Psuedo kode

OPPGAVE 1A

push_back

loop til arrayen er ferdig

- Loop finner slutten av elementet
- Loopen setter inn elementet bakerst i noden

loop (i < lengde av array)

If i er på slutten

Then sette in element

• **push_front**

E => element

A => det vi vil sette inn

Vanlig liste E > E > E > E

Etter metoden A > E > E > E > E

Finner første node I lenketlisten

Setter in elementet før det første elementet

(går inni den første noden)

currentNode = nyNode

nyNode.neste = currentNode

• **push_middle**

setinn((k + 1)/2, nyElement) {

loop (k+1/2 ganger) gjennon lenketlisten og når vi kommer til den noden

currentNode = nyElement

nyElement.neste = currentNode

}

Vi vil få dette før og etter setINN metoden:

E => element // A => det vi vil sette inn

E > E > E > E

E > E > A > E > E

- **get**

Get (i) {

loop (i ganger) gjennon lenketlisten og når vi kommer til den noden

return currentNode

}

Oppgave b

Se koden

OPPGAVE C

Push back: O(n)

Push front: O(1)

Push middle: O(n)

Get: O(n)

OPPGAVE D

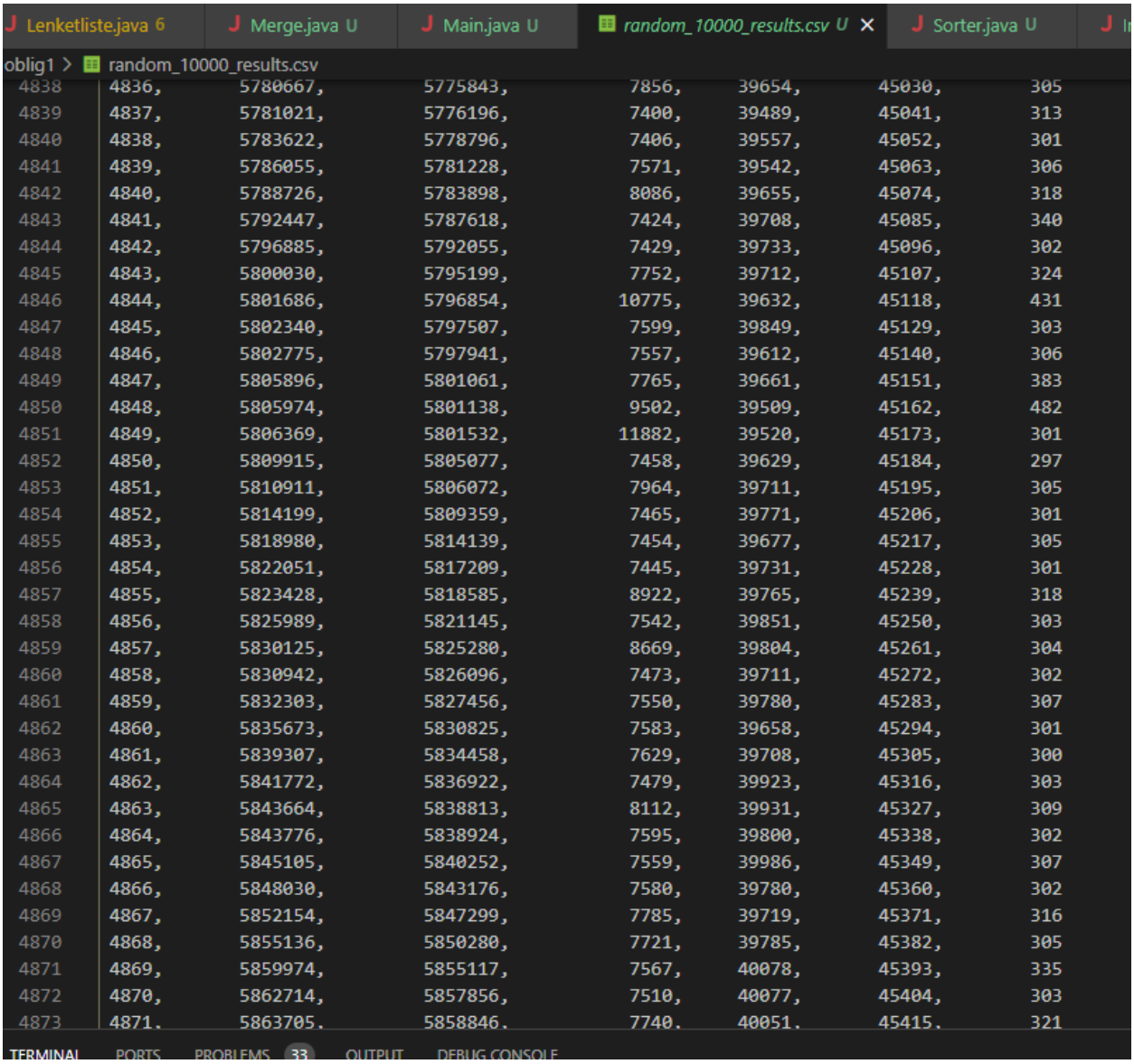
Fordi hvis N blir begrenset og antatt at java kan maksimalt ha en begrensning på 10^6, så fordi vi alltid antar det verste tilfelle, antar vi alltid at O(10^6) som kan skrives til O(1). Dette gjør at det alltid blir konstant tid.

Sortering

Se koden.

Eksperimenter

- Stemmer ganske bra, her er forskjellen mellom merge sort og insertionsort, N ble < 4000 så insertion sort brukte mye mer tid.



4838	4836,	5780667,	5775843,	7856,	39654,	45030,	305
4839	4837,	5781021,	5776196,	7400,	39489,	45041,	313
4840	4838,	5783622,	5778796,	7406,	39557,	45052,	301
4841	4839,	5786055,	5781228,	7571,	39542,	45063,	306
4842	4840,	5788726,	5783898,	8086,	39655,	45074,	318
4843	4841,	5792447,	5787618,	7424,	39708,	45085,	340
4844	4842,	5796885,	5792055,	7429,	39733,	45096,	302
4845	4843,	5800030,	5795199,	7752,	39712,	45107,	324
4846	4844,	5801686,	5796854,	10775,	39632,	45118,	431
4847	4845,	5802340,	5797507,	7599,	39849,	45129,	303
4848	4846,	5802775,	5797941,	7557,	39612,	45140,	306
4849	4847,	5805896,	5801061,	7765,	39661,	45151,	383
4850	4848,	5805974,	5801138,	9502,	39509,	45162,	482
4851	4849,	5806369,	5801532,	11882,	39520,	45173,	301
4852	4850,	5809915,	5805077,	7458,	39629,	45184,	297
4853	4851,	5810911,	5806072,	7964,	39711,	45195,	305
4854	4852,	5814199,	5809359,	7465,	39771,	45206,	301
4855	4853,	5818980,	5814139,	7454,	39677,	45217,	305
4856	4854,	5822051,	5817209,	7445,	39731,	45228,	301
4857	4855,	5823428,	5818585,	8922,	39765,	45239,	318
4858	4856,	5825989,	5821145,	7542,	39851,	45250,	303
4859	4857,	5830125,	5825280,	8669,	39804,	45261,	304
4860	4858,	5830942,	5826096,	7473,	39711,	45272,	302
4861	4859,	5832303,	5827456,	7550,	39780,	45283,	307
4862	4860,	5835673,	5830825,	7583,	39658,	45294,	301
4863	4861,	5839307,	5834458,	7629,	39708,	45305,	300
4864	4862,	5841772,	5836922,	7479,	39923,	45316,	303
4865	4863,	5843664,	5838813,	8112,	39931,	45327,	309
4866	4864,	5843776,	5838924,	7595,	39800,	45338,	302
4867	4865,	5845105,	5840252,	7559,	39986,	45349,	307
4868	4866,	5848030,	5843176,	7580,	39780,	45360,	302
4869	4867,	5852154,	5847299,	7785,	39719,	45371,	316
4870	4868,	5855136,	5850280,	7721,	39785,	45382,	305
4871	4869,	5859974,	5855117,	7567,	40078,	45393,	335
4872	4870,	5862714,	5857856,	7510,	40077,	45404,	303
4873	4871,	5863705,	5858846,	7740,	40051,	45415,	321

- Man kan legge merke til at insertion sort starter ganske bra og raskt når den har veldig får swaps, sånn 200 swaps og 200 comparisons, og insertion klarer å gi ganske raske results som i forhold til merge sort er også raskere med noen millisekunder, problemet blir at insertion sort bygger opp swaps og comparisons mye raskere enn merge sort, når merge sort er på 200 swaps og comparisons så er insertionsort allerede på 2000 swaps og comparisons. Det som overasket meg veldig mye var hvor effektivt insertion sort er på nesten sorterte lister enn merge sort.

- *Bilde nedenfor viser insertion sort som klarer kjøretiden ganske bra på lav teskel*

5,	7,	4,	0,	1,	3,	7
6,	12,	9,	0,	3,	4,	11
7,	18,	15,	0,	2,	6,	0
8,	25,	21,	0,	3,	8,	0
9,	28,	23,	0,	7,	10,	0
10,	35,	29,	1,	7,	12,	2
11,	36,	29,	0,	9,	14,	0
12,	46,	38,	1,	11,	16,	1
13,	50,	41,	1,	11,	18,	1
14,	56,	46,	1,	13,	20,	0
15,	60,	49,	0,	17,	23,	1
16,	69,	57,	0,	13,	26,	1
17,	72,	59,	0,	19,	29,	1
18,	87,	73,	1,	17,	32,	2
19,	97,	82,	1,	20,	35,	1
20,	101,	85,	1,	26,	38,	1
21,	112,	95,	1,	27,	41,	1
22,	113,	95,	1,	29,	44,	1
23,	116,	97,	1,	35,	47,	1
24,	126,	106,	1,	35,	50,	1
25,	147,	126,	1,	36,	53,	1
26,	159,	137,	1,	38,	56,	1
27,	176,	153,	1,	44,	59,	1
28,	182,	158,	2,	44,	62,	1
29,	189,	164,	2,	44,	65,	2

- *Nedenfor bilde ser vi hvordan insertion sort faller off +ratio*

82,	1933,	1858,	28,	205,	296,	6
83,	1936,	1860,	28,	207,	301,	5
84,	1980,	1903,	29,	217,	306,	5
85,	2050,	1972,	28,	224,	311,	7
86,	2053,	1974,	31,	248,	316,	5
87,	2089,	2009,	33,	250,	321,	7
88,	2129,	2048,	31,	238,	326,	5
89,	2191,	2109,	31,	269,	331,	7
90,	2270,	2187,	29,	260,	336,	11
91,	2323,	2239,	38,	269,	341,	9
92,	2333,	2248,	37,	269,	346,	6
93,	2400,	2314,	39,	268,	351,	8
94,	2425,	2338,	41,	278,	356,	8
95,	2454,	2366,	39,	282,	361,	8
96,	2542,	2453,	21,	286,	366,	5
97,	2630,	2540,	23,	281,	371,	5
98,	2695,	2604,	22,	300,	376,	5

- Av de to jeg har testet vile jeg bruk insertion sort for lav teskel og merge sort for større operasjoner. Det er også mulig å bruke andre algoritmer og teste dem.
- Når inputfilen er nesten sortert så funker insertion sort veldig bra, spesielt gjo mer elemter som blir lagt til, mens merge sort funker veldig bra på random sorterte lister spesielt gjo større innhold de har.

- Jeg har observert mellom insertion sort og merge sort. Jeg har lagt merke til at insertion sort funker ganske raskt på små arrays som har lite elementer i seg, mens merge sort blir mye mer effektivt gjo større arrayen blir, så stor effektivitet at insertion sort kunne ikke holde seg sammenlignet med merge sort, jeg ventet… ok mindre 2 minutter i loading screen.
- Dette stemte med at insertion sort er O(n^2) og merge sort er O(n log (n)) og jeg ble ikke særlig overasket over at det ble så stor forskjell i kjøretid når det kom til random_10000, men ble veldig overasket over nearly_sortet_1000.