CSCI-UA.0310-004/005 Basic Algorithms April 22, 2015

Solutions to Problem 1 of Homework 10 (9 (+6) points)

Name: Jason Yao *Due: Wednesday, April 22*

Let $B = 64$, and the actual block alphabet $\Sigma = \{0, \dots 9, a, \dots, z, A, \dots, Z,$ "." (period), " " (space)$\}$. To translate to more usual $\{0, \dots, 63\} = [64]$, let us encode the digits $0 \dots 9$ as $0 \dots 9$, letters $a \dots z$ as $10 \dots 35$, capital letters $A \dots Z$ as $36 \dots 61$, period "." as 62 and space " " as 63. As usual, EOF is encoded as 64.

(a) (3 points) With the convention above, characters in which position of the input string does the 3rd character of the SOLE encoding depend on? Assume that the input is a word "SoLe.". Find the 3rd character of the SOLE encoding of this string. Explain how exactly you determined that character.

(**Hint**: One option is to do the full encoding. However, if you are smart, you do not need to!)

Solution:

The 3rd character of the SOLE encoding depends upon:

The previous and next block of the SOLE encoding, so the 3rd character depends upon the 1st, 2nd and 4th characters

3rd character of the SOLE encoding = 60 = Y

Explanation:

By going through the encoding for the just the first 4 blocks, we gain the ability to determine the 3rd character. □

(b) (3 points) Assume somebody sent you the SOLE encoding which reads "ON7DMa1". What is the 3rd character of the original message? Justify your answer.

(**Hint**: One option is to do the full decoding. However, if you are smart, you do not need to!)

Solution:

The 3rd character of the original message is = 62 = .

In a similar vein to the above question, the decoding depends upon having the 2nd, 3rd and 4th block to find the 3rd block's original message.

This is because in the recovering the 2nd pass, we require the 2nd and 3rd block, and the 4th and 5th block.

After that, we regroup, and recover the 1st pass, and recovering the 3rd block requires having the 3rd and 4th block.

Thus, we require blocks 2, 3, 4, 5 to decode for character 3, and we find the original 3rd block message by going through the normal decode utilizing these blocks

The previous and next block of the SOLE encoding, so the 3rd character depends upon the 1st, 2nd and 4th characters

□

(c) (3 points) Change the encoded string in part (b) in a minimum number of places, such that your answer in part (b) changes to EOF.

(**Hint**: One option is to do the full decoding, change letter, and then do full re-encoding. However, if you are smart, you do not need to!)

Solution:

Change block 2 to 57 in the encoded message.

Change block 3 to 9 in the encoded message.

Answer was found by going backwards through the decoding process, and backtracing through the algorithm.

□

(d) (**Extra credit**: 6 points)

Get full encoding in part (a) and full decodings in parts (b) and (c).[1]

Solution:

Block Number	1	2	3	4	5	6	7
Input Alphabet	[64]	[64]	[64]	[64]	[64]	{EOF}	{0}
Actual Input	54∈[64]	24∈[64]	47∈[64]	14∈[64]	62∈[64]	EOF	n/a
With EOF	54∈[65]	24∈[65]	47∈[65]	14∈[65]	62∈[65]	64∈[65]	0∈[52]
Pass 1	14∈[64]	25∈[68]	57∈[60]	15∈[72]	22∈[56]	75∈[76]	0∈[52]
Regroup	14∈[64]	25∈[68]	57∈[60]	15∈[72]	22∈[56]	75∈[76]	0∈[52]
Pass 2	14∈[64]	61∈[64]	60∈[64]	63∈[64]	24∈[64]	11∈[64]	1∈[64]

Encoding of a: {14, 61, 60, 63, 24, 11, 1} = {e, Z, Y, " ", o, b, 1}

Pass 1:

1st double block: encode (54, 24)∈ [65]x[65] as 2 numbers in range [64]x[68]

$z = 24 * 65 + 54 = 1,614$

$z \ \% \ 64 = 25 * 64 + 14$

1st double block = (14, 25) ∈ [64]x[68]

2nd double block: encode (47, 14)∈ [65]x[65] as 2 numbers in range [60]x[72]

$z = 14 * 65 + 47 = 957$

$z \ \% \ 60 = 15 * 60 + 57$

2nd double block = (57, 15) ∈ [60]x[72]

3rd double block: encode (62, 64)∈ [65]x[65] as 2 numbers in range [56]x[76]

$z = 64 * 65 + 62 = 4,222$

$z \ \% \ 56 = 75 * 56 + 22$

[1] Compare the complexity with correct solutions to parts (a)-(c).

3rd double block = (22, 75) $\in$ [56]x[76]

Pass 2:

1st double block: encode (25, 57)$\in$ [68]x[60] as 2 numbers in range [64]x[64]

z = $57 * 68 + 25 = 3,901$

z % $64 = 60 * 64 + 61$

1st double block = (61, 60) $\in$ [64]x[64]

2nd double block: encode (15, 22)$\in$ [72]x[56] as 2 numbers in range [64]x[64]

z = $22 * 72 + 15 = 1,599$

z % $64 = 24 * 64 + 63$

2nd double block = (63, 24) $\in$ [64]x[64]

3rd double block: encode (75, 0)$\in$ [76]x[52] as 2 numbers in range [64]x[64]

z = $0 * 76 + 75 = 75$

z % $64 = 1 * 64 + 11$

3rd double block = (11, 1) $\in$ [64]x[64]

Encoding of b: {29, 21, 62, 13, 27} = {t, l, ".", d, r}

Recover Pass 2:

1st double block: encode (49, 7) $\in$ [64]x[64] as 2 numbers in range [68]x[60]

z = $7 * 64 + 49 = 497$

z % $68 = 7 * 68 + 21$

1st double block = (21, 7) $\in$ [68]x[60]

2nd double block: encode (39, 48) $\in$ [64]x[64] as 2 numbers in range [72]x[56]

z = $48 * 64 + 39 = 3,111$

z % $72 = 43 * 72 + 15$

2nd double block = (15, 43) $\in$ [72]x[56]

3rd double block: encode (10, 1) $\in$ [64]x[64] as 2 numbers in range [76]x[52]

z = $1 * 64 + 10 = 74$

z % $76 = 0 * 76 + 74$

3rd double block = (74, 0) $\in$ [76]x[52]

Recover Pass 1:

1st double block: encode (50, 21) $\in$ [64]x[68] as 2 numbers in range [65]x[65]

z = $21 * 64 + 50 = 1,394$

z % $65 = 21 * 65 + 29$

1st double block = (29, 21) $\in$ [65]x[65]

2nd double block: encode (7, 15) $\in$ [60]x[72] as 2 numbers in range [65]x[65]

z = $15 * 60 + 7 = 907$

z % 65 = $13 * 65 + 62$

2nd double block = (62, 13) $\in$ [65]x[65]

3rd double block: encode (43, 74) $\in$ [56]x[76] as 2 numbers in range [65]x[65]

z = $74 * 56 + 43 = 4,187$

z % 65 = $64 * 65 + 27$

3rd double block = (27, 64) $\in$ [65]x[65]

Encoding of c: {29, 21, 64} = {t, l}

Recover Pass 2:

1st double block: encode (57, 9) $\in$ [64]x[64] as 2 numbers in range [68]x[60]

z = $9 * 64 + 57 = 633$

z % 68 = $9 * 68 + 21$

1st double block = (21, 9) $\in$ [68]x[60]

Recover Pass 1:

1st double block: encode (50, 21) $\in$ [64]x[68] as 2 numbers in range [65]x[65]

z = $21 * 64 + 50 = 1,394$

z % 65 = $21 * 65 + 29$

1st double block = (29, 21) $\in$ [65]x[65]

2nd double block: encode (9, 15) $\in$ [60]x[72] as 2 numbers in range [65]x[65]

z = $15 * 60 + 9 = 909$

z % 65 = $13 * 65 + 64$

2nd double block = (64, 13) $\in$ [65]x[65]

□

CSCI-UA.0310-004/005 Basic Algorithms April 22, 2015

Solutions to Problem 2 of Homework 10 (10 points)

Name: Jason Yao *Due: Wednesday, April 22*

You own a zoo with n animals. Some pairs of animals cannot co-exist with each other (e.g., one will eat the other), while others can. You worked hard, and developed a complete set of m pairs of animals which cannot co-exist. Design $O(m + n)$ algorithm to partition (if at all possible) the animals into two groups, such that each animal in a group can co-exist with every other animal in the same group.
(**Hint**: Use BFS on an appropriate graph and use the BFS tree to derive the only *possible* partition. Then verify that this partition is indeed OK.)

Solution:

Input: Graph with animals as vertex, edges are between each vertex that **cannot** coexist.

Algorithm: Run BFS on any vertex, label root vertex as a part of group A.

During the BFS run, first consult the original graph to see if its edges between itself and another of the same group (i.e. A with an edge to A). If this is true, then it is not possible to divide the animals into just 2 groups, so break out of the algorithm.

Continuing in the BFS run, label the current vertex as a part of the opposite group as your predecessor (so if you came from a vertex in group A, label current vertex as in group B).

At the end of the BFS run, it will either be an unsolvable graph, or all the animals (vertices) in the graph are labeled as either in group A or B.

As the algorithm is basically a BFS run, our running time analysis is going to be:

$T(n) = \Theta(m + n)$ □

CSCI-UA.0310-004/005 Basic Algorithms April 22, 2015

Solutions to Problem 3 of Homework 10 (15 points)

Name: Jason Yao *Due: Wednesday, April 22*

A fellow Moe and his buddy Joe live in a city $G = (V, E)$ which is an undirected graph on n verteces and m edges, given in the adjacency list form. Moe lives in a vertex a and owns a crazy dog Mimi, while Joe lives at a vertex b and owns a crazy dog Kiki. This Sunday Moe wants to take Mimi to a veterinarian clinic located at vertex c, while Joe wants to take Kiki to the dance competition located at a vertex d. One problem, though: the dogs hate each other, and if one of them smells the other, all hell breaks lose. Luckily, they can smell each other only if within distance at most 15 in G, and you know that $dist_G(a, b), dist_G(c, d) > 15$. Moe and Joe would like to start at the same time (with Moe and Mimi at a and Joe and Kiki at b) and get both dogs to their respective destinations c and d in the smallest number of *steps* t. A *step* consists of both dogs going to their respective neighboring verteces, or one dog going to a neighboring vertex, and the other dog staying put, barking at the pedestrians. Of course, such a step is possible only if the dogs stay within distance 16 or more both before and after the step.

Your job is to design an algorithm for Moe and Joe to compute t (and the optimal route), if the route exists, and analyze its running time.

(a) Using one or more runs of the BFS algorithm on G, fill in the matrix $OK[x, y]$, where $OK[x, y] = 1$, if it is OK for Mimi to be at vertex x and Kiki to be at vertex y at the same time, and $OK[x, y] = 0$ otherwise. How long did it take you to fill this matrix $OK[x, y]$?

Solution:

As matrix OK[x,y] is an (n x n) matrix, and we are going through the graph via BFS, which completes each row.

T(n) = rows * (BFS time)

T(n) = n * (m + n)

T(n) = n(m + n) □

(b) Design a graph $H = (V', E')$ whose vertex set consists of possible "ok configurations" for Mimi and Kiki, and whose edge set represents the possible single steps of your algorithm. Be sure to formally define V' and E' as functions of V and E and the matrix OK from part (a). How long (in the worst case) did it take you to create an adjacency list for H (not counting what you did in part (a))? What is the maximum $|V'|$ and $|E'|$?

Solution:

H = (V', E')

V' = $\{V_iV_j \mid \text{OK[i,j]} == 1\}$

E' = $\{ (v_i, v_j) \mid v_i = v_{xy}, v_j = v_{yz}$ & $OK[x, y] == 1$ & $OK[y, z] == 1\}$

Time to make adjacency list:

T(n) = 1 pass of BFS, so

$T(n) = \Theta(m + n)$

Maximum $|V'| = |V^2| - |V|$

Maximum $|E'| = \Theta(n^4)$ □

(c) Describe the original problem as a shortest path computation on H. Finally, solve the original problem, and help Moe and Joe. Analyze the overall running time of your algorithm, as a function of n and m.

Solution:

Run BFS on the start node:

BFS(v_{ab}), and whatever is the shortest distance found at the end node v_{cd} will be the shortest valid path for both Moe and Joe to reach their destinations.

T(n) = BFS

T(n) = $\Theta(m + n)$ □