CSC2016 - Algorithm Design and Analysis

Assignment 1 : Implementing and Comparing Searching Algorithms

Borja De La Viuda
110320041
Monday 3 December 2012
Discussion

Test 1:

Key	Index	Hash Index	Sequential Search Comp	Binary Search Comp	Hash Search Comp
18	-1	-1	12	18	4
69	-1	-1	38	16	8
201	-1	-1	94	17	4
331	-1	-1	132	15	46
492	-1	-1	196	16	26
17	4	17	9	17	1
67	16	67	33	16	1
209	48	58	97	14	1
372	72	70	145	12	1
498	98	49	197	11	8

Observations from test 1:

- From the tests it seems that Sequential and Binary searches produce a similar number of comparisons when the number is in the beginning of the array (or should be, in the case of 20 in test 1).
- In cases where the key is near the very beginning Sequential can even be more adequate than using Binary, however not Hash search.
- If the key is in the array, Hash seems to be overall the best choice, as it consistently proves to have fewer number of comparisons, however in certain cases where the key isn't present Binary proved to have stopped needed less comparisons to report the key missing.
- it seems that for key 331 and 492, the Hash search has performed the worst compared from all the keys, although it still didn't have more comparisons than sequential search.

Test 2:

Number	Index	Hash Index	Sequential Search Comp	Binary Search Comp	Hash Search Comp
20	-1	-1	18	28	8
832	-1	-1	614	25	8
1452	-1	-1	1084	25	4
1937	-1	-1	1458	24	4
2615	-1	-1	1940	25	4
87	28	87	57	22	1
851	308	851	617	23	1
1350	496	1350	993	21	1
1990	747	491	1495	18	1
2631	973	1133	1947	22	2

Observations from test 2:

- Even in a larger array, Binary search seems to maintain a similar number of comparisons each time.
- The Hash search didn't need too many comparisons to find the key, even with the larger array.

Search Algorithm	**Test 1 Total Comparisons**	**Test 1 Avg Comparisons**	**Test 2 Total Comparisons**	**Test 2 Avg Comparisons**
Sequential Search	953	95	11176	1117
Binary Search	152	15	385	38
Hash Search (LP)	100	10	134	13

Overall Discussion:

- As expected sequential search requires many more comparisons than the other two searches in total, Binary the second best and Hash being the one with the least number of comparisons

- The number of comparisons of sequential search clearly follows it's limitations: it has a decent performance when the key is near the beginning (Big O(1)) like in the cases of key = 17 in test one, however it suffers greatly if the key is towards the end of the array (as it then has a big O(n)) as the with key = 2631 in test 2. Therefore it reaffirms the idea that Sequential search is good for small arrays.
- Binary search seems to perform the similar number of comparisons regardless of where in the array the number could be, as such it's much more effective when it comes to searching larger arrays than sequential search.
- In these tests, the best case of hashing has been in most cases, finding the key in one comparison (thus in big O (1)) and even in the cases where it didn't, it usually still needed less comparisons than the other two search methods. As such it seems to be overall the quickest search algorithm of the three.

Test Number	Number to mod by:	Total Number of Collisions when entering the data into Hash array	% of collisions
Data 1	151	34	34%
Data 2	1499	275	27.5%

- In terms of the collisions when entering the data into the hash array, it is interesting to note that the second number, 1499, produced even less key collisions than 151 for the first test. This could explain why overall hashing had an even lower number of comparisons in test 2. It could also mean that in fact hashing is more suited to very large arrays, as long as the hash function is good.