Homework 5 Solutions

CSc 345 – Summer 2014

Problems (55 points possible)

1. (20) External Sorting. Consider this sequence of keys, in the order shown:

   7, 109, 131, 53, 173, 73, 61, 43, 101, 157, 137, 3, 67, 97, 37, 113, 103, 29, 149,
   47, 197, 223, 167, 17, 2, 181, 79, 83, 151, 71, 23, 229, 191, 227, 89, 193, 31, 11,
   211, 179, 5, 59, 107, 41, 13, 163, 139, 127, 19, 199

   Use each of the following sorts to sort this list:

   - (10) 3-way external merge sort with 3 input files and 3 output files (so that the input and output roles can switch for each pass). At the end of each pass, show the content of the output files.

```
Pass 0:
File 1:  7   53  61 157  67 113 149 223   2  83  23 227  31 179 107 163  19
File 2: 109 173  43 137  97 103  47 167 181 151 229  89  11   5  41 139 199
File 3: 131  73 101   3  37  29 197  17  79  71 191 193 211  59  13 127

Pass 1:

File 4: (  7 109 131) (  3 137 157) ( 47 149 197) ( 71  83 151)
        ( 11  31 211)  (127 139 163)
File 5: ( 53  73 173) ( 37  67  97) ( 17 167 223) ( 23 191 229)
        (  5  59 179)  ( 19 199)
File 6: ( 43  61 101) ( 29 103 113) (  2  79 181) ( 89 193 227)
        ( 13  41 107)

Pass 2:
File 1: (  7  43  53  61  73 101 109 131 173)
        ( 23  71  83  89 151 191 193 227 229)
File 2: (  3  29  37  67  97 103 113 137 157)
        (  5  11  13  31  41  59 107 179 211)
File 3: (  2  17  47  79 149 167 181 197 223)
        ( 19 127 139 163 199)

Pass 3:
File 4: (  2   3   7  17  29  37  43  47  53  61  67  73  79
           97 101 103 109 113 131 137 149 157 167 173 181 197 223)
File 5: (  5  11  13  19  23  31  41  59  71  83  89 107 127
          139 151 163 179 191 193 199 211 227 229)

Pass 4:
File 6: (   2   3   5   7  11  13  17  19  23  29  31  37  41  43  47  53  59
           61  67  71  73  79  83  89  97 101 103 107 109 113 127 131 137 139
          149 151 157 163 167 173 179 181 191 193 197 199 211 223 227 229 )
```

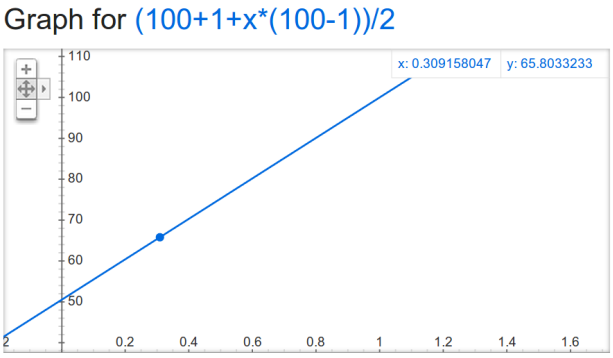
- (10) Improved External Merge Sort (assume that we have four buffers, each with room for four values). Clearly indicate the content of the runs at the end of each pass, including pass 0.

```
  7 109 131  53 173  73  61  43 101 157 137   3  67  97  37 113
103  29 149  47 197 223 167  17   2 181  79  83 151  71  23 229
191 227  89 193  31  11 211 179   5  59 107  41  13 163 139 127
 19 199

Pass 0:
(  3   7  37  43  53  61  67  73  97 101 109 113 131 137 157 173)
(  2  17  23  29  47  71  79  83 103 149 151 167 181 197 223 229)
(  5  11  13  31  41  59  89 107 127 139 163 179 191 193 211 227)
( 19 199)

Pass 1:
(  2   3   5   7  11  13  17  23  29  31  37  41  43  47  53  59
  61  67  71  73  79  83  89  97 101 103 107 109 113 127 131 137
 139 149 151 157 163 167 173 179 181 191 193 197 211 223 227 229)

( 19 199)

Pass 2:
(  2   3   5   7  11  13  17  19  23  29  31  37  41  43  47  53
  59  61  67  71  73  79  83  89  97 101 103 107 109 113 127 131
 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211 223
 227 229)
```

2. (5) Consider the basic Quicksort algorithm, but with the pivot always chosen to be the key value at index $\lfloor \frac{n}{2} \rfloor$, where n in the quantity of values in whichever range is being partitioned currently. Describe the kind of sequences that would force this variant of Quicksort to run in $\Theta(n^2)$ time.

The worst case is would be having the largest elements in alternating descending order (e.g. 4 6 8 10 9 7 5) starting at $\lfloor \frac{n}{2} \rfloor$. Hand-wavy proof: suppose the largest element is at $\lfloor \frac{n}{2} \rfloor$ in an odd-length array. Then the next call to Quicksort will be to $\lfloor \frac{n-1}{2} \rfloor$, which is where the next largest value will be. This forces Quicksort to always pick the largest values and thus make the recurrence $T(n) = T(n-1) + n$, which is $\Theta(n^2)$.

3. (5 points) Linear search



From the graph, we can see that the expected cost grows linearly as the probability of an unsuccessful search increases.

4. (25 points) Hashing

   (a) (5) 9.13(a,b)

   - (a) Not acceptable. if $k = n^2 + n$, then it would hash to $n + 1$, which is a not an index on the hash table.
   - (b) Acceptable, but not good. All values go to 1, which will guarantee collisions leading to linear search/insertion.

   (b) (5) Bloom Filters. Let $m = 17, h_1(x) = (k + 15)\%m, h_2(x) = (4k + 11)\%m$, and $h_3(x) = (7k + 2)\%m$. Insert the keys 23, 7, 50, and 91 into the bit vector, and show the resulting vectors content. Then, find a key that is a false positive; that is, find a key that appears to have been inserted, but wasnt.

Initially, we have a zero bit-vector.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

23 hashes to 4 1 10, 7 hashes to 5 5 0, 50 hashes to 14 7 12, and 91 hashes to 4 1 10. So, we then get

1	1	0	0	1	1	0	1	0	0	1	0	1	0	1	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Next, we try to find a key that has gives us a false positive. Hashing 6 gives us 4 1 10, which is the same signature as 23 and 91. So, we found a key that gives us a false positive.

(c) (5) Consider a hash table with 13 slots. Draw the content of this hash table after hashing the values 72, 61, 80, 43, 17, 91, 13, 4, 10, and 1 using the hash function $h'(k) = (3k+5)\%13$. Assume collisions are handled using chaining.

k	$h'(k)$
72	0
61	6
80	11
43	4
17	4
91	5
13	5
4	4
10	9
1	8

Assume a sequence of values across a column = linked list of values

0	72		
1			
2			
3			
4	43	17	4
5	91	13	
6	61		
7			
8	1		
9	10		
10			
11	80		
12			

(d) (5) Repeat 4c using quadratic probing and this function: $h(k, i) = (h(k) + i + 2i^2)\%13$. (Note that quadratic probing is a type of open addressing; chaining is not used.) If not all values can be stored, show the content of the table at the point of the failure, and explain the problem.

k	i	$h(k, i)$
72	0	0
61	0	6
80	0	11
43	0	4
17	1	7
91	0	5
13	1	8
4	2	1
10	0	9
1	3	3

0	72
1	4
2	
3	1
4	43
5	91
6	61
7	17
8	13
9	10
10	
11	80
12	

(e) (5) Rehash the content of the resulting table from 4d to a table of 18 slots using linear probing ($h(k,i) = (h'(k) + i)\%18$) where $h'(k) = (3k + 5)\%18$. Take the values in the order they are encountered in 4d's table, starting with slot 0. Then, hash into the new table any values that you were unable to store in 4d's table, and then hash the values 12 and 69. Show the table as it exists after the content from 4d's table has been rehashed, and again after 69 has been hashed.

k	i	$h(k,i)$
72	0	5
4	0	17
1	0	8
43	1	9
91	2	10
61	3	11
17	0	2
13	4	12
10	1	0
80	2	13
12	1	6
69	1	15

0	10
1	
2	17
3	
4	
5	72
6	
7	
8	1
9	43
10	91
11	61
12	13
13	80
14	
15	
16	
17	4

0	10
1	
2	17
3	
4	
5	72
6	12
7	
8	1
9	43
10	91
11	61
12	13
13	80
14	69
15	
16	
17	4