UNIVERSITETI I PRISHTINËS

FAKULTETI I SHKENCAVE MATEMATIKORE DHE NATYRORE

DEPARTAMENTI I MATEMATIKËS

Programi: Shkencë Kompjuterike

LËNDA: Analizë e algoritmeve

TEMA: Analizë dhe Optimizim i disa Algoritmeve të sortimit

Profesori:

Elver Bajrami

Besnik Duriqi

Studenti:

Erlisa Lokaj

Ermira Haziri

Përmbajtja

Abstrakt

Në këtë punim kemi bërë krahasimin në mes të MergeSort-2 Way, MergeSort-3 Way dhe Bottom-Up MergeSort, si teorikisht ashtu edhe empirikisht, për të zbuluar raportin në mes këtyre çasjeve të implementimit, për madhësi të ndryshme të vargjeve të gjeneruar rastësisht. Dihet se në teori Bottom-Up ka kompleksitet më të ulët, dhe ne do të provojmë nëse një tezë e tillë qëndron apo jo, duke e implementuar dhe testuar atë (duke na dërguar në një optimizim të mundshëm).
Në këtë punim gjithashtu do të analizohen algoritmet e sortimit MergeSort dhe QuickSort si dhe variantët e tij me qëllim optimizimi. Këta algoritme do të analizohen për madhësi të ndryshme të hyrjeve duke fituar rezultate të ndryshme sa i përket kohës së ekzekutimit.

Përshkrimi i algoritmëve

MergeSort - është një algoritëm renditjeje që funksionon duke e ndarë një grup në nënvargje më të vogla, duke renditur çdo nëngrup dhe më pas duke bashkuar nëngrupet e renditura përsëri së bashku për të formuar grupin përfundimtar të renditur.
Me fjalë të thjeshta, mund të themi se procesi i mergesort është të ndash grupin në dy gjysma, të renditësh secilën gjysmë dhe më pas të bashkosh përsëri gjysmat e renditura. Ky proces përsëritet derisa i gjithë grupi të renditet.
Kompleksiteti kohor : $O(n\ log\ n)$
Kompleksiteti hapsinor: $O(n)$

Algoritmi MergeSort

Hapi 1: Fillo
Hapi 2: Deklaro vargun dhe variblat majtas, djathtas dhe në mes.
Hapi 3: Kryej funksionin merge

```
If majtas > djathtas
   Return
 Mes = (majtas + djathtas)/2
 Mergesort(vargu, majtas,mes)
 Mergesort(vargu, mes +1 , djathtas)
 Mergesort(vargu, majtas, mes, djathtas)
```

Hapi 4: Stop

MergeSort 3-way – Ky algoritëm është një variant i algoritmit MergeSort ku në vend që ta ndajmë grupin në 2 pjesë e ndajmë në 3 pjesë. Pseudokodi është i ngjashëm me MergeSort.
Kompleksiteti kohor: $O(n \log_3 n)$
Kompleksiteti hapsinor**:** $O(n)$

Bottom-up MergeSort – është një variant i algoritmit MergeSort i cili implementohet në mënyrë rekurzive në një qasja nga poshtë-lartë. Nuk ka nevojë të ruaj thirrjet e funksionit në varg andaj ky algoritëm ka hapsirë më efikase deri në 25%.
Kompleksiteti kohor : $O(n\ log\ n)$

Algoritmi Bottom-up Merge Sort

```
Funksioni sort(arr[0… n-1])
        n ← madhësia e vargut
        krijo c[0… n-1]
```

```
len ← 1
while len < n do
        low ← 0
        while low < n- len do
                mid ← low + len -1
                high ← min(low+ 2*len -1, n-1)
                merge(arr, c, low + 2*len)
                low ← low + 2* len
      len ← len *2
```

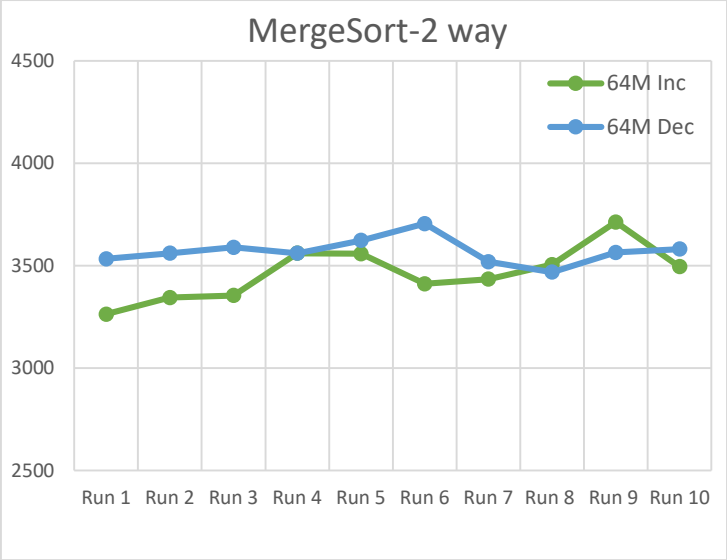
Krahasimi në mes të MergeSort-2 way dhe MergeSort 3-way

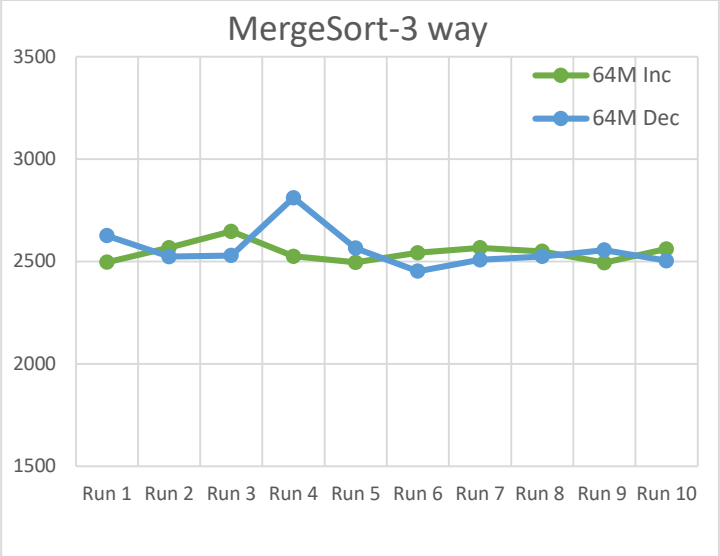
Për të hulumtuar ndryshimet në performancë në mes këtyre dy algoritmeve, i bëjmë 10 testime të njëpasnjëshme (run 1-10), për madhësi të ndryshme të vargjeve të gjeneruara rastësisht (4M-128M), ku vlera e bërthamës (seed) merr vlera nga 0-9000, për MergeSort-2way dhe MergeSort 3-way respektivisht. Në tabelën më posht është paraqitur koha (në milisekonda) që marrin këto testime të njëpasnjëshme.

2- Way	seed	4M (ms)	8M (ms)	16M (ms)	32M (ms)	64M (ms)	128M (ms)	64M(inc) (ms)	64M(dec) (ms)
Run 1	0	640	1265	2658	5765	12146	25705	3263	3633
Run 2	1000	670	1303	2624	5739	12470	25525	3344	3560
Run 3	2000	682	1255	2661	5801	12392	25644	3354	3589
Run 4	3000	664	1282	2789	5943	12692	25255	3560	3560
Run 5	4000	664	1262	2852	6004	12688	25866	3558	3623
Run 6	5000	674	1335	2886	5938	12742	25021	3411	3705
Run 7	6000	613	1326	2943	5887	12839	25706	3434	3519
Run 8	7000	648	1325	2850	5822	12970	25973	3505	3468
Run 9	8000	652	1284	2814	5875	12981	26169	3713	3564
Run 10	9000	618	1249	2765	5806	12613	24264	3495	3581
Mesatarja		652.5	1288.6	2784.2	5858	12653.3	25512.8	3463.7	3380.2

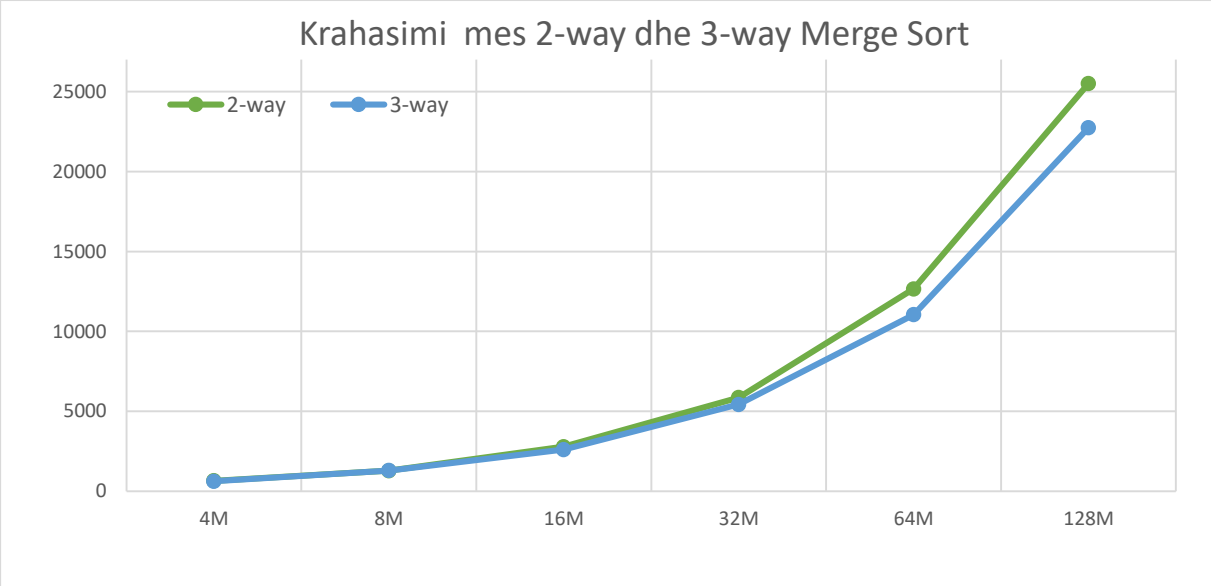
3-way	seed	4M (ms)	8M (ms)	16M (ms)	32M (ms)	64M (ms)	128M (ms)	64M-Inc (ms)	64M-Dec (ms)
Run 1	0	621	1292	2568	5394	11123	23670	2496	2626
Run 2	1000	578	1323	2563	5465	11049	23569	2567	2523
Run 3	2000	608	1313	2537	5474	11170	22648	2647	2529
Run 4	3000	617	1324	2745	5585	11158	22737	2525	2811
Run 5	4000	619	1384	2541	5617	11044	22844	2495	2565
Run 6	5000	620	1273	2630	5417	10762	22972	2542	2452
Run 7	6000	627	1301	2622	5476	11035	22501	2566	2508
Run 8	7000	679	1222	2610	5260	11073	22280	2549	2524
Run 9	8000	622	1260	2572	5295	11008	22363	2494	2555
Run 10	9000	597	1218	2632	5287	11006	21919	2561	2503
Average		618.8	1291	2602	5427	11042.8	22750.3	2544.2	2559.6

Grafet bazuar në këtë tabelë gjithashtu shihen në vijim:





Krahasimi i këtyre dy algoritmeve vizualisht në një grafik shpërndarjeje duke përdorur rezultatet mesatare të marra për grupet e rastësishme të madhësive 4M deri në 128M



Sipas rezultateve, çfarë mund të themi për pohimet teorike për secilin prej efikasitetit të algoritmit?

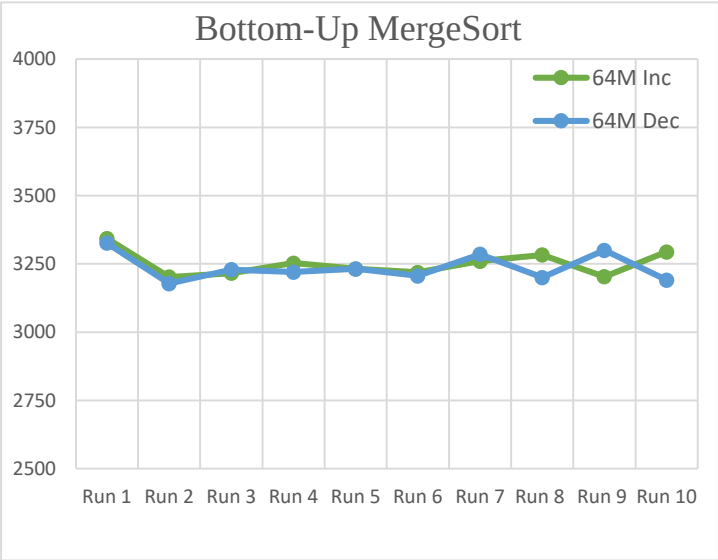
Për 64 milionë madhësi të vargut , vargjet rritëse dhe zvogëluese krijohen dhe pastaj vargjet sortohen me algoritmet MergeSort 2-way apo 3-way. Duke u bazuar në grafin e parë dhe të dytë, vargjet rritëse dhe zvogëluese sortohen në kohë të ngjajshme. Kjo situatë është e njëjtë për të dy algoritmet. Krijohen vargje të rastësishme me nga 4 milion deri ne 128 milion elemente. Sipas grafit të tretë, MargeSort 3-way kërkon më pak kohë se ai 2-way. Kjo është për shkak se kompleksiteti kohor i MergeSort 3-way është $O(n \log_3 n)$. Pra, mund të themi se 3-way MergeSort është me efikas për madhësi të mëdha të vargut. Ky rezultat është ashtu siç pritej me pohimet teorike.

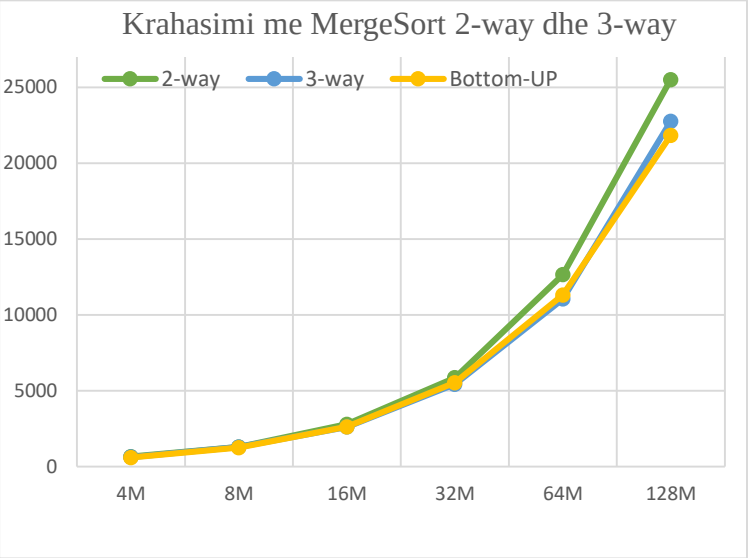
Krahasimi në mes të 2-Way dhe 3-way Merge Sort me Bottom-Up Merge Sort

Më poshtë gjendet tabela për testimet e kryera (të njejta me ato të bëra për algoritmet paraprake) për Bottom-Up MergeSort.

Bottom-up	seed	4M (ms)	8M (ms)	16M (ms)	32M (ms)	64M (ms)	128M (ms)	64M(inc) (ms)	64M(dec) (ms)
Run 1	0	572	1245	2650	5579	11552	24495	3343	3326
Run 2	1000	561	1252	2582	5621	11783	24225	3202	3178
Run 3	2000	594	1225	2575	5534	11339	22263	3216	3229
Run 4	3000	601	1256	2581	5629	10949	21238	3253	3220
Run 5	4000	589	1237	2603	5552	11057	21137	3232	3232
Run 6	5000	585	1234	2556	5584	11171	20753	3219	3206
Run 7	6000	600	1232	2652	5621	11107	21077	3260	3285
Run 8	7000	586	1262	2767	5388	11356	21101	3283	3200
Run 9	8000	586	1292	2603	5393	11412	20897	3203	3300
Run 10	9000	601	1214	2641	5401	11361	21027	3294	3191
Mesatarja		587.5	1244.9	2621	5530.2	11308.7	21821.3	3250.5	3236.7

Krahasimi i Bottom-Up Merge Sort 64M rritës dhe zvogëluesv(grafi në të majtë), dhe krahasimi me MergeSort 2-way dhe 3-way (grafi në të djathtë)





A siguron ndonjë përmirësim të performancës mbi MergeSort të zakonshëm?

Sipas rezultateve, përgjigja e shkurtër është po. Algoritmi Bottom-Up Merge Sort siguron p*ërmirësim* të performancës mbi algoritmin e zakonshëm MergeSort (2-way). Edhe pse të dy algoritmet kanë të njëjtin kompleksitet kohor, Bottom-Up merr më pak kohë. Kjo ndodh sepse Bottom-Up është një algoritëm iterativ ndërsa Merge Sort është një algoritëm rekurziv. Algoritmet rekurzive duhet ti ruajnë thirrjet e funksionit në stack, gjë që e dallon nga algoritmet iterative. Gjithashtu, algoritmet përsëritëse janë përgjithësisht më të shpejtë se algoritmet rekurzive. Për shkak të kësaj, mund të themi se Bottom-Up Merge Sort është më efikas. Ky rezultat është siç pritej.

Krahasimi në mes të InsertionSort , QuickSort dhe optimizimit të tij

Një zbatim hibrid i QuickSort me një optimizim me përdorimin e Insertion Sort kur ndarjet marrin një madhësi të caktuar është bërë në vijim. Algoritmi InsertionSort është përcaktuar të thirret kur madhësia e ndarjeve arrin 10. Duke parë tabelën, ne shohim menjëherë përfitimet e përdorimit të Insertion Sort nga quickInsertionSort1 në ndarje të vogla, pasi ai tejkalon dukshëm QuickSort në madhësi rreth 100. Duket se ruan rastin mesatar të O(nlogn). InsertionSort u aplikua në të gjitha ndarjet në grup së bashku dhe jo individualisht.

Parametrat	QuickSort	QuickInsertSort1	QuickInsertSort2
Koha për n=10	0.1014	0.0122	0.017
Koha për n=50	0.016944	0.0181	0.0165
Koha për n=100	0.03134	0.03449	0.03501
Koha për n=250	0.110405	0.058979	0.05913
Koha për n=500	0.187842	0.06874	0.096072
Koha për n=1000	0.39388	0.193305	0.170573
Koha për n=1500	0.506569	0.135763	0.29776
Koha për n=2000	0.702158	0.264855	0.379141
Koha për n=2500	0.766395	0.319759	0.23633
Koha për n=3000	0.813225	0.381438	0.195207
Koha për n=3500	0.922858	0.465465	0.243442
Koha për n=4000	1.209174	0.504231	0.26219
Koha për n=4500	1.030819	0.491554	0.392979
Koha për n=5500	1.259988	0.488271	0.556112
Koha për n=6500	1.274577	0.623492	0.646929