FLAT STABLE SORT

New stable sort algorithm

Copyright (c) 2017 Francisco José Tapia (fjtapia@gmail.com)

1.- INTRODU TION

!.- DES RI" TION OF T#E AL\$ORIT#%

!.1.- BASI ON E"TS

!.1.1.- Blo&' merge

!.1.!.- Se()e*&e merge

! .1.+.- E, am-le

!.1...- I*ter*al /etails

!.!.- NU%BER OF ELE%ENTS NO %ULTI" LE OF T#E BLO O SI1E

!.+.- S" E IAL ASES

! .+.1.- N)mber o2 eleme*ts *ot sorte/ less or e()al tha* the /o)ble o2 the blo&' si3e

! .+.! .- N) mber o2 eleme*ts *ot sorte/ greater tha* the /o) ble o2 the blo&' si3e

! .+.+.- Ba&' war/ sear&h

+.- BEN #%AROS

1.- INTRODUCTION

2lat_stable_sort is a *ew stable sort algorithm4 &reate/ a*/ im-leme*te/ b5 the a)thor4 whi&h)se a 6er5 low a//itio*al memor5 7 aro)*/ 18 o2 the /ata si3e9. The best &ase is O7N94 a*/ the a6erage a*/ worst &ase are O7NlogN9.

The si3e o2 the a//itio*al memor5 is : si3e o2 the /ata; ! <= > ?0

Data size	Additional memory	Percent
1%	1! 0	1.! 8
1\$. %	e 8

The algorithm is 2ast sorti*g)*sorte/ eleme*ts4 b)t ha/ bee* /esig*e/ 2or to be e, tremel5 e22i&ie*t whe* the /ata are *ear sorte/. B5 e, am-le :

- Sorte/ eleme*ts with)*sorte/ eleme*ts a//e/ at e*/ or at he begi**i*g.
- Sorte/ eleme*ts with)*sorte/ eleme*ts i*serte/ i* i*ter*al -ositio*s4 or eleme*ts mo/i2ie/ whi&h alter the or/ere/ se()e*&e.
- Re6erse sorte/ eleme*ts
- ombi*atio* o2 the three -re6io)s -oi*ts.

The res)lts obtai*e/ i* the sorti*g o2 1@@ @@@ *)mbers -I)s a -er&e*t o2) *sorte/ eleme*ts i*serte/ at e^* / or i* the mi//le 4 was

random	10.78
sorted sorted + 0.1% end sorted + 1% end sorted + 10% end	0.07 0.36 0.49 1.39
sorted + 0.1% middle sorted + 1% middle sorted + 10% middle	2.47 3.06 5.46
reverse sorted + 0.1% end reverse sorted + 1% end reverse sorted + 10% end	0.14 0.41 0.55 1.46
reverse sorted + 0.1% middle reverse sorted + 1% middle reverse sorted + 10% middle	3.16

The res)lts obtai*e/ with stri*gs a*/ oble&ts o2 se6eral si3es with /i22ere*t &om-ariso*s4 was e()all5 satis2a&tor5 a*/ a&&or/i*g with the e, -e&te/.

2.- DESCRIPTION OF THE ALGORITHM

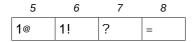
2.1.- INTRODUCTION

The -roblem of the merge algorithms is where -)t the merge/ eleme*tsB. This is the \$) stificatio* of the a//itio*al memor5 of the stable sort algorithms 7)s) all 5 a half of the memor5) se/ b5 the /ata9

This algorithm habe a /i22ere*t strateg5. The /ata are gro)-e/ i* blo&' s o2 2i, e/ si3e. All eleme*ts i*si/e a blo&' are or/ere/. S)--ose4 i*itiall54 tha* the *) mber o2 eleme*ts is a m) lti-le o2 the blo&' si3e. F) rther we see whe* this is *ot tr)e.

Other im-orta*t &o*&e-t is the se()e*&e. Ce &a* ha6e se6eral blo&'s or/ere/4 b)t the blo&'s are *ot -h5si&all5 &o*tig)o)s. Ce ha6e a 6e&tor o2 -ositio*s4 i*/i&ati*g the -h5si&all5 -ositio* o2 the blo&'s logi&all5 or/ere/ b5 the 6e&tor o2 -ositio*s. The se()e*&e is /e2i*e/ b5 a* iterator to the 2irst a*/ other to the a2ter the last eleme*t o2 the 6e&tor -ositio*.

B5 e, am-le

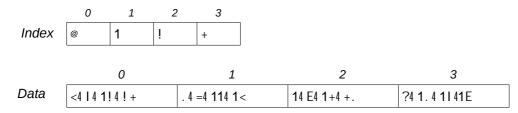


This se()e*&e \mathbb{I} <4 E \mathbb{I} im-I5 that the blo&'s o2 the se()e*&e are i* the -h5si&all5 -ositio*s 1@4 1!4? a*/ =. The /ata i*si/e this se()e*&e o2 *o* &o*tig)o)s blo&'s are or/ere/

The i/ea o2 the algorithms is similar to others merge algorithms. I*itiall5 sort N blo&'s a*/ ha6e N se()e*&es o2 1 blo&'. %erge the se()e*&e @ a*/ 14 ! a*/ +4 + a*/. a*/ 4 at the e*/ ha6e N;! se()e*&es o2 two blo&'s. I* the *ew se()e*&es obtai*e/4 merge the @ a*/ 14! a*/ + F a*/ obtai* N; . se()e*&es.

At e^*/o^2 this -ro&ess4 we ha6e o^*l5 1 se()e*&e. The logi&al or/er is i* a 6e&tor o² -ositio*s4)si*g this 6e&tor a*/ with a sim-le algorithms4 mo6e the eleme*ts 2rom their -h5si&all5 -ositio* to the logi& -ositio*4 a*/ the -ro&ess is 2i*ishe/. This -ro&ess is o^*e 0 o*e time a*/ is the last o-eratio* o² the algorithm.

2.1.1.- Block merge

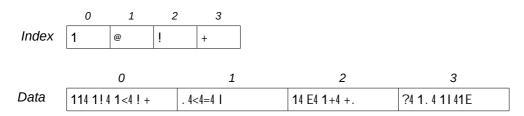


Ce ha6e . se()e*&es o2 1 blo&' . %erge the se()e*&es @ with 1 a*/ ! with +. A2ter this we will ha6e two se()e*&es o2 two blo&' s. Che* merge the two se()e*&es o2 two blo&' s 4 we obtai* a se()e*&e o2 . blo&' s4 whi&h is the 2i*al se()e*&e.

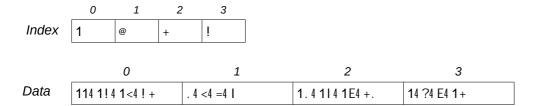
%erge o2 blo&' s @ a*/1

Mixer	E	Block 0	Block 1
. 4<4=41 41141! 41<	[!	! +	

The blo&' 1 is em-t5. Ce 2ill with the eleme*t 2rom the 2ro*t o2 the mi, er. $A^*/a//1$ to o)t-)t se()e*&e. The 2irst se()e*&e is em-t5. The remai*i*g eleme*ts o2 the mi, er are mo6e/ to the 2ro*t o2 the blo&' @4 a*/a// the @ to the o)t-)t se()e*&e. The o)t-)t se()e*&e is 0 @4 ! 9. Ce see i* the i*/e,



Doi*g the same with the blo&'s! a*/ +4 we obtai*



Ce ha6e *owl two se()e*&es o2 two blo&'s ea&h. The se()e*&es are /e2i*e/i* the i*/e,4 0 @4 !9 a*/ 0 !4 . 9. The 2irst se()e*&e are the -ositio*s @ a*/ 1 o2 the i*/e,4 a*/ the se&o*/ are the -ositio*s ! a*/ +.

For to merge4 ta'e the 2irs blo&' o2 the 2irst se()e*&e 7 1 % with the 2irst blo&' o2 the se&o*/ se()e*&e 7+% a*/ begi* the merge.

The 2irst em-t5 blo&' is the 1. Now we 2ill the blo&' 1 with the 2ro*t /ata o2 the mi, er4 a*/ a// 1 to the o)t-)t se()e*&e. For to s)bstit)te the blo&' 1 4 ta'e the *e, t o2 the 2irst se()e*&e4 the blo&' @4 a*/ &o*ti*)e with the merge.

Now the 2irst em-t5 blo&' is the +. Ce 2ill 2rom the 2ro*t o2 the mi, er4 a*/ i*sert the *)mber i* the o)t-)t se()e*&e. For to s)bstit)te the blo&' +4 ta'e the *e,t blo&' o2 the se()e*&e4 the !4 a*/ &o*ti*)e with the merge.

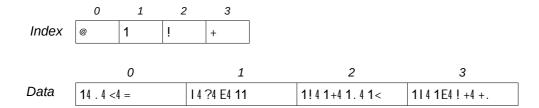
The *e,t em-t5 blo&' is the @. Ce 2ill 2rom the 2ro*t o2 the mi, er 4 a*/ a// their *) mber to the o)t-)t se()e*&e. The 2irst se()e*&e is em-t5. Now we habe o*l5 the blo&'! -artiall5 2ille/4 we 2ill 2rom the mi, er a*/ a// their -ositio* to the o)t-)t list

At e*/4 we &a* see

	0	1	2	3
Index	1	+	@	į

	0	1	2	3
Data	1! 4 1+4 1. 4 1<	14 . 4 <4 =	1 4 1 E 4 ! + 4 +.	I 4 ?4 E4 11

Now! we m)st mo6e the blo&'s! with a 6er5 sim-le algorithm! $a^*/$ -ass to logi&al sorte/ with a i^*/e , to a -h5si&all5 or/er. This is $/o^*e$ o*15 o*e time $a^*/$ is the last o-eratio* o2 the algorithm.



2.1.4.- Internal details

The a//itio*al memor5 *ee/e/ b5 the algorithm are the two blo&'s o2 the mi, er4 a*/ the list with the -ositio*s o2 the blo&'s. Che* merge two se()e*&es4 &o-5 ea&h se()e*&e i* a 6e&tor 4 a*/ the res)lt is o6er the i*/e,. I* the last merge4 the s)m o2 the si3e o2 these two 6e&tors is the same tha* the i*/e,. D)e this4 *ee/ the si3e o2 the i*/e, 4 m)lti-l5 b5 two.

I* the merge -ro&ess4 -arti&i-ate two blo&' s4 a*/ the two blo&' s o2 the mi, er. The si3e o2 the blo&' s is /esig*e/ 2or to allo&ate the . blo&' s i* the L1 &a&he o2 the -ro&essor. Normall5 the5 ha6e +! O 2or &ore4 a*/ ea&h &ore ha6e two threa/s4 the* 4 we ha6e 1=O 2or ea&h threa/.

D)e this the blo&' si3e m)st be . O4 a^* / the*: Si3e o2 o*e oble&t , Blo&' si3e J . O

Size of the object	Block Size
	1@! .
?	<1!
1=	!<=
+!	1!?
=,	=.

2.2.- NUMBER OF ELEMENTS NOT MULTIPLE OF THE BLOCK SIZE

Che* the *)mber o2 eleme*ts to sort is *ot a m)lti-le o2 the blo&' si3e4 we ha6e a* i*&om-lete 2i*al blo&' 4 &alle/ tail4 a*/ alwa5s is the last. Che* merge two se()e*&es4 i2 we ha6e tail4 alwa5s is i* the se&o*/ se()e*&e.

Che* e, ist tail blo&' i* the se&o*/ se()e*&e4 the merge is as showe/ be2ore. I2 the 2irst se()e*&e is em-t54 the -ro&e/)re is as /es&ribe/ be2ore.

B)t i2 the se()e*&e em-t5 is the se&o*/4 a*/ ha6e tail4 m)st /o a /i2ere*t -ro&e/)re. Ce see with a* e, am-le

	0	1	2	3	4	5
Index	1	ļ.	@		+	<

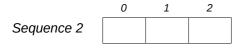
	0	1	2	3	4	5
Data	+?4 . ! 4 4 . =	14 <4 1+4 1E	! +4 ! E4 +. 4 +=	!!4!<4! 4!?	. 4 1. 4 114!@	+14 +!

Ce wa*t to merge the se() e*&e \mathbb{I} @4 + 94 5 \mathbb{I} +4 = 9. The blo&' < is i*&om-lete or tail blo&'.

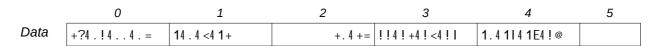
Begi* the merge4 a*/ the se&o*/ se()e*&e is em-t5. I* this i*sta*t the state o2 the /ata is



The blo&'! is -artiall5 em-t5.







Now we a// the tail blo&' to the $e^*/o2$ the 2irst se() e^* &e.

 Data pending of the sequence 1
 2
 0
 5

 + . 4 += +?4 . ! 4 . . 4 . =

The tail blo&' ha6e a si3e o2 two i* this e, am-le. Ce shi2t to the right the /ata $-e^*/i^*g^4$ the si3e o2 the tail blo&' 7! -ositio*s94 a*/ i*sert the /ata o2 the mi, er b5 the le2t si/e4 a*/ we ha6e

 Data pending of the sequence 1
 2
 0
 5

 !?4!E4+14+!
 +.4+=4+?4.!
 ..4.=

Now4 we habe the blo&'s 2ille/4 a^* / *ow we m)st i*sert their *)mbers i* the o)t-)t se()e*&e

 Output
 0
 1
 2
 3
 4
 5

 sequence
 1
 .
 +
 !
 @
 <</td>

 O
 1
 2
 3
 4
 5

 Data
 + . 4 += 4 +? 4 . !
 14 . 4 < 4 1 +</td>
 ! ? 4 ! E 4 + 14 +!
 ! ! 4 ! + 4 ! < 4 ! !</td>
 1 . 4 1 ! 4 1 E 4 ! @
 . . 4 . =

2.3.- SPECIAL CASES

The algorithm ha/ bee* /esig*e/ 2or to be e, tremel5 e2i&ie*t with the /ata *ear sorte/. Ce &a* &lassi25 i* 2o)r &ases:

- Sorte/ eleme*ts4 a*/ a//)*sorte/ eleme*ts to the begi**i*g or the e*/.
- Sorte/ eleme*ts a*/)*sorte/ eleme*ts are i*serte/ i* i*ter*al -ositio*s4 or eleme*ts mo/i2ie/4 whi&h alter the or/er o2 the eleme*ts.
- Re6erse sorte/ eleme*ts
- ombi*atio* o2 the 2irst + &ases

Begi* 2rom the 2irst -ositio*4 loo' i*g 2or sorte/ or re6erse sorte/ eleme*ts. I2 the *)mber o2 sorte/ or re6erse sorte/ eleme*ts is greater tha* a 6al)e 7)s)all5 K o2 the *)mber o2 eleme*ts% begi* with a* s-e&ial -ro&ess. I2 *ot4 a--I5 the ge*eral -ro&ess/es&ribe/ -re6io)sl5.

For to e, -lai* the s-e&ial -ro&ess4 we /o with a* e, am-le

12 the eleme*ts are referse sorte/4 mo6e betwee* them 2or to be sorte/.

B5 e, am-le4 i2 ha6e a* arra5 o2 1= eleme*ts4 with a blo&' si3e o2.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
+		<	I	?	1@	1+	1.	11	!1	=	1!	Е	11	!	1=

Range 1, 10 sorted elements

Range 2, 6 unsorted elements

Sort the ra*ge !4 a*/ obtai*

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
+		<	I	?	1@	1+	1.	11	! 1	!	=	E	11	1!	1=

Range 1, 10 sorted elements

Range 2, 6 sorted elements

The i/ea is to merge the two ra*ges4 a*/ &a* a--ear two &ases:

- 1. Che* the ra*ge! is lower or e()al tha* the /o)ble o2 the blo&' si3e
- !. Che* the ra*ge! is greater tha* the /o)ble o2 the blo&' si3e

2.3.1.- Number of elements lower or equal than the double of the block size.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
+		<	1	?	1@	1+	1.	11	! 1	=	1!	E	11	!	1=

Sort the ra*ge ! 4 a*/ a2ter this4 the i/ea is to) se as a), iliar5 memor5 the i*ter*al memor5 o2 the mi, e/ 7 ! blo&' s9. A*//o a* i*sertio* o2 sorte/ eleme*ts o2 the ra*ge ! i* the sorte/ eleme*ts o2 the ra*ge 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
+		<	1	?	1@	1+	1.	11	! 1	!	=	E	11	1!	1=

%o6e the ra*ge! i* the a), iliar5 memor5 a*/ obtai*

0	1	2	3	4	5
ļ	=	Е	11	1!	1=

3.- BENCHMARKS

The measured memory in the sorting of 100 000 000 numbers of 64 bits was:

In the time benchmark, the random, sorted and reverse sorted elements are 100000000 numbers of 64 bits. To these numbers, add 0.1%, 1% and 10% of unsorted elements inserted at the end and in the middle, uniformly spaced.

	[1]	[2]	[3]
random	8.51	9.45	10.78
sorted	4.86	0.06	0.07
sorted + 0.1% end	4.89	0.41	0.36
sorted + 1% end	4.96	0.55	0.49
sorted + 10% end	5.71	1.31	1.39
sorted + 0.1% middle	6.51	1.85	2.47
sorted + 1% middle	7.03	2.07	3.06
sorted + 10% middle	9.42	3.92	5.46
reverse sorted	5.10	0.13	0.14
reverse sorted + 0.1% end	5.21	0.52	0.41
reverse sorted + 1% end	5.27	0.65	0.55
reverse sorted + 10% end	6.01	1.43	1.46
reverse sorted + 0.1% middle	6.51	1.85	2.46
reverse sorted + 1% middle	7.03	2.07	3.16
reverse sorted + 10% middle	9.42	3.92	5.46