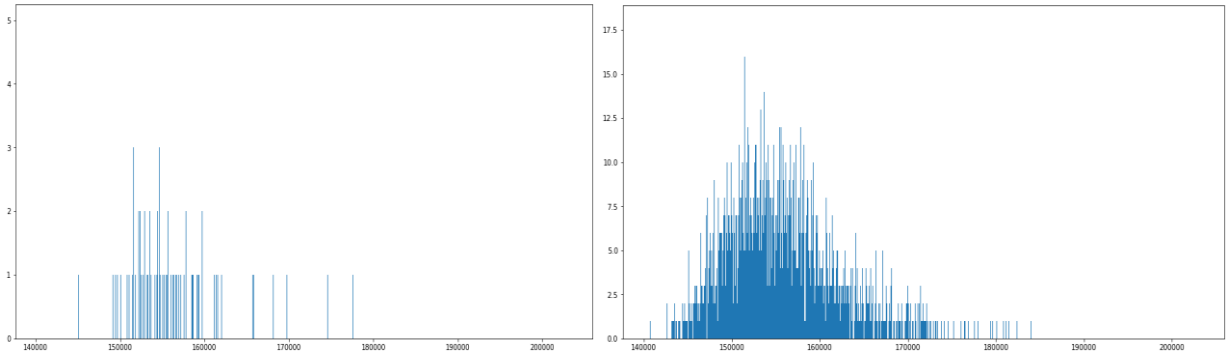
```
  """
  for i in range(0,R):
      A = []
      f=open('numbers.txt',"r")
      for line in f:
        A.append(int(line.strip('\n')))
      left = 0
      right = len(A)-1
      LVQuickSort(A,left,right)
      list_of_comparisons.append(comparisons)
      totale_compparisons+=comparisons
      comparisons = 0

#salvo i risultati in un file
 with open('comparisons.txt', 'w') as f:
   for item in list_of_comparisons:
       f.write("%s\n" % item)
 with open('totale_comparisons.txt', 'w') as f:
      f.write("%s\n" % totale_comparisons)

  """
#se ho già i risultati li apro
 with open('comparisons.txt',"r") as f:
      for line in f:
          list_of_comparisons.append(int(line.strip('\n')))

 with open('totale_comparisons.txt') as f:
   totale_comparisons = int(f.readline())

 #stampa la figura
 plt.figure(figsize=(16,8))
 plt.hist(list_of_comparisons, bins=int(math.log2(totale_comparisons)+1))

 #stima di C
 estimate_C = 1/R*totale_comparisons / (n*math.log(n))
 print("Stima di C: ", estimate_C)
main()
```

Punto 1



L'istogramma presenta sulle ascisse il numero di confronti, sulle ordinate la loro frequenza. Il numero di bins è stato calcolato con la formula di Struges: $[\log_2 n] + 1$, la quale è derivata da una distribuzione binomiale e assume implicitamente una distribuzione approssimativamente normale.

Punto 2



Se il numero di bins è troppo grande (grafico di sinistra) perdo le caratteristiche della distribuzione, se invece è troppo piccolo (grafico di destra) ho un'incertezza sulla forma.

Punto 3

Stima il valore di C

$R = 10^4\ run$

$n = 10^4\ dimensione\ della\ sequenza$

$N_j\ numero\ di\ confronti\ di\ in\ ogni\ singolo\ run\ j$

$N\ (numero\ totale\ di\ confronti) = \sum_{j=0}^{R} N_j = 1558058930$

$\frac{1}{R}\sum_{j=0}^{R} N_j \approx Cn\ln n$

$C \approx \frac{\frac{1}{R}N}{n\ln n} \approx \frac{\frac{1}{10^4}1558058930}{10^4 \ln 10^4} \approx 1.6916$