IN2010 - Oblig 3

Stian Østgaard
Thuan Tran
Embrik Thoresen

05.11.2021

Nyttig informasjon:

Nedenfor er det informasjon knyttet til endring av prekode som er gjort, hvordan mappestrukturen vår ser ut og forklaring på hvordan man kjører koden.

Endring av prekode:

Vi har endret følgende kode fra 'oblig3runner.py'. Dette har vi gjort slik at når vi printer resultatet på deloppgave 1 til outputfilen som vi ønsker å skrive til, så printer vi filen til mappen 'outputs' som inneholder resultatene ved å kjøre en gitt algoritme på en inputfil fra mappen 'inputs'.

```
def run_algs_part1(A, infilename):
    infilename = infilename.split("/")[1]
    for alg in ALGS1:
        countA = CountSwaps([CountCompares(x) for x in A])
        outfilename = "outputs/" + infilename + '_' + algname(alg) + '.out'
        outstr = '\n'.join(map(str, alg(countA)))
        with open(outfilename, 'w') as f:
            f.write(outstr)
```

Videre har vi også endret følgende kode fra 'oblig3runner.py'. Grunnen til dette er at vi igjen kjører algoritmene på en gitt inputfil fra mappen 'inputs' og skriver til en ny fil som befinner seg i mappen 'outputs'.

```
def run_algs_part2(A, infilename):
    infilename = infilename.split("/")[1]
    outfilename = "outputs/" + infilename + '_results.csv'
    discarded = set()
    ...
```

Til slutt la vi til en konstruktør i 'countcompares.py' som man kan se på teksten nedenfor. Dette er ment for å overskride 'less than' slik at vi også kan teste 'less than or equal'.

```
from functools import total_ordering

@total_ordering
class CountCompares:
    def __init__(self, elem):
        self.elem = elem
        self.compares = 0
```

```
8
9       def reset(self):
10          self.compares = 0
11
12      def __eq__(self, other):
13          return self.elem == other.elem
14
15      def __lt__(self, other):
16          self.compares += 1
17          return self.elem < other.elem
18
19      def __le__(self, other):
20          self.compares += 1
21          return self.elem <= other.elem
22
23      def __repr__(self):
24          return self.elem.__repr__()
```

Mappestrukturen vår:

```
embrik@Embrik-Ubuntu:~/UiO/IN2010/innlevering/IN2010-Oblig3/python-scr$ ls
countcompares.py  heap.py       oblig3.py        __pycache__
countswaps.py     inputs        oblig3runner.py  quick.py
```

Figur 1 - Mappestrukturen hvor filene er lagret inne i mappen 'python-scr'.

Kjøring av programmet:

```
embrik@Embrik-Ubuntu:~/UiO/IN2010/innlevering/IN2010-Oblig3$ python3 oblig3.py inputs/xxx
```

Figur 2 - Hvordan kjøre programmene, der xxx står for ønsket inputfil fra mappen 'inputs'.

Deloppgave 1: Korrekthet

```
embrik@Embrik-Ubuntu:~/UiO/IN2010/innlevering/IN2010-Oblig3/python-scr/inputs$ ls
nearly_sorted_10    nearly_sorted_10000    random_10    random_10000
nearly_sorted_100   nearly_sorted_100000   random_100   random_100000
nearly_sorted_1000  nearly_sorted_1000000  random_1000  random_1000000
```

Figur 3 - Hvordan inputfilen vår ser ut.

```
embrik@Embrik-Ubuntu:~/UiO/IN2010/innlevering/IN2010-Oblig3/python-scr/outputs$ ls
nearly_sorted_1000000_insertion.out  random_100000_quick.out
nearly_sorted_100000_insertion.out   random_100000_results.csv
nearly_sorted_10000_insertion.out    random_10000_quick.out
nearly_sorted_10000_results.csv      random_10000_results.csv
nearly_sorted_1000_heap.out          random_1000_heap.out
nearly_sorted_1000_insertion.out     random_1000_insertion.out
nearly_sorted_1000_quick.out         random_1000_quick.out
nearly_sorted_1000_results.csv       random_1000_results.csv
nearly_sorted_1000_selection.out     random_1000_selection.out
nearly_sorted_100_heap.out           random_100_heap.out
nearly_sorted_100_insertion.out      random_100_insertion.out
nearly_sorted_100_quick.out          random_100_quick.out
nearly_sorted_100_results.csv        random_100_results.csv
nearly_sorted_100_selection.out      random_100_selection.out
nearly_sorted_10_heap.out            random_10_heap.out
nearly_sorted_10_insertion.out       random_10_insertion.out
nearly_sorted_10_quick.out           random_10_quick.out
nearly_sorted_10_results.csv         random_10_results.csv
nearly_sorted_10_selection.out       random_10_selection.out
```

Figur 4 - Hvordan outputfilen vår ser ut.

Som man kan se på figur 4, har vi implementert insertion sort, selection sort, heapsort og quicksort. Vi har valgt å kun kjøre de algoritmene vi vet er raskest, basert på testing, på de største filene. For å teste at alle algoritmene gir rimelige svar har vi valgt å ...

Deloppgave 2: Sammenligninger, bytter og tid

Vi teller antall sammenligninger og bytter ved å bruke CountSwaps og CountCompares. For bytte to elementer i A kaller må man kalle på metoden A.swap(i,j) som bytter elementene A[i] og A[j] samtidig som man øker variablelen self.swaps med 1. Dermed vil self.swaps telle antall bytter i en algoritme. For å telle sammenligninger overskrider vi 'less than or equal' ($\leq$) og 'less than' ($<$) operasjonene til å øke variablelen self.comparisons med 1. Så self.comparisons teller antall sammenligninger i en algoritme. A eksisterer som en lokal instans i hvert algoritme-kall, dermed eksisterer disse telle-variablene lokalt.

Figurene under viser outputen når vi kjører 'random' og 'nearly sorted' med 1000 inputs. Til slutt printer vi ut den totale tiden målt i mikrosekunder. Det skal sies at vi fikk ulik svar når vi kjørte det på tre ulike PCer. Dette er grunnet timelimiten som er oppgitt i prekoden, hvor vi alle har forskjellige PCer.

```
embrik@Embrik-Ubuntu:~/UiO/IN2010/innlevering/IN2010-Oblig3/python-scr$ python3 oblig3.py inputs/nearly_sorted_1000
   n, insertion_cmp, insertion_swaps, insertion_time, quick_cmp, quick_swaps, quick_time, selection_cmp, selection_swaps, selection_time, heap_cmp, heap_swaps, heap_time
 283,          413,           132,          109,      2783,        417,        782,        39621,           79,          7057,     3923,      2224,      1498
 364,          543,           181,          147,      3808,        546,       1043,        65703,          104,         11706,     5350,      3019,      2012
 421,          630,           211,          171,      4098,        641,       1143,        87990,          125,         15953,     6344,      3569,      2365
 466,          688,           224,          186,      4415,        715,       1253,       107880,          134,         20107,     7136,      4006,      2696
 504,          745,           243,          203,      5151,        763,       1413,       126253,          147,         23365,     7804,      4374,      2930
 538,          799,           263,          218,      6116,        793,       1626,       143916,          155,         26007,     8465,      4748,      3224
 568,          843,           277,          258,      6221,        837,       1668,       160461,          161,         28991,     9052,      5068,      3425
 595,          882,           289,          261,      6539,        870,       1713,       176121,          170,         32809,     9576,      5343,      3588
 620,          915,           297,          252,      6635,        931,       1797,       191271,          175,         35635,    10064,      5611,      3780
 643,          954,           313,          261,      6998,        949,       1877,       205761,          185,         38868,    10520,      5858,      4022
 665,          990,           327,          276,      7039,       1007,       1916,       220116,          192,         40813,    10954,      6097,      4133
 686,         1018,           334,          284,      7836,       1004,       2037,       234270,          196,         46118,    11365,      6326,      4426
 706,         1055,           351,          289,      7419,       1071,       2054,       248160,          204,         46938,    11752,      6537,      4460
 725,         1079,           356,          296,      8085,       1085,       2140,       261726,          209,         48821,    12128,      6738,      4516
 743,         1105,           364,          304,      8737,       1100,       2281,       274911,          213,         51626,    12482,      6935,      4715
 760,         1135,           377,          330,      8762,       1152,       2312,       287661,          219,         53234,    12814,      7123,      4858
 776,         1161,           387,          318,      8749,       1150,       2335,       299925,          226,         55686,    13134,      7298,      4945
 792,         1179,           389,          390,      8209,       1175,       2218,       312445,          228,         58807,    13448,      7463,      5060
 807,         1195,           390,          330,      8606,       1181,       2312,       324415,          229,         60448,    13756,      7634,      5150
 822,         1219,           399,          333,      9474,       1219,       2506,       336610,          232,         63244,    14048,      7791,      5409
 836,         1240,           406,          375,      9412,       1242,       2517,       348195,          234,         64721,    14318,      7936,      5471
 849,         1253,           406,          374,      9377,       1260,       2493,       359128,          237,         67717,    14583,      8072,      5466
 862,         1272,           412,          347,     10447,       1292,       2734,       370230,          243,         67960,    14837,      8225,      5650
 875,         1291,           418,          353,     10123,       1289,       2657,       381501,          245,         71447,    15088,      8359,      5631
 888,         1308,           422,          357,      9349,       1317,       2514,       392941,          249,         73314,    15342,      8505,      5808
 900,         1325,           427,          363,     11506,       1333,       2984,       403651,          252,         74912,    15579,      8625,      5894
 912,         1343,           433,          395,     11093,       1361,       2873,       414505,          256,         76422,    15819,      8762,      6119
 924,         1358,           436,          372,     11816,       1397,       3187,       425503,          259,         80108,    16054,      8883,      6011
 935,         1376,           443,          375,     12420,       1380,       3265,       435711,          264,         81541,    16273,      9002,      6063
 946,         1393,           449,          380,     11498,       1424,       3014,       446040,          268,         83475,    16485,      9118,      6222
 957,         1407,           452,          384,     10647,       1418,       2865,       456490,          271,         85303,    16702,      9232,      6280
 968,         1420,           454,          388,     11519,       1458,       3043,       467061,          272,         86865,    16918,      9350,      6328
 978,         1434,           458,          390,     11012,       1468,       2928,       476776,          277,         88417,    17112,      9455,      6373
 988,         1449,           463,          394,     12067,       1466,       3131,       486591,          280,         90899,    17307,      9559,      6456
 998,         1460,           464,          393,     11409,       1488,       3021,       496506,          281,         90623,    17501,      9671,      6655
Den totale tiden målt i mikrosekunder: 36217819.638 µs
```

Figur 5 - Outputen ved å kjøre filen 'nearly sorted' med 1000 inputs.

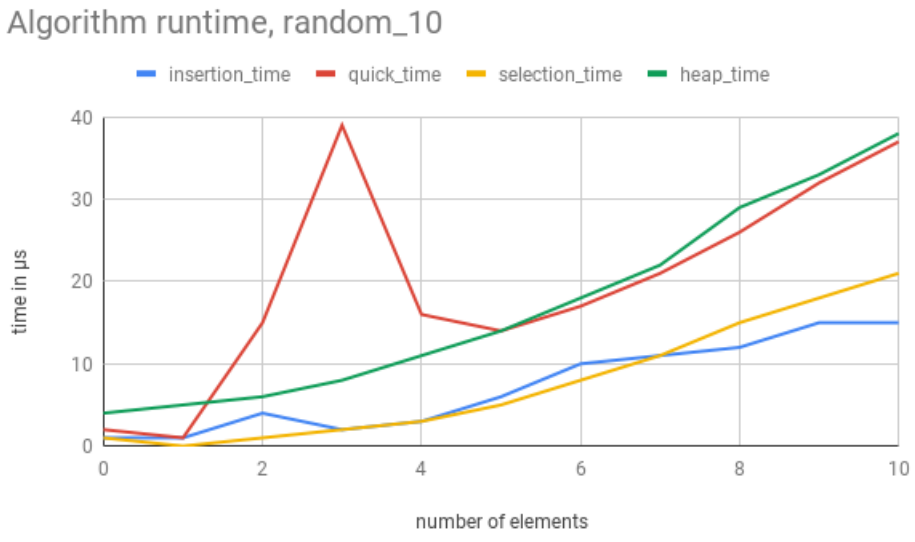
```
embrik@Embrik-Ubuntu:~/UiO/IN2010/innlevering/IN2010-Oblig3/python-scr$ python3 oblig3.py inputs/random_1000
   n, insertion_cmp, insertion_swaps, insertion_time, quick_cmp, quick_swaps, quick_time, selection_cmp, selection_swaps, selection_time, heap_cmp, heap_swaps, heap_time
 231,         13205,          12979,          5015,      2646,        616,        751,        26335,          224,          4539,     2926,      1611,      1100
 296,         22050,          21759,          8564,      3801,        771,       1025,        43365,          291,          7537,     3933,      2173,      1475
 342,         28539,          28202,         11154,      4191,        905,       1157,        57970,          337,         10206,     4718,      2591,      1809
 379,         35257,          34883,         13865,      4677,       1035,       1296,        71253,          367,         12623,     5326,      2930,      2021
 410,         42475,          42071,         16736,      5133,       1129,       1423,        83436,          405,         14852,     5871,      3213,      2259
 437,         46714,          46283,         18259,      5322,       1218,       1499,        94830,          427,         17029,     6325,      3449,      2389
 461,         52312,          51857,         20652,      5892,       1259,       1616,       105570,          453,         19619,     6752,      3679,      2526
 483,         57196,          56720,         22627,      6172,       1353,       1705,       115921,          475,         20959,     7123,      3872,      2700
 503,         61179,          60683,         24294,      6667,       1409,       1824,       125751,          496,         22638,     7499,      4089,      2810
 522,         66648,          66133,         27941,      6611,       1487,       1899,       135460,          515,         25014,     7833,      4262,      2979
 539,         71311,          70779,         28677,      7301,       1535,       1994,       144453,          530,         26521,     8152,      4442,      3070
 555,         75396,          74848,         30044,      7531,       1562,       2084,       153181,          549,         27954,     8422,      4567,      3280
 571,         79918,          79354,         31913,      7812,       1617,       2119,       162165,          561,         29421,     8709,      4747,      3616
 586,         85383,          84804,         34075,      7723,       1673,       2177,       170820,          573,         31834,     8997,      4895,      3406
 600,         90335,          89742,         36521,      8347,       1695,       2244,       179101,          592,         33834,     9274,      5044,      3504
 613,         94534,          93928,         37785,      8989,       1738,       2383,       186966,          603,         34095,     9500,      5145,      3559
 626,         98681,          98062,         39395,      8421,       1837,       2321,       195000,          618,         35440,     9731,      5280,      3726
 638,        102066,         101435,         40643,      9030,       1851,       2464,       202566,          628,         38637,     9957,      5412,      3861
 650,        105345,         104702,         42807,      8749,       1894,       2455,       210276,          645,         39249,    10174,      5516,      3849
 661,        108557,         107903,         43687,      9168,       1920,       2476,       217470,          648,         40543,    10389,      5621,      3917
 672,        111617,         110952,         45060,      9166,       1979,       2560,       224785,          661,         42072,    10594,      5735,      3995
 683,        114639,         113963,         46296,      9749,       1976,       2656,       232221,          676,         43193,    10792,      5823,      4123
 694,        117646,         116959,         47647,      9324,       2038,       2603,       239778,          684,         44301,    10969,      5942,      4211
 704,        119495,         118798,         48167,      9711,       2058,       2688,       246753,          695,         45525,    11162,      6029,      4304
 714,        122639,         121932,         49724,     10968,       2048,       2944,       253828,          707,         47598,    11370,      6169,      4287
 724,        126530,         125813,         51191,     10325,       2127,       2787,       261003,          718,         49185,    11563,      6265,      4421
 734,        130453,         129726,         53312,     10127,       2122,       2789,       268278,          725,         50546,    11758,      6353,      4431
 743,        134274,         133538,         54454,     10061,       2211,       2814,       274911,          740,         52944,    11927,      6457,      4497
 752,        138218,         137473,         56142,     10201,       2213,       2792,       281625,          744,         52670,    12117,      6554,      4619
 761,        140677,         139923,         59465,     10893,       2251,       2995,       288420,          750,         53429,    12282,      6635,      4741
 770,        143942,         143179,         58529,     11430,       2233,       3083,       295296,          765,         54998,    12426,      6725,      4753
 778,        147486,         146715,         59786,     10671,       2295,       2961,       301476,          768,         56478,    12581,      6795,      4806
 786,        149923,         149144,         61155,     10803,       2331,       3022,       307720,          778,         65000,    12739,      6885,      4890
 794,        153327,         152540,         62523,     11518,       2382,       3134,       314028,          787,         59109,    12898,      6972,      4861
 802,        156579,         155784,         64815,     10971,       2446,       3080,       320400,          796,         60521,    13067,      7059,      4929
 810,        159314,         158511,         64719,     11504,       2424,       3182,       326836,          802,         61978,    13200,      7129,      4978
 818,        163281,         162471,         67908,     11207,       2446,       3204,       333336,          811,         63875,    13369,      7220,      5098
 826,        166845,         166027,         67866,     11294,       2475,       3125,       339900,          818,         65520,    13510,      7303,      5319
```

```
833,        169199,         168374,         68964,      12207,      2456,       3318,       345696,         826,        65116,      13638,      7370,       5252
840,        172423,         171591,         70255,      11961,      2481,       3274,       351541,         833,        66143,      13821,      7472,       5293
847,        174932,         174093,         72674,      13014,      2468,       3726,       357435,         841,        67436,      13925,      7524,       5287
854,        177341,         176495,         71996,      12224,      2533,       3362,       363378,         846,        68279,      14056,      7597,       5455
861,        180702,         179849,         73657,      12261,      2570,       3378,       369370,         857,        70415,      14186,      7657,       5336
868,        183020,         182160,         74592,      13082,      2582,       3512,       375411,         865,        71503,      14321,      7725,       5384
875,        186335,         185468,         76246,      12206,      2638,       3374,       381501,         864,        73071,      14447,      7809,       5456
882,        190122,         189248,         77463,      13953,      2588,       3700,       387640,         873,        73291,      14581,      7861,       5524
889,        192951,         192070,         79635,      13955,      2623,       4266,       393828,         881,        74857,      14715,      7946,       5614
896,        195241,         194353,         80478,      12738,      2717,       3532,       400065,         888,        75163,      14809,      7975,       5772
902,        197547,         196653,         81840,      14531,      2697,       3868,       405450,         892,        75680,      14951,      8058,       5726
908,        200799,         199899,         81964,      12730,      2755,       3546,       410871,         898,        77220,      15038,      8111,       5822
914,        204289,         203383,         83138,      14005,      2735,       3734,       416328,         906,        78359,      15201,      8195,       5776
920,        206461,         205549,         85052,      13724,      2744,       3825,       421821,         911,        79746,      15329,      8270,       5857
926,        209680,         208762,         85445,      12995,      2772,       4794,       427350,         918,        82068,      15429,      8312,       5828
932,        211100,         210176,         86169,      13890,      2801,       3759,       432915,         927,        82276,      15548,      8386,       5880
938,        215454,         214524,         88398,      13486,      2880,       3689,       438516,         929,        83445,      15693,      8456,       5962
944,        217404,         216468,         88573,      13855,      2847,       3773,       444153,         940,        84367,      15770,      8485,       5994
950,        219796,         218854,         89902,      13676,      2906,       3759,       449826,         940,        85454,      15919,      8593,       6060
956,        222631,         221683,         92737,      13172,      2913,       3745,       455535,         946,        86007,      16037,      8656,       6151
962,        224886,         223932,         92490,      14465,      2883,       3922,       461280,         954,        86970,      16140,      8704,       6222
968,        226856,         225896,         93494,      14230,      2894,       3868,       467061,         960,        87860,      16278,      8769,       6320
974,        230980,         230014,         95222,      14819,      2933,       4006,       472878,         961,        89765,      16385,      8832,       6278
980,        234111,         233139,         96450,      14084,      2929,       3868,       478731,         975,        91094,      16501,      8900,       6388

Giving up on insertion

989,                ,               ,               ,   14696,      3007,       4003,       487578,         978,        97620,      16661,      8954,       6751

Giving up on selection

Den totale tiden målt i mikrosekunder: 67214198.694 µs
```

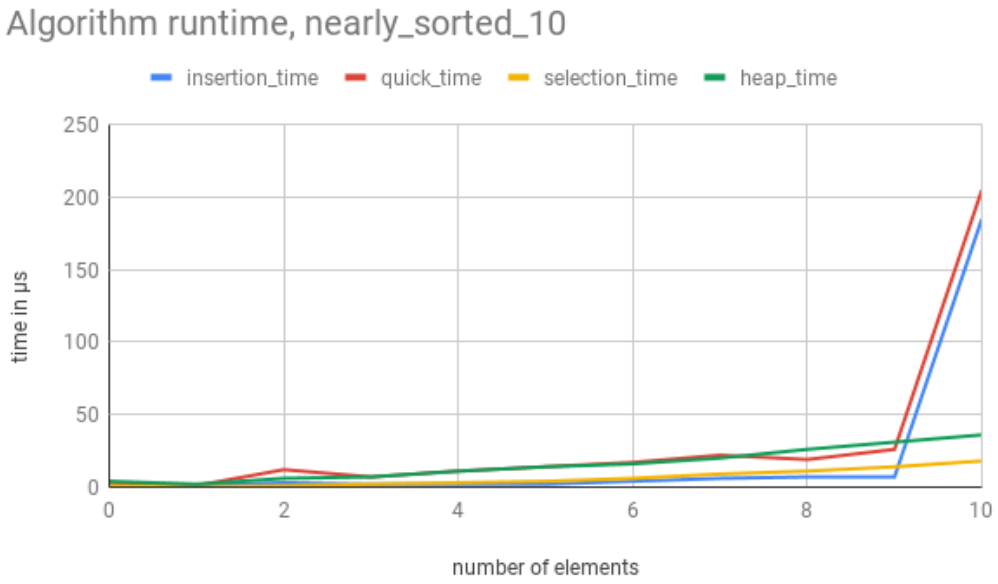
Figur 6 - Outputen ved å kjøre filen 'random' med 1000 inputs.

Deloppgave 3: Eksperimentér

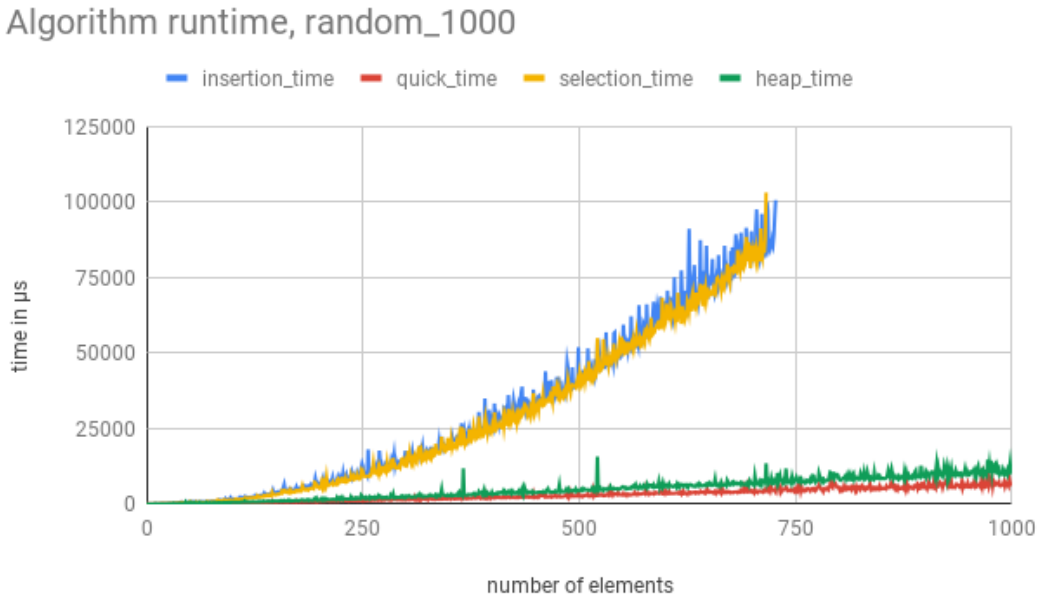
Nedenfor ser man ulike linjediagrammer som svarer til kjøretid for de ulike sorteringsalgoritmene hvor man ser hvordan antall inputs påvirker kjøretiden. Disse algoritmene er kjørt på både 'nearly sorted' og 'random' slik at man får et innblikk i når de ulike algoritmene er mest eller minst effektive. Dersom grafen som representerer en gitt sorteringsalgoritme plutselig stopper opp, så betyr det at kjøretiden for spesifikt et input bruker for lang tid i forhold til timelimiten som er oppgitt i prekoden. Da vil sorteringsalgoritmen breake, og vil dermed ikke fortsette å kjøre.



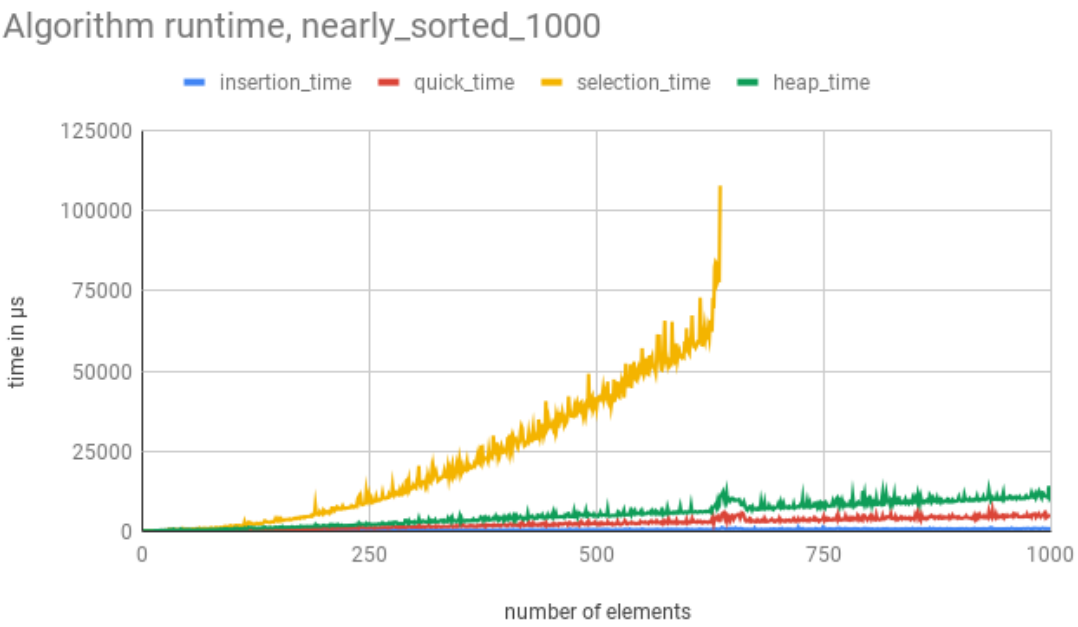
Figur 7 - Kjøretiden på de ulike sorteringsalgoritmene brukt på random-filen med 10 inputs.



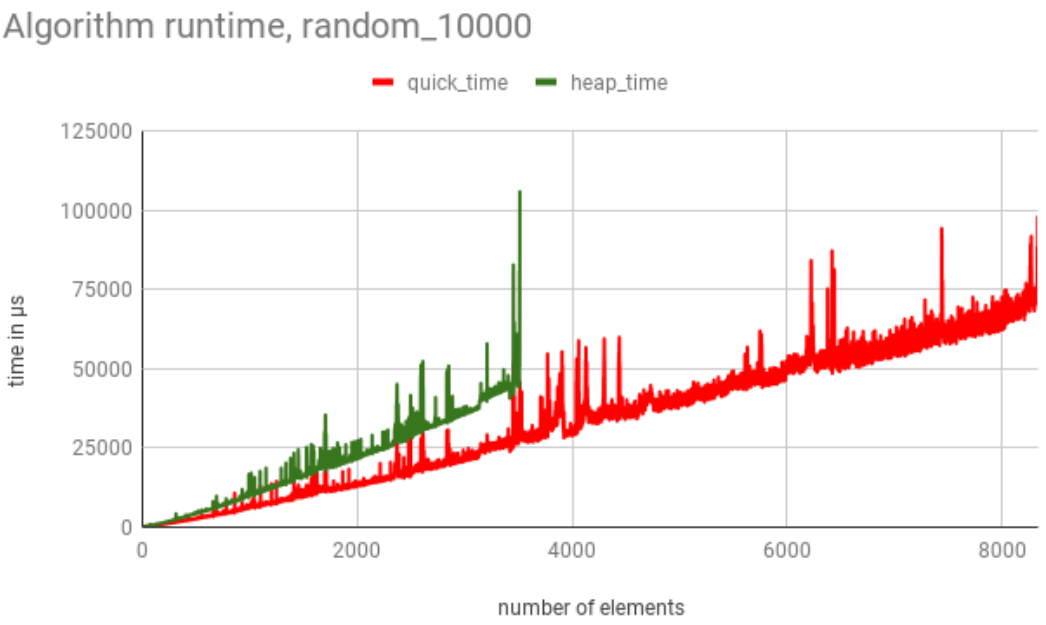
Figur 8 - Kjøretiden på de ulike sorteringsalgoritmene brukt på nearly-sorted-filen med 10 inputs.



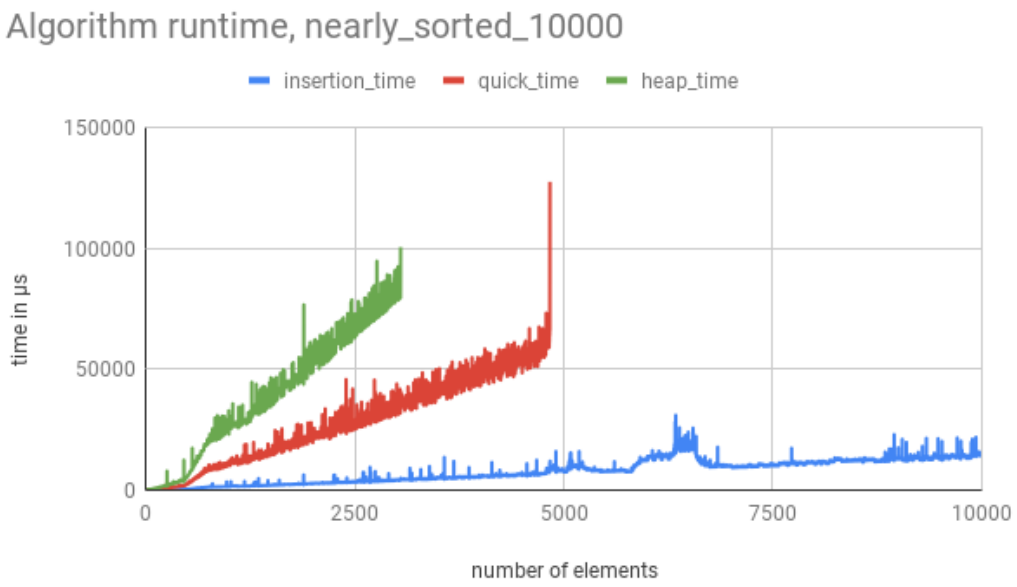
Figur 9 - Kjøretiden på de ulike sorteringsalgoritmene brukt på random-filen med 1000 inputs.



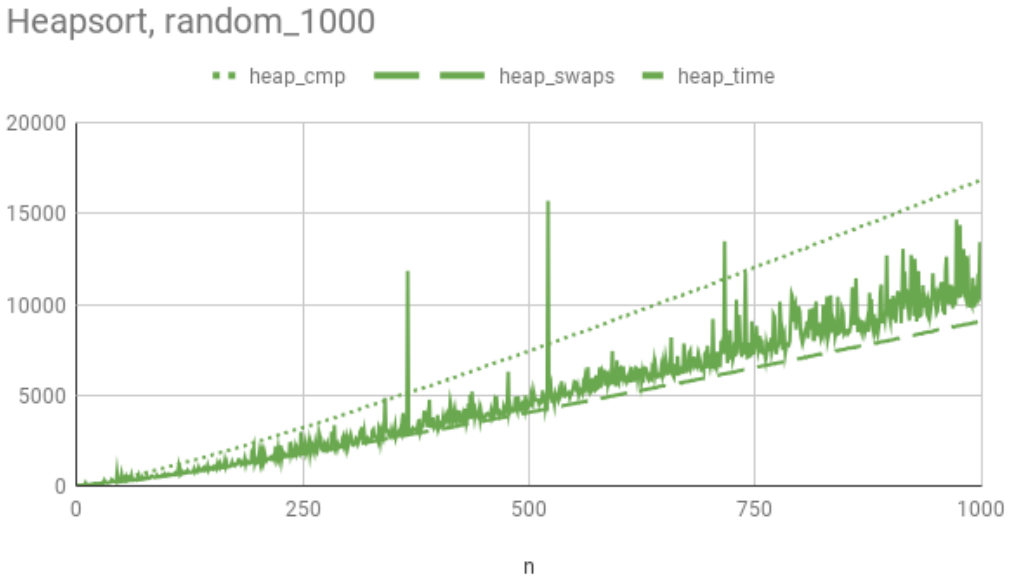
Figur 10 - Kjøretiden på de ulike sorteringsalgoritmene brukt på nearly-sorted-filen med 1000 inputs.



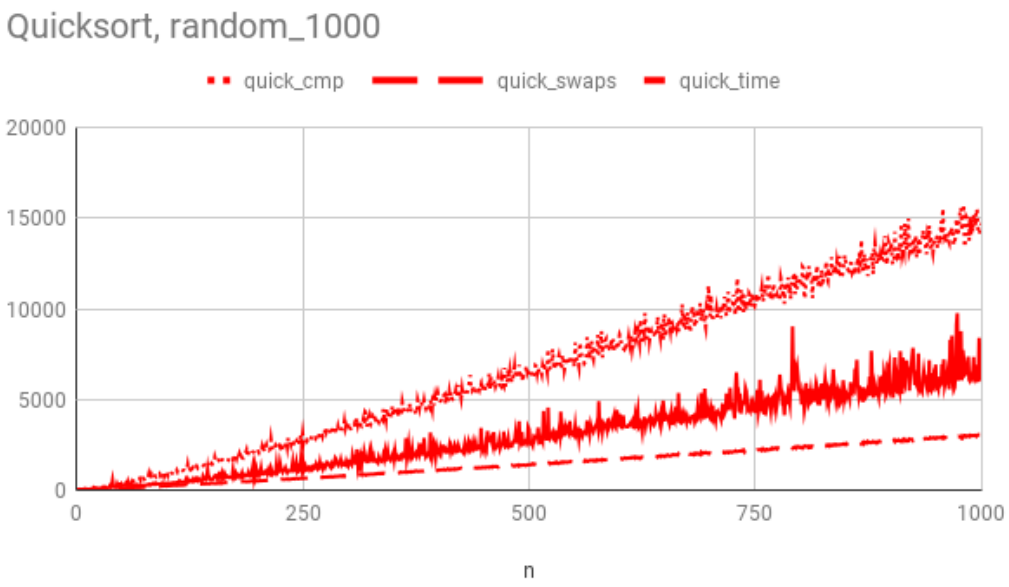
Figur 11 - Kjøretiden på de ulike sorteringsalgoritmene brukt på random-filen med 10000 inputs.



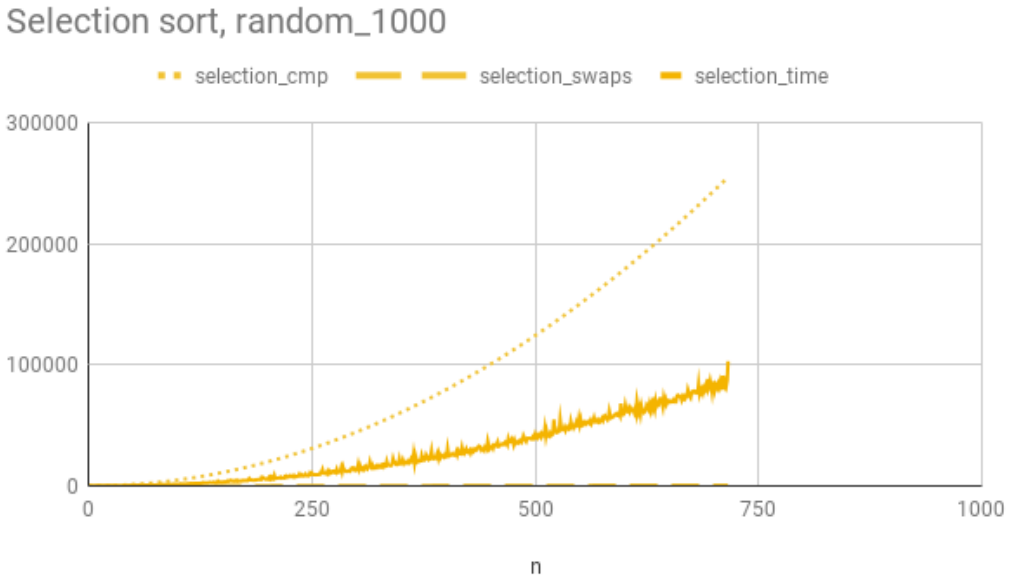
Figur 12 - Kjøretiden på de ulike sorteringsalgoritmene brukt på nearly-sorted-filen med 10000 inputs.



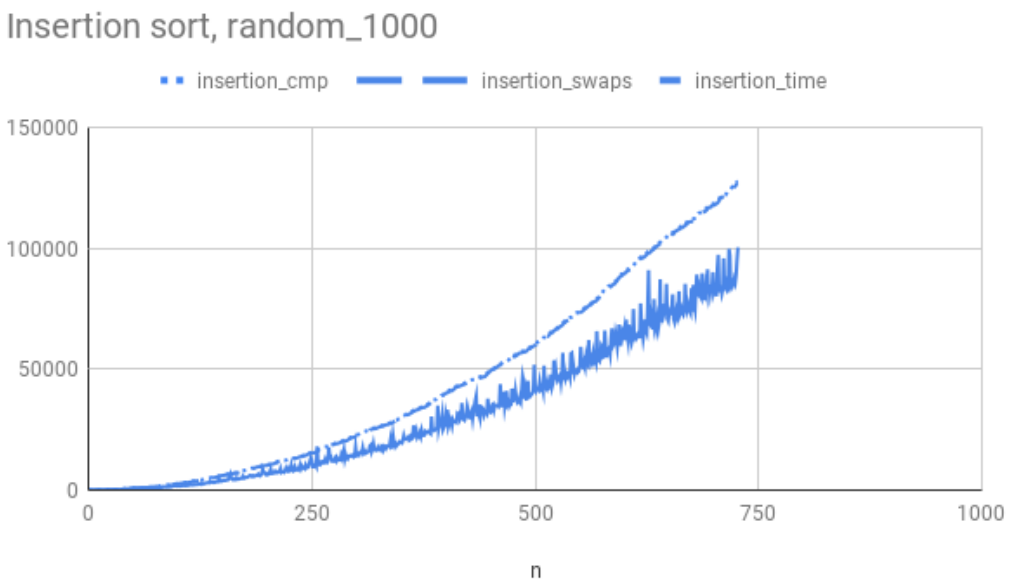
Figur 13 - Antall sammenligninger, bytter og kjøretiden for heapsort brukt på random-filen med 1000 inputs.



Figur 14 - Antall sammenligninger, bytter og kjøretiden for quicksort brukt på random-filen med 1000 inputs.



Figur 15 - Antall sammenligninger, bytter og kjøretiden for selection sort brukt på random-filen med 1000 inputs.



Figur 16 - Antall sammenligninger, bytter og kjøretiden for insertion sort brukt på random-filen med 1000 inputs.

Hvilke sorteringsalgoritmer som utmerker seg positivt er avhengig av hva slags inputfil vi tester med. Dersom det er random-filen med 10 inputs, ser vi fra figur 7 at både insertion sort og selection sort gjør det best. Dersom det er nearly-sorted-filen med 10 inputs ser vi fra figur 8 at heapsort og selection sort gjør det best. Ved å se på disse to resultatene så vil det i gjennomsnitt være heapsort og selection sort som tar kortest tid.

På den andre siden kan vi se på hvilke sorteringsalgoritmer som er raskest dersom n er svært stor. Fra figur 9 observerer vi at for random-filen med relativt få inputs så har både insertion sort og selection sort svært høy kjøretid. Vi ser derfor bort fra disse sorteringalgoritmene for en stor n. Med samme argument ser man fra figur 10 at selection sort har lang kjøretid for nearly-sorted-filen. Dette fører oss til figur 11 og figur 12. Vi observerer her at quicksort er raskest for random-filen med 10000 inputs, ettersom heapsort bruker for lang tidd etter 3000 inputs og ender med å stoppe grunnet timelimiten vår. Derimot ser vi at insertion sort er raskest for nearly-sorted-filen med 10000 inputs. Denne er mye raskere enn de to andre, og klarer å bruke kort tid for hver input. I likhet med figur 11 ser vi at quicksort og heapsort også stopper etter x antall inputs.

Dersom vi fokuserer på random-filen med 10000 inputs ser vi at quicksort og heapsort er relativt like med tanke på kjøretid. Den eneste store forskjellen er at heapsort blir avsluttet når n er litt over 3000. Siden både heapsort og quicksort har kjøretidskompleksitet $O(n\ log\ n)$ stemmer dette relativt bra med forventet resultat. På den andre siden stemmer ikke dette hvis vi ser på nearly-sorted-filen med 10000 inputs. Her har vi insertion sort som den raskeste som

fullfører og er betraktelig raskere enn både quicksort og heapsort. Insertion sort har kjøretidskompleksitet $O(n^2)$ mens de to andre har kjøretidskompleksitet $O(n\ log\ n)$. Dette er på grunn av at filen er nesten sortert fra før av, og følgende passer insertion svært godt for å løse det problemet. For å sammenligne kan vi referere til figur 9 som viser kjøretiden for alle sorteringsalgoritmene brukt på random-filen med 1000 inputs. Her ser vi at insertion sort og selection sort er omtrent like raske og at quicksort og heapsort er omtrent like raske. I tillegg er både heapsort og quicksort betydelig raskere enn de to andre sorteringsalgoritmene. Dette stemmer godt overens med kjøretidskompleksiteten på de ulike algoritmene.

ANTALL SAMMENLIGNINGER OG ANTALL BYTTER KORRELERT MED KJØRETID