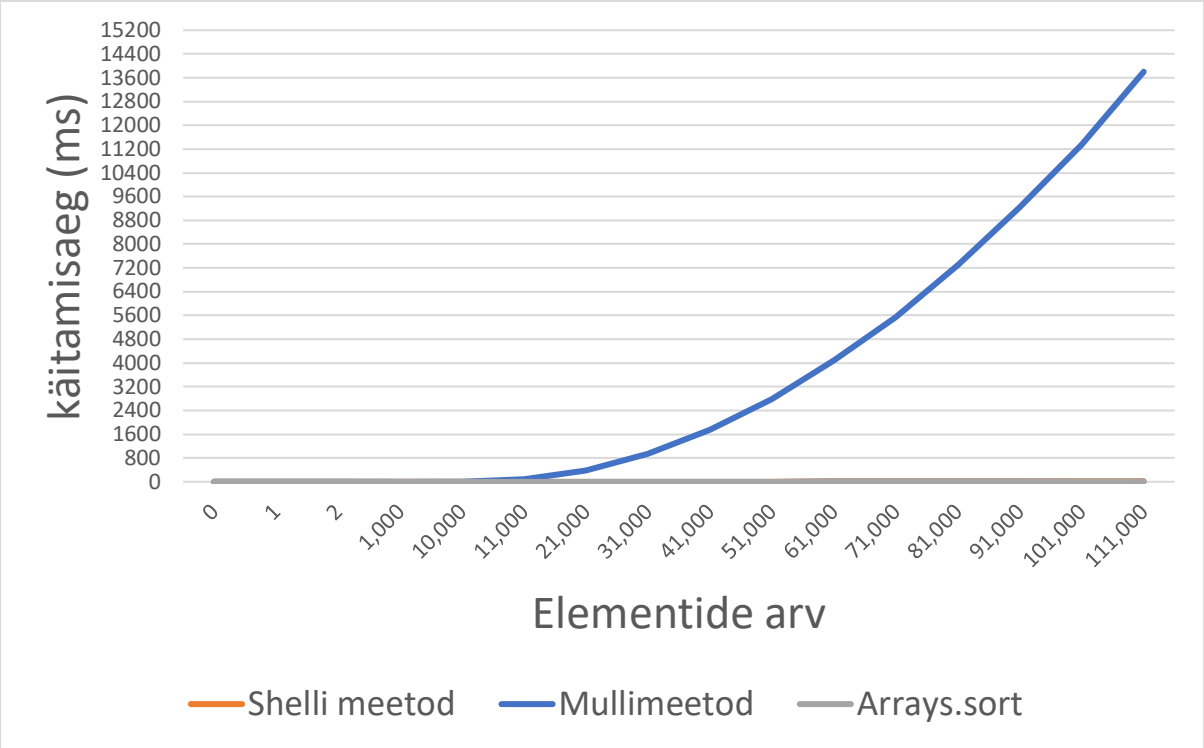
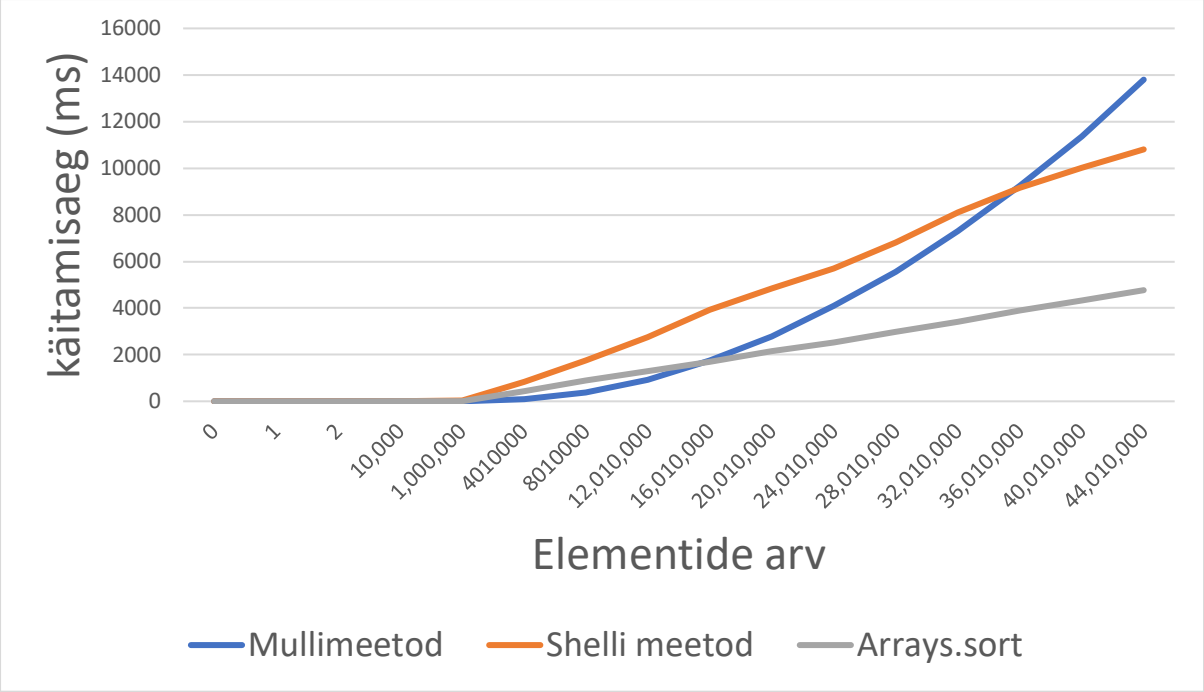
1.



Andmed:

Mullimeetod			Shelli meetod			Arrays.sort	
n	t		n	t		n	t
0	0		0	0		0	0
1	0		1	0		1	0
2	0		2	0		2	0
1 000	4		1 000	0		1 000	0
10 000	0		10 000	0		10 000	1
11 000	84		11 000	3		11 000	2
21 000	383		21 000	3		21 000	3
31 000	933		31 000	5		31 000	4
41 000	1741		41 000	8		41 000	6
51 000	2777		51 000	8		51 000	7
61 000	4093		61 000	10		61 000	9
71 000	5548		71 000	11		71 000	10
81 000	7301		81 000	14		81 000	10
91 000	9238		91 000	16		91 000	9
101 000	11363		101 000	18		101 000	11
111 000	13801		111 000	19		111 000	11



Andmed:

Mullimeetod		Shelli meetod		Arrays.sort	
n	t	n	t	n	t
0	0	0	0	0	0
1	0	1	0	1	0
2	0	2	0	2	0
1 000	4	10 000	3	10 000	3
10 000	0	1 000 000	22	1 000 000	1
11 000	84	4010000	835	4010000	434
21 000	383	8010000	1747	8010000	880
31 000	933	12 010 000	2743	12 010 000	1298
41 000	1741	16 010 000	3917	16 010 000	1708
51 000	2777	20 010 000	4853	20 010 000	2147
61 000	4093	24 010 000	5704	24 010 000	2537
71 000	5548	28 010 000	6804	28 010 000	2988
81 000	7301	32 010 000	8092	32 010 000	3399
91 000	9238	36 010 000	9149	36 010 000	3900
101 000	11363	40 010 000	10007	40 010 000	4316
111 000	13801	44 010 000	10809	44 010 000	4768

Mullimeetod - sorteerimisalgoritm, mille eesmärk on sorteerida järjend kasvavas või kahanevas järjekorras, võrreldes elemente paarikaupa ja vahetades neid vajadusel, nii kaua kuni järjend on sorteeritud.

Algoritmi tööpõhimõte:

Algoritm läbib järjendit ja võrdleb elemente paarikaupa ja kui esimene element on väiksem, kui teine, siis toimub nende vahetamine. Esimese järjendi läbimise protsessi

lõpus on kõige väiksem element viimane. Kui järjendi läbimise jooksul elementide ümberpaigutamist ei juhtu, siis see tähendab, et järjend on sorteeritud.
Selle algoritmi keskmine ajaline keerukus on $\theta(n^2)$.

Shelli meetod on sorteerimisalgoritm, mis kasutab mitmeid järjestikuseid samme järjendi järkjärguliseks sorteerimiseks ja lõpuks täiesti sorteeritud järjendini jõudmiseks.
Siin on Shell meetodi tööpõhimõte: valitakse samm, kus iga järgmine samm on väiksem kui eelmine. Järjend jagatakse mitmesse ossa, kus iga osa koosneb elementidest, mis on üksteisest sammhaaval eraldatud. Seejärel sorteeritakse iga osa eraldi. Kui tegu on sammuga 1, tehakse lõplik sorteerimine, kus võrreldakse naabruses asuvaid elemente, ning selleks ajaks on järjend juba osaliselt sorteeritud, nii et vahetusi on vähem.
Selle algoritmi ajaline keerukus on $\theta(nlog(n))$.

Arrays.sort on Java programmeerimiskeele sisseehitatud meetod, mida kasutatakse massiivi sorteerimiseks kasvavas järjekorras. Arrays.sort meetod kasutab kiiret sorteerimisalgoritmi (Timsort). Selle meetodi keskmine ajaline keerukus on $\theta(nlog(n))$. (Allikas: https://www.gregorygaines.com/blog/what-is-the-time-complexity-arrays-and-collections-sort/)

Tavaliselt on ootuspärane, et järjendid ei ole juba täielikult sorteeritud ja ei ole vastupidiselt sorteeritud. Selle asemel peaksid elemendid olema enamasti juhuslikus järjekorras või lähedal keskmisele juhtumile, kuna arve genereeritakse juhuslikult.
2.
Mullimeetodi graafik näitab, kuidas selle sorteerimismeetodi tööaeg suureneb vastavalt sisendandmemahtude kasvule. Graafikul on näha, et mullimeetodi tööaeg suureneb peaaegu eksponentsiaalselt sisendandmete arvu suurenemisega. See tähendab, et mullimeetod on ajalises mõttes ebaefektiivne suurte andmemahtude korral, kuna, kui suurem on andmemaht seda kiirem kasvab algoritmi tööaeg. Graafikul on selgelt näha, kuidas tööaeg tõuseb järsult, eriti kui andmete arv ületab 20000 elementi. Seega võib

väita, et saadud graafik on sarnane oodatud graafikuga. Mõned kõrvalekalded võivad olla põhjustatud operatsioonisüsteemi valikuga ja arvuti jõudlusega.

Shelli meetodi graafik näitab, kuidas selle sorteerimismeetodi tööaeg suureneb vastavalt sisendandmemahtude kasvule. Graafik näitab, et Shelli meetod suudab hoida suhteliselt madalat ajalist keerukust isegi suuremate andmemahtude korral. Seda on väga hästi näha, kuna sõltumatu andmemahu suurusest kasvab graafik peaaegu lineaarselt, mis tähendab, et ta on sarnane oodatud funktsiooni graafikuga

Arrays sort graafik näitab, et Arrays sort toimib väga kiiresti suurte andmemahtude korral. Tööaeg suureneb samuti üsna lineaarselt andmemahu suurenemisega. Lineaarne kasv andmete suurenemisel näitab, et Arrays sort suudab hoida madalat ajalist keerukust ka suurte andmete puhul, kuna ta kasutab oluliselt kiiremaid meetodeid, kui teised vaadeldud algoritmid.

3.
Mullimeetodil keskmine ja halvem ajaline keerukus on samad $\theta(n^2)$, kuna ei ole vahet, kas järjend on sorteeritud või üldse ei ole algoritm nii kui nii võrdleb kõiki paare.

Shelli meetodil võib realiseeruda ka keskmisest juhust halvema ajalise keerukusega juht, siis kui järjendi elemendid on sorteeritud vastupidi.

Arrays sort meetodil võib realiseeruda ka keskmisest juhust halvema ajalise keerukusega juht, siis kui järjendi elemendid on juba sorteeritud või on sorteeritud vastupidi.

4.
Kõikide kolme algoritmide puhul valitud andmemahu ja väärtusvaru korral on tegemist selle sorteerimismeetodi jaoks keskmise juhuga.
Selle kontrollimiseks oli kasutatud valem T(n) / oodatud ajaline keerukus.

Mullimeetod ($T(n) \,/\, n^2$) – saadud väärtused on iga n jaoks peaaegu samalt fikseeritud lõigult [1.0; 1.14].

Shelli meetod ($T(n) \,/\, nlog(n)$) – saadud väärtused on iga n jaoks peaaegu samalt fikseeritud lõigult [13.0; 14].

Arrays.sort meetod ($T(n) \,/\, nlog(n)$) – saadud väärtused on iga n jaoks peaaegu samalt fikseeritud lõigult [9.5; 12].

Ainukesed erandit olid katsed, kus andmemaht oli suhteliselt väike ja kui algoritmid töötasid järjendiga, mis oli juba sorteeritud.