Amedeo Maria Paniccia Francesco Legrottaglie Giovanni Pica

RELAZIONE

In seguito al progetto assegnato, ossia quello di implementare una nuova versione dell'algoritmo **select** e di modificare il **quickSort,** la scelta del pivot non è randomica, bensì avviene tramite un algoritmo di selezione.
Riguardo alla scelta del sottoinsieme abbiamo utilizzato una funzione che prende in input la lista e il numero di elementi che vogliamo inserire.
Questa funzione consiste nell' inizializzare una variabile **"i"**, che ci servirà nel ciclo **while,** in modo da prendere esattamente m elementi.
Pertanto, abbiamo creato due liste, una chiamata **v** che avrà al suo interno tutti gli m elementi presi casualmente ed una **index** in cui metteremo al suo interno gli indici presi, in modo che si eviti di acquisire indici uguali.
All' interno del ciclo while è presente una variabile h che ha l'utilità di prendere indici randomici da 0 alla len (Lista) -1 (dal primo all' ultimo indice), in seguito verificheremo che h non sia stata già presa.
Riguardo alla scelta di **m**, abbiamo optato di renderlo variabile al fine di verificare più casi possibili ed una volta trovato il sottoinsieme lo impiegheremo nel nostro **sampleMedianSelect,** che prende in input: lista, l'indice dell'elemento che vogliamo "returnare"(in questo caso len(L)/2) e il nostro m.
Per l'implementazione di quest'ultimo abbiamo preso spunto dal sampleSelect di Floyd e Rivest, cambiando la condizione d'uscita inserendo al posto di len(L) <= 10, len(L) <= m, in questo modo eviteremo il caso che m sia > 10 e mandi in loop il programma.
Se len(L) è maggiore di **m** creiamo un sottoinsieme **V** e attraverso una variabile x troveremo il mediano di **V** richiamando **sampleMedianSelect** che avrà come indice di ritorno len(V)/2.
Una volta trovato lo utilizzeremo come pivot, partizioneremo la nostra lista rispetto ad esso in base alle condizioni da noi imposte (verifica delle condizioni di maggioranza, minoranza e uguaglianza).
Attraverso questa procedura troveremo il mediano perfetto della lista e lo utilizzeremo come pivot nel quickSort, effettuando uno scambio tra il primo elemento della lista e il pivot.

- File utilizzato per il codice: Codice.py

Amedeo Maria Paniccia Francesco Legrottaglie Giovanni Pica

TEST

(Il tempo impiegato è espresso in secondi)

I primi 3 test sono calcolati in base alla media di 5 prove. (Numeri di elementi 10000)

TEST 1: Lista già ordinata	**TEST 2: Lista Ordinata Inversamente**
Caso con un m piccolo, es. 100 tempo medio impiegato--> 2.2770360469818116 **Caso con m pari alla metà** tempo medio impiegato--> 3.8090388774871826 **Caso con m pari alla lunghezza** tempo medio impiegato--> 2.2662856578826904	**Caso con un m piccolo, es.100** tempo medio impiegato--> 2.8361432552337646 **Caso con m pari alla metà** tempo medio impiegato--> 4.714389181137085 **Caso con m pari alla lunghezza** tempo medio impiegato--> 11.953534364700317

TEST 3: Input Random
Caso con un m piccolo, es.100 tempo medio impiegato--> 0.03802857398986816 **Caso con m pari alla metà** tempo medio impiegato--> 1.6966344833374023 **Caso con m pari alla lunghezza** tempo medio impiegato--> 1.5782346248626709

TEST 4: Lunghezza Lista Variabile, Input Random

Caso con m piccolo es.100

50000 tempo impiegato--> 0.11606097221374512
55000 tempo impiegato--> 0.1265242099761963
60000 tempo impiegato--> 0.14020800590515137
65000 tempo impiegato--> 0.15989041328430176
70000 tempo impiegato--> 0.18741774559020996
75000 tempo impiegato--> 0.18165874481201172
80000 tempo impiegato--> 0.19362449645996094
85000 tempo impiegato--> 0.22972631454467773
90000 tempo impiegato--> 0.2247171401977539
95000 tempo impiegato--> 0.2353668212890625
100000 tempo impiegato--> 0.2539844512939453
105000 tempo impiegato--> 0.25351595878601074
110000 tempo impiegato--> 0.27639150619506836
115000 tempo impiegato--> 0.285250186920166
120000 tempo impiegato--> 0.2951974868774414
125000 tempo impiegato--> 0.30379700660705566
130000 tempo impiegato--> 0.3222830295562744
135000 tempo impiegato--> 0.332226037979126
140000 tempo impiegato--> 0.3498964309692383
145000 tempo impiegato--> 0.3608555793762207
150000 tempo impiegato--> 0.37874865531921387
155000 tempo impiegato--> 0.42052388191223145
160000 tempo impiegato--> 0.41282200813293457
165000 tempo impiegato--> 0.41211891174316406
170000 tempo impiegato--> 0.4300706386566162
175000 tempo impiegato--> 0.4579603672027588
180000 tempo impiegato--> 0.4588146209716797
185000 tempo impiegato--> 0.4697740077972412
190000 tempo impiegato--> 0.4956021308898926
195000 tempo impiegato--> 0.4924006462097168
200000 tempo impiegato--> 0.5147368907928467
205000 tempo impiegato--> 0.5245323181152344
210000 tempo impiegato--> 0.5416035652160645
215000 tempo impiegato--> 0.5596742630004883
220000 tempo impiegato--> 0.5718178749084473
225000 tempo impiegato--> 0.5751006603240967
230000 tempo impiegato--> 0.6173274517059326
235000 tempo impiegato--> 0.6080846786499023
240000 tempo impiegato--> 0.6479659080505371
245000 tempo impiegato--> 0.6339380741119385

TEST 4: Lunghezza Lista Variabile, Input Random

Caso con m pari alla metà

50000 tempo impiegato--> 1.0061123371124268
55000 tempo impiegato--> 0.9878380298614502
60000 tempo impiegato--> 0.9805524349212646
65000 tempo impiegato--> 0.9822444915771484
70000 tempo impiegato--> 0.9838137626647949
75000 tempo impiegato--> 0.980637788772583
80000 tempo impiegato--> 1.7200069427490234
85000 tempo impiegato--> 1.3773424625396729
90000 tempo impiegato--> 1.3869755268096924
95000 tempo impiegato--> 1.3160145282745361
100000 tempo impiegato--> 1.3145556449890137
105000 tempo impiegato--> 1.2987825870513916
110000 tempo impiegato--> 1.2834253311157227
115000 tempo impiegato--> 1.323732852935791
120000 tempo impiegato--> 1.3084735870361328
125000 tempo impiegato--> 1.3053500652313232
130000 tempo impiegato--> 1.3234906196594238
135000 tempo impiegato--> 1.3282482624053955
140000 tempo impiegato--> 1.3386788368225098
145000 tempo impiegato--> 1.364081859588623
150000 tempo impiegato--> 1.3647260665893555
155000 tempo impiegato--> 1.369882583618164
160000 tempo impiegato--> 1.842904806137085
165000 tempo impiegato--> 1.900019645690918
170000 tempo impiegato--> 1.8844964504241943
175000 tempo impiegato--> 1.768043041229248
180000 tempo impiegato--> 1.7795135974884033
185000 tempo impiegato--> 1.7855162620544434
190000 tempo impiegato--> 1.784661054611206
195000 tempo impiegato--> 1.7420673370361328
200000 tempo impiegato--> 1.7580862045288086
205000 tempo impiegato--> 1.7601406574249268
210000 tempo impiegato--> 1.7690081596374512
215000 tempo impiegato--> 1.7673988342285156
220000 tempo impiegato--> 1.7964487075805664
225000 tempo impiegato--> 1.7959766387939453
230000 tempo impiegato--> 1.8065180778503418
235000 tempo impiegato--> 1.8144168853759766
240000 tempo impiegato--> 1.8465664386749268
245000 tempo impiegato--> 1.820983648300171

TEST 4: Lunghezza Lista Variabile, Input Random

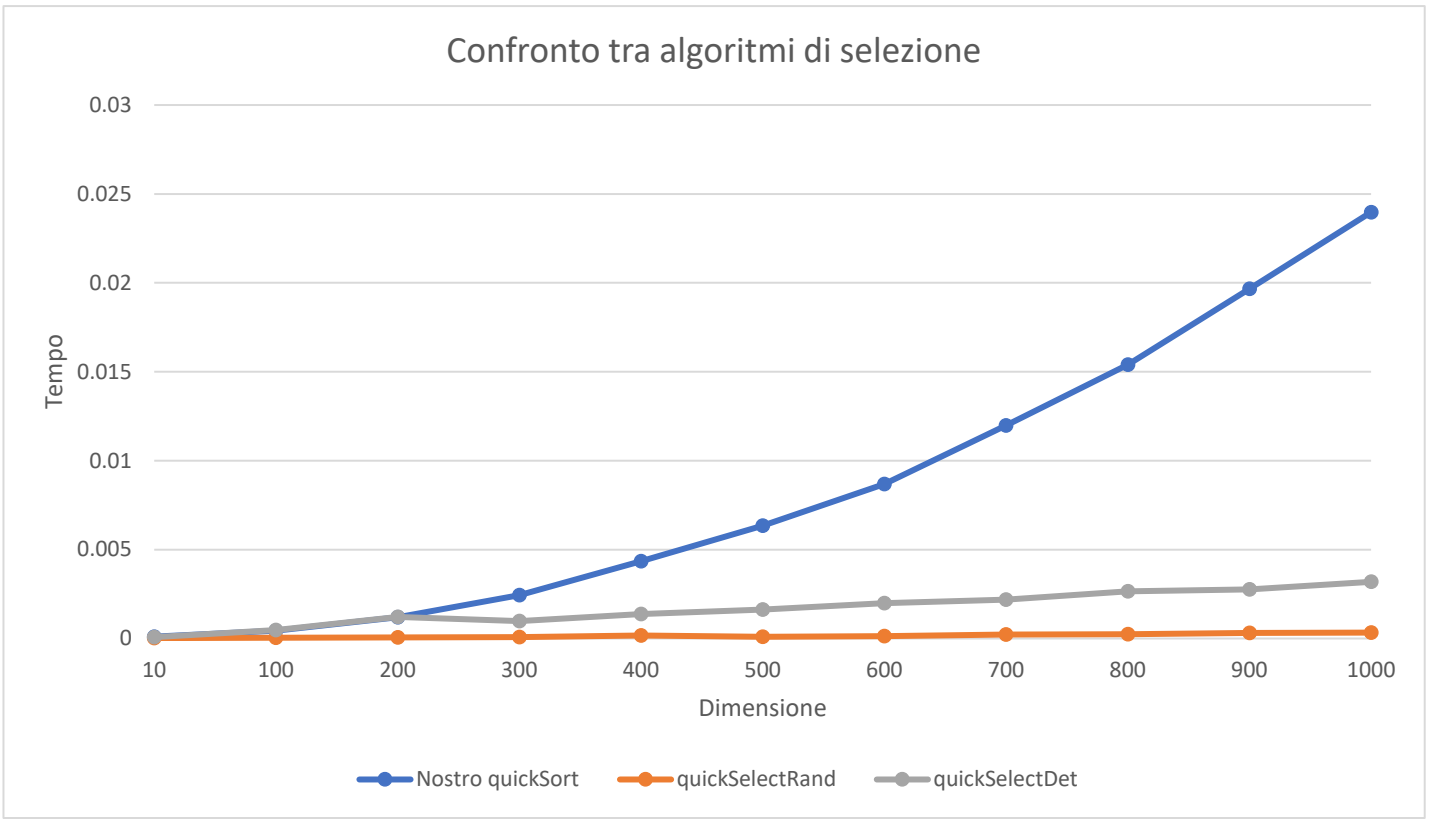
Caso con m pari alla lunghezza

50000 tempo impiegato--> 2.834660530090332
55000 tempo impiegato--> 2.6739675998687744
60000 tempo impiegato--> 2.612048864364624
65000 tempo impiegato--> 2.550551176071167
70000 tempo impiegato--> 2.525015354156494
75000 tempo impiegato--> 2.5126683712005615
80000 tempo impiegato--> 4.9265148639678955
85000 tempo impiegato--> 4.068574905395508
90000 tempo impiegato--> 3.865429639816284
95000 tempo impiegato--> 3.696943759918213
100000 tempo impiegato--> 3.6198301315307617
105000 tempo impiegato--> 3.5886688232421875
110000 tempo impiegato--> 3.696378707885742
115000 tempo impiegato--> 3.6991379261016846
120000 tempo impiegato--> 3.78180193901062
125000 tempo impiegato--> 3.682814359664917
130000 tempo impiegato--> 3.453493356704712
135000 tempo impiegato--> 3.428187370300293
140000 tempo impiegato--> 3.4230141639709473
145000 tempo impiegato--> 3.54036021232605
150000 tempo impiegato--> 3.5465495586395264
155000 tempo impiegato--> 3.5626142024993896
160000 tempo impiegato--> 6.003544807434082
165000 tempo impiegato--> 5.363947868347168
170000 tempo impiegato--> 4.872703552246094
175000 tempo impiegato--> 4.87328314781189
180000 tempo impiegato--> 4.795601844787598
185000 tempo impiegato--> 4.646936893463135
190000 tempo impiegato--> 4.678833723068237
195000 tempo impiegato--> 4.604268789291382
200000 tempo impiegato--> 4.637166976928711
205000 tempo impiegato--> 4.5418665409088135
210000 tempo impiegato--> 4.541441440582275
215000 tempo impiegato--> 4.500332832336426
220000 tempo impiegato--> 4.510225534439087
225000 tempo impiegato--> 4.531631231307983
230000 tempo impiegato--> 4.550189733505249
235000 tempo impiegato--> 4.513029336929321
240000 tempo impiegato--> 4.5441577434539795
245000 tempo impiegato--> 4.4852821826934814

TEST 5: m variabile, Input Random di 30 elementi

1 tempo impiegato--> 5.793571472167969e-05
2 tempo impiegato--> 6.318092346191406e-05
3 tempo impiegato--> 6.461143493652344e-05
4 tempo impiegato--> 5.578994750976562e-05
5 tempo impiegato--> 0.00010013580322265625
6 tempo impiegato--> 8.988380432128906e-05
7 tempo impiegato--> 9.870529174804688e-05
8 tempo impiegato--> 8.845329284667969e-05
9 tempo impiegato--> 0.00010442733764648438
10 tempo impiegato--> 6.175041198730469e-05
11 tempo impiegato--> 0.00011897087097167969
12 tempo impiegato--> 7.104873657226562e-05
13 tempo impiegato--> 0.00013971328735351562
14 tempo impiegato--> 6.937980651855469e-05
15 tempo impiegato--> 0.00013303756713867188
16 tempo impiegato--> 0.00010180473327636719
17 tempo impiegato--> 9.036064147949219e-05
18 tempo impiegato--> 0.00012135505676269531
19 tempo impiegato--> 9.632110595703125e-05
20 tempo impiegato--> 0.00014090538024902344
21 tempo impiegato--> 0.00010371208190917969
22 tempo impiegato--> 0.0001277923583984375
23 tempo impiegato--> 0.00013566017150878906
24 tempo impiegato--> 0.0001494884490966797
25 tempo impiegato--> 0.00014400482177734375
26 tempo impiegato--> 0.00017786026000976562
27 tempo impiegato--> 0.00018668174743652344
28 tempo impiegato--> 0.00016927719116210938
29 tempo impiegato--> 0.0001583099365234375
30 tempo impiegato--> 6.96182509765625e-05

- File utilizzato per i testing: testing.py



Abbiamo utilizzato come unità di misura nei grafici:

-asse x: dimensione lista che varia da 10 a 1000;

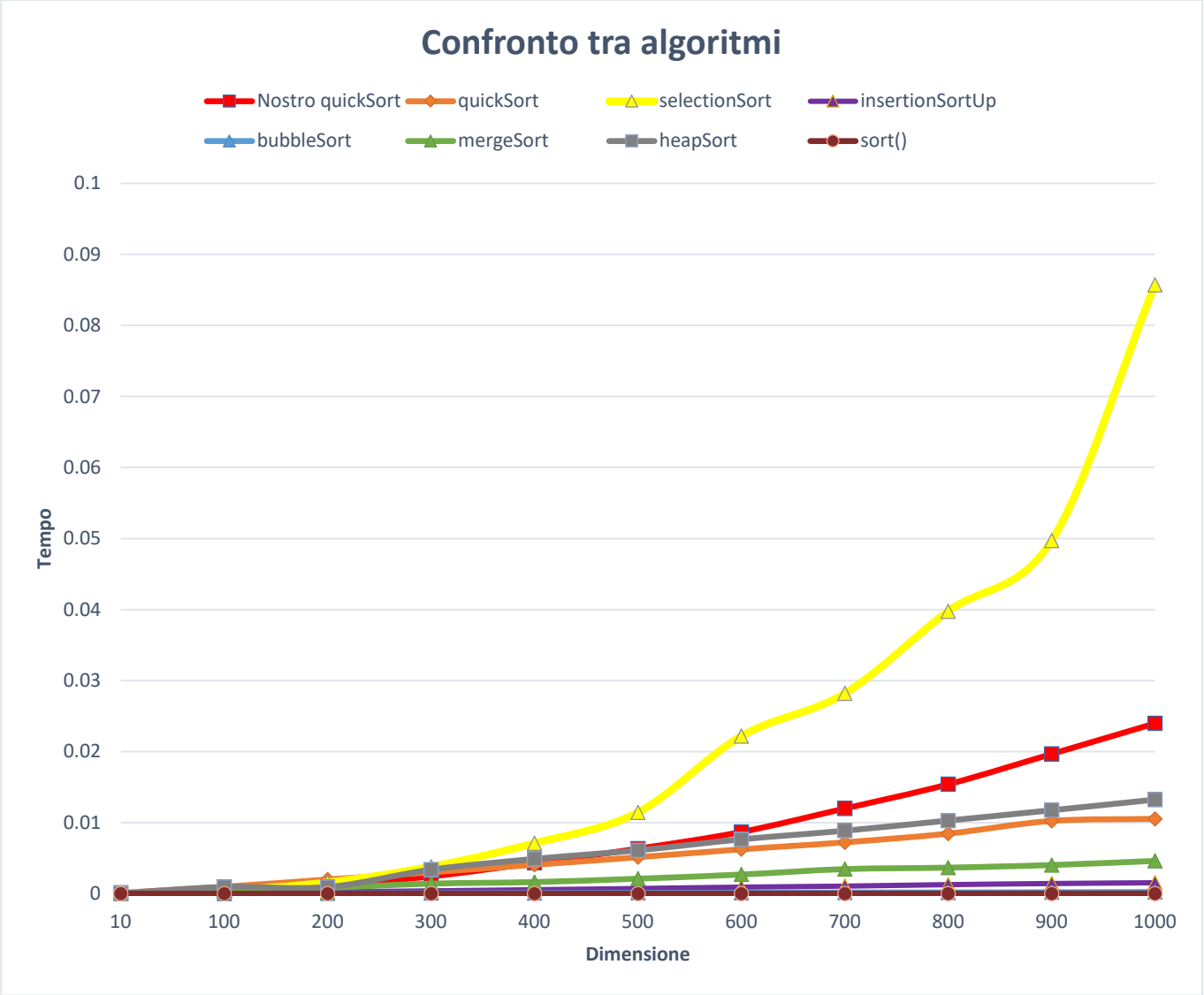
-asse y: tempo espresso in secondi

Inoltre abbiamo utilizzato come parametri nei test:

-m=5 a nostra scelta;

-lista ordinata all'aumentare della dimensione

In questo confronto effettuato con gli altri algoritmi di selezione, si nota come il **quickSelectRandom** sia quello più costante e veloce nel tempo.



In quest'ultimo test abbiamo confrontato la nostra variante di **quickSort** con gli altri algoritmi di ordinamento, utilizzando sempre una lista ordinata e m=5.

Dai test che abbiamo effettuato, evince che il **sort** di python è l'algoritmo più veloce e costante nel tempo all'aumentare delle dimensioni della lista.

- File utilizzato per il confronto tra algoritmi: testalgo.py