Algorithms: Homework 2

Li-Yuan Wei

May 8, 2023

Problem 1

Using Figure 8.3 as a model, illustrate the operation of RADIX-SORT on the following list of English words: NOD, HOG, SHY, BAN, BAR, JET, EBB, PAR, ASH, PET, TED, ROT, FIG.

0(*initial*)	*digit* 1	*digit* 2	*digit* 3
NOD	EB**B**	B**A**N	**A**SH
HOG	NO**D**	B**A**R	**B**AN
SHY	TE**D**	P**A**R	**B**AR
BAN	HO**G**	EBB	**E**BB
BAR	FI**G**	T**E**D	**F**IG
JET	AS**H**	J**E**T	**H**OG
EBB	BA**N**	P**E**T	**J**ET
PAR	BA**R**	S**H**Y	**N**OD
ASH	PA**R**	F**I**G	**P**AR
PET	JE**T**	N**O**D	**P**ET
TED	PE**T**	H**O**G	**R**OT
ROT	RO**T**	R**O**T	**S**HY
FIG	SH**Y**	A**S**H	**T**ED

Problem 2

Show how to sort n integers in the range 0 to $n^2 - 1$ in $\mathcal{O}(n)$ time.

Solution. Psuedocode for sorting n integers in $\mathcal{O}(n)$, X is an array storing all n integers:

SORT-N-LINEAR(X, n)

```
1:   if n == 1
2:       return
3:   for i = 1 to n
4:       // change X[i] into base n
5:       TO-BASE-N(X[i])
6:   // each integer will have at most log_n n^2 = 2 digits
7:   // radix sort uses counting sort to sort n at most 2 digits integers
8:   RADIX-SORT(X)
9:   for i = 1 to n
10:      // convert X[i] back to its original value because we sorted them with base n
11:      CONVERT-BASE-N(X[i])
```

TO-BASE-N converts an integer into a base n number. E.g. 100 integers have the largest number of 9999, which can be converted to base 100, $9999 = 99 * 100^1 + 99 * 100^0$. Originally, radix sort will sort 4 digits for

9999. After converting its base to n, radix sort will need to sort **2 digits** 99_{100} and 99_{100}. Another example that the integer 400 will be converted to 4_{100}. The **digit** here is not limit from 0 to 9. ∎

Problem 3

Using Figure 8.4 as a model, illustrate the operation of BUCKET-SORT on the array $A=<.49, .30, .56, .74, .68, .50, .69, .37, .41, .72>$

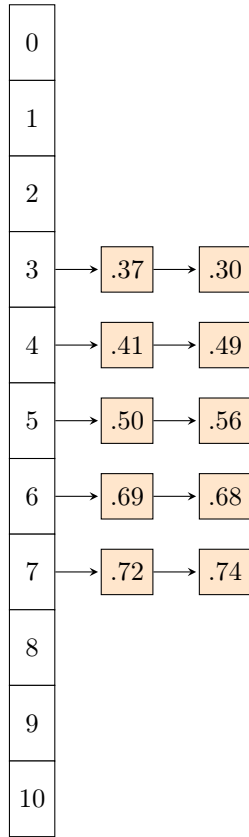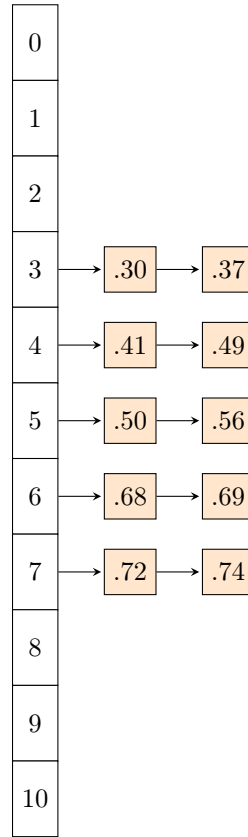Figure 1: Insert data

Figure 2: Sort each bucket

Solution. To get sorted data, chain all the nodes in Figure 2 from bucket 0 to bucket 10:
$< .30, .37, .41, .49, .50, .56, .68, .69, .72, .74 >$ ∎

Problem 4

Suppose that you have a "black-box" worst-case linear time median subroutine. Give a simple, linear-time algorithm that solves the selection problem for an arbitrary order statistic.

Solution. Psuedocode for finding the k-th smallest element in array $X[1..n]$:

FIND-K-TH-SMALLEST(X, n, k)

```
1:   // returns the index of the median for array X in linear time
2:   b = BLACK-BOX(X)
3:   if k == n / 2 // k-th smallest is median
4:      return X[b]
5:   // splits array X into two arrays, Y contains elements less than b and Z contains elements greater than b
6:   Y, Z = SPLIT(X, b)
7:   if k < n/2
8:      return FIND-K-TH-SMALLEST(Y, n/2, k)
9:   else
10:     // Find n/2 − k-th smallest for Z since it's half the size of X
11:     return FIND-K-TH-SMALLEST(Z, n/2, n/2 − k)
```

$T(n) = T(n/2) + \mathcal{O}(n)$
Apply master method $n^{\log_b a} = n^{\log_2 1} = n^0 = 1 < \mathcal{O}(n)$, may apply case 3.
First, we find $\epsilon = 1$ where $f(n) = \Omega(n^{(\log_2 1)+\epsilon})$.
Second, we find $c = 1/2$ where $af(n/b) = n/2 \leq 1/2n = cf(n)$ is true.
Thus, we can conclude that the running time of this algorithm is $\Theta(f(n)) = \Theta(n)$. ∎

Problem 5

Let $X[1..n]$ and $Y[1..n]$ be two arrays, each containing n numbers already in sorted order. Give an $\mathcal{O}(\lg n)$-time algorithm to find the median of all $2n$ elements in arrays X and Y.

Solution. Psuedocode for finding the median of arrays $X[1..n]$ and $Y[1..n]$, xi is X array's starting index and xj is X array's last index; yi is Y array's starting index and yj is Y array's last index:

FIND-MEDIAN$(X, Y, xi, xj, yi, yj, n)$

1:    **if** $n == 1$
2:      // Both arrays have only one element, pick the smaller one as median
3:        **return** MIN$(X[1], Y[1])$
4:    $xmid = X[(xi + xj)/2]$ // X's median
5:    $ymid = Y[(yi + yj)/2]$ // Y's median
6:    **if** $xmid < ymid$
7:      // median must be greater than $xmid$ and less than $ymid$
8:        **return** FIND-MEDIAN$(X, Y, xmid + 1, xj, yi, ymid, n/2)$
9:    **else**
10:      // median must be less than $xmid$ and greater than $ymid$
11:        **return** FIND-MEDIAN$(X, Y, xi, xmid, ymid + 1, yj, n/2)$

$T(n) = T(n/2) + \mathcal{O}(1)$
Apply master method case 2 because $n^{\log_b a} = n^{\log_2 1} = n^0 = 1 = f(n)$.
$T(n) = \Theta(n^{\log_2 1} \lg n) = \Theta(\lg n)$. ∎

Problem 6

Demonstrate the insertion of the keys 23, 38, 19, 40, 29, 43, 12, 27, 11 into a hash tale with collisions resolved by chaining. Let the table have 9 slots, and let the hash function be $h(k) = k \pmod 9$.

Solution. ∎

(1) insert $23 \to 23 \pmod 9 = 5$

(2) insert $38 \to 38 \pmod 9 = 2$

(3) insert $19 \to 19 \pmod 9 = 1$

(4) insert $40 \to 40 \pmod 9 = 4$

(5) insert $29 \to 29 \pmod 9 = 2, \text{insert head}$

(6) insert $43 \to 43 \pmod 9 = 7$

(7) insert $12 \to 12 \pmod 9 = 3$

(8) insert $27 \to 27 \pmod 9 = 0$

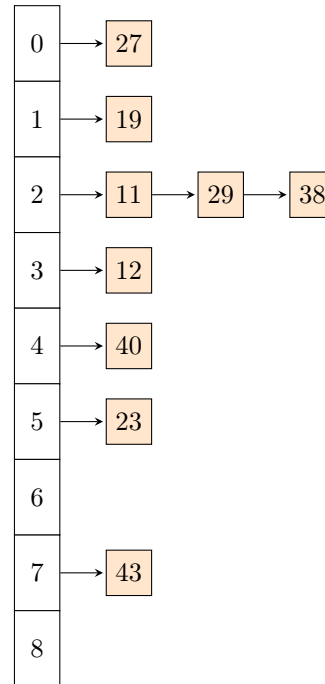(9) insert $11 \to 11 \pmod 9 = 2, \text{insert head}$

Figure 3: Hash table with chaining

Problem 7

Consider inserting the keys 10, 23, 31, 4, 12, 28, 17, 87, 58 into a hash table of length $m = 11$ using open addressing with the auxiliary hash function $h'(k) = k \pmod{m}$. Illustrate the result of inserting these keys using **linear probing**, using **quadratic probing** with $c1 = 1$, and $c2 = 3$, and using **double hashing** with $h_2(k) = 1 + (k \pmod{(m-1)})$.

(1) insert $10 \to 10 \pmod{11} = 10$

(2) insert $23 \to 23 \pmod{11} = 1$

(3) insert $31 \to 31 \pmod{11} = 9$

(4) insert $4 \to 4 \pmod{11} = 4$

(5) insert $12 \to 12 \pmod{11} = 1 \ \times$ collision

insert 12 to slot 2 because slot 1 is occupied

(6) insert $28 \to 28 \pmod{11} = 6$

(7) insert $17 \to 17 \pmod{11} = 6 \ \times$

insert 17 to index 7 because slot 6 is occupied

(8) insert $87 \to 87 \pmod{11} = 10 \ \times$

insert 87 to index 0 because slot 10 is occupied

(9) insert $58 \to 58 \pmod{11} = 3$

0	87
1	23
2	12
3	58
4	4
5	x
6	28
7	17
8	x
9	31
10	10

Figure 4: linear probing

resolve collision: $h(k) + 1 * i + 3 * i^2 \pmod{11}$

(1) insert $10 \to 10 \pmod{11} = 10$

(2) insert $23 \to 23 \pmod{11} = 1$

(3) insert $31 \to 31 \pmod{11} = 9$

(4) insert $4 \to 4 \pmod{11} = 4$

(5) insert $12 \to 12 \pmod{11} = 1 \ \times$ collision

$(1 + 1 + 3 * 1 * 1) \pmod{11} = (1 + 1 + 3) \pmod{11} = 5$

insert 12 to slot 5

$-------------------$

(6) insert $28 \to 28 \pmod{11} = 6$

(7) insert $17 \to 17 \pmod{11} = 6 \ \times$

$(6 + 1 + 3 * 1 * 1) \pmod{11} = (6 + 1 + 3) \pmod{11} = 10 \ \times$

$(6 + 2 + 3 * 2 * 2) \pmod{11} = (6 + 2 + 12) \pmod{11} = 9 \ \times$

$(6 + 3 + 3 * 3 * 3) \pmod{11} = (6 + 3 + 27) \pmod{11} = 3$

insert 17 to slot 3

$-------------------$

(8) insert $87 \to 87 \pmod{11} = 10 \ \times$

$(10 + 1 + 3 * 1 * 1) \mod 11 = (-1 + 1 + 3) \mod 11 = 3 \ \times$

$(10 + 2 + 3 * 2 * 2) \mod 11 = (-1 + 2 + 12) \mod 11 = 2$

insert 87 to slot 2

$-------------------$

(9) insert $58 \to 58 \pmod{11} = 3 \ \times$

$(3 + 1 + 3 * 1 * 1) \mod 11 = (3 + 1 + 3) \mod 11 = 7$

insert 58 to slot 7

0	x
1	23
2	87
3	17
4	4
5	12
6	28
7	58
8	x
9	31
10	10

Figure 5: quadratic probing

resolve collision: $h_2(k) = 1 + k \pmod{10}$

$new_k = (h_1(k) + i * h_2(k)) \pmod{11}$

(1) insert $10 \to 10 \pmod{11} = 10$

(2) insert $23 \to 23 \pmod{11} = 1$

(3) insert $31 \to 31 \pmod{11} = 9$

(4) insert $4 \to 4 \pmod{11} = 4$

(5) insert $12 \to 12 \pmod{11} = 1 \ \times$ collision

$h_2(12) = 1 + 12 \pmod{10} = 3$

$1 + 1 * 3 \pmod{11} = 4 \ \times$

$1 + 2 * 3 \pmod{11} = 7$

insert 12 to slot 7

$-\,-$

(6) insert $28 \to 28 \pmod{11} = 6$

(7) insert $17 \to 17 \pmod{11} = 6 \ \times$

$h_2(17) = 1 + 17 \pmod{10} = 8$

$8 + 1 * 8 \pmod{11} = 5$

insert 17 to slot 5

$-\,-$

(8) insert $87 \to 87 \pmod{11} = 10 \ \times$

$h_2(87) = 1 + 87 \pmod{10} = 8$

$8 + 1 * 8 \pmod{11} = 5 \ \times$

$8 + 2 * 8 \pmod{11} = 2$

insert 87 to slot 2

$-\,-$

(9) insert $58 \to 58 \pmod{11} = 3$

0	x
1	23
2	87
3	58
4	4
5	17
6	28
7	12
8	x
9	31
10	10

Figure 6: double hashing

Problem 8

Solve the following assembly-line problem:

$e_1 = 1, e_2 = 1, x_1 = 7, x_2 = 3$

$a_{1,1} = 3, a_{1,2} = 2, a_{1,3} = 5, a_{1,4} = 4, a_{1,5} = 2$

$a_{2,1} = 4, a_{2,2} = 2, a_{2,3} = 4, a_{2,4} = 6, a_{2,5} = 4$

$t_{1,1} = 1\, t_{1,2} = 2, t_{1,3} = 3, t_{1,4} = 1$

$t_{2,1} = 2, t_{2,2} = 3, t_{2,3} = 1, t_{2,4} = 2.$

	1	2	3	4	5	
$f_1[j]$	4	6	11	15	17	24
$f_2[j]$	5	7	11	17	20	23

Figure 7: f table

	2	3	4	5
$l_1[j]$	1	1	1	1
$l_2[j]$	2	2	2	1

Figure 8: l table

Solution. $f^* = 23, l^* = 2$. The best solution for this assembly line is to start from $a_{1,1}$, and goes through $a_{1,2}, a_{1,3}, a_{1,4}, t_{1,4}, a_{2,5}, x_2$. ∎

Problem 9

Find an optimal parenthesization of a matrix-chain product whose sequence of dimension is $<5, 7, 3, 2, 4, 5>$.
$A_1 = A_{5\times7}, A_2 = A_{7\times3}, A_3 = A_{3\times2}, A_4 = A_{2\times4}, A_5 = A_{4\times5}$
$p_0 = e, p_1 = 7, p_2 = 3, p_3 = 2, p_4 = 4, p_5 = 5$

j\i	1	2	3	4	5
5	202	152	70	40	0
4	152	98	24	0	
3	112	42	0		
2	105	0			
1	0				

Figure 9: m table

j\i	1	2	3	4
5	3	4	3	4
4	3	3	3	
3	1	2		
2	1			

Figure 10: s table

Solution. Positions of parenthesis: $((A_1(A_2 A_3))(A_4 A_5))$ *or* $(((5 \times 7)((7 \times 3)(3 \times 2)))((2 \times 4)(4 \times 5)))$ ∎

Problem 10

Determine an LCS of $<1, 1, 0, 1, 0, 1>$ and $<0, 0, 1, 1, 0, 1, 1>$.

Figure 11: c table for LCS

Solution. Thus, the longest common sequence is $< 1, 1, 0, 1, 1>$. ∎