CSCI-UA.0310-004/005 Basic Algorithms April 1, 2015

Solutions to Problem 1 of Homework 7 (10 points)

Name: Jason Yao *Due: Wednesday, April 1*

Assume that you attend a boring science convention that takes place in a conference center that is a m by n rectangle with a mini-exhibition at each corner. To attract customers, each exhibition offers a certain number of free pizza slices to passing customers. You want to move from north-west corner to the south-east corner of the conference center. Since you do not want to spend too much time there, in each step you may either move south or east. Your goal is to maximize the total number of free pizza slices you can get.

More formally, if you pass by the corner exhibition located at (i, j) where $0 \leq i \leq m$ and $0 \leq j \leq n$, you get $p(i, j)$ slices of pizza. You are told all of the $p(i, j)$ a priori. Your goal is to design a path from $(0, 0)$ to (m, n) allowing you to consume as much of pizza as you can!

Give an efficient (i.e., polynomial in m and n) dynamic programming algorithm for this problem, and analyze its running time.

Solution:

```
int B[][];
char reversedMap[];

int, char[] getPizzaDriver(m, n)
  B[][] = [m][n]
  B[0][0] = p(0, 0)
  maxPizza = getPizza(0, 1)

  for i = 1 to n/2
    swap (reversedMap[i], reversedMap[i + n/2])
  RETURN (maxPizza, reversedMap)
END getPizzaDriver

int getPizza(i,j)
  if (i == 0)
    B[i][j] = B[i][j - 1] + p(i, j)
  elseif (j == 0)
    B[i][j] = B[i - 1][j] + p(i, j)
  else
    if (B[i][j - 1] < B[i - 1][j])
      B[i][j] = B[i - 1][j] + p(i, j)
      map += 'S'
    else
      B[i][j] = B[i][j - 1] + p(i,j)
      map += 'E'
  if ((i == m) && (j == n))
```

```
    RETURN B[m][n]
  elseif (j ≤ n)
    RETURN getPizza(i, j + 1)
  else
    RETURN getPizza(i + 1, 0)
END getPizza
```

□

CSCI-UA.0310-004/005 Basic Algorithms April 1, 2015

Solutions to Problem 2 of Homework 7 (12 points)

Name: Jason Yao *Due: Wednesday, April 1*

You have an $n \times n$ square field. Unfortunately, in some of the squares there is a beautiful little pony. You have two friends Jerry and Elaine who do not like ponies, so you want to find some (possibly) smaller square included in the original field that is completely ponyless.

More formally, you are given an $n \times n$ matrix A of bits, where A$[i][j] = 1$ iff there is a pony in a square at position (i, j). Find an efficient dynamic programming algorithm for finding the biggest possible ponyless sub-square in your field.

Define an auxiliary $n \times n$ matrix B such that $B[i][j]$ memoizes the length (which is the same as the width) of the biggest possible ponyless square with the bottom-right corner at position (i, j).

(a) (4 points) Show a recursive relationship between $B[i][j]$ and $B[i-1][j-1]$. Based on this, find $O(n^3)$ dynamic programming algorithm for computing the matrix B.

```
badPonyCheck(A)
maxLength = 0
n = A.length
B[][] = [n][n]
for i = 1 to n
  for j = 1 to n
    if (A[i][j] == 1)
      B[i][j] = 1
    else
      B[i][j] = B[i - 1][j - 1] + 1
      for k = 1 to (B[i][j] - 1)
        if (B[k][j] == 1)
          B[i][j] = 1
        if (B[i][k] == 1)
          B[i][j] = 1

        if (maxLength < B[i][j])
          maxLength = B[i][j]
      endfor
    endelse
  endfor
endfor
END badPonyCheck
```

(b) (7 points) Show a recursive relationship between $B[i][j]$ and $(B[i-1][j], B[i][j-1])$. Based on this, find a faster $O(n^2)$ dynamic programming for computing the matrix B.

Solution:

```
int betterPonyCheck(A)
maxLength = 0
n = A.length
B[][] = [n][n]
for i = 1 to n
  for j = 1 to n
    if (A[i][j] == 1)
      B[i][j] = 1
    else if ((i == 1) && (j == 1))
      B[i][j] = 1
    elseif (i == 1)
      B[i][j] = B[i][j - 1] + 1
    elseif (j == 1)
      B[i][j] = B[i - 1][j] + 1
    elseif (B[i - 1][j] == B[i][j - 1])
      B[i][j] = B[i - 1][j - 1] + 1
    endelseif

    if (maxLength < B[i][j])
      maxLength = B[i][j]
    endif
  endfor
endfor
RETURN maxLength
END betterPonyCheck
```

□

(c) (1 point) Explain how to give the final solution to the original problem, assuming you already computed B. What is the final time complexity for the best final solution?

Solution:

If we are already given B, then T(n) = O(1), since all we need to do is return the value that we kept incrementing during our algorithm, as that will have the largest value of ponyless squares that we have found so far. □

CSCI-UA.0310-004/005 Basic Algorithms April 1, 2015

Solutions to Problem 3 of Homework 7 (8 points)

Name: Jason Yao *Due: Wednesday, April 1*

Recall, Fibonacci number $F_0, F_1, F_2, ...$ are defined by setting $F_0 = F_1 = 1$, and $F_i = F_{i-1} + F_{i-2}$, for $i \geq 2$. Ignoring the issue of F_n being exponentially large, one can easily compute F_n in linear time:

$F_0 = F_1 = 1$
For $i = 2$ **to** n
    $F[i] := F[i-1] + F[i-2]$

One can also write a recursive procedure:

FIB(n):
    If $n = 0$ or $n = 1$ **Return** 1
    Else Return (FIB($n-1$) + FIB($n-2$))

(a) Let $T(n)$ be the running time of FIN(n). Clearly, $T(n) = T(n-1) + T(n-2) + O(1)$. Prove by induction that $T(n) \geq c^n$, for some constant $c > 1$. What is the largest value of c that you can use in your induction?

Solution:

T(n) = T(n - 1) + T(n - 2) + O(1)

$T(n) \geq c^n$

$T(n-1) + T(n-2) + O(1) \geq c^n$

$c^{n-1} + c^{n-2} \geq c^n$, since Professor Dodis said it was alright to drop the O(1) term

$c^n(\frac{1}{c} + \frac{1}{c^2}) \geq c^n$

$\frac{1}{c} + \frac{1}{c^2} \geq 1$

$c + 1 \geq c^2$

$1 + c - c^2 \geq 0$

By the quadratic formula,

$c \geq \frac{1}{2}(1 \pm \sqrt{5})$, but since $c > 1$,

$c \geq \frac{1}{2}(1 + \sqrt{5})$ □

(b) Using memoization, write a variant of the recursive procedure FIB above, called SMART-FIB, which will compute F_n in time $O(n)$, like we expect a good procedure should.

Solution:

```
B[];

int smartFibDriver(n)
  if (n < 2)
    RETURN 1
  else
    B = [n]
    B[0] = B[1] = 1
    RETURN smartFib(n, 2)
END smartFibDriver

int smartFib(n, counter)
  if (counter == n)
    RETURN B[n - 1]
  else
    B[counter] = B[counter - 1] + B[counter - 2]
    RETURN smartFib(n, counter + 1)
END smartFib
```

□

CSCI-UA.0310-004/005 Basic Algorithms April 1, 2015

Solutions to Problem 4 of Homework 7 (6 Points)

Name: Jason Yao *Due: Wednesday, April 1*

Find the Longest Common Subsequence of the following two strings: $X = BARRACUDA$, $Y = ABRACADABRA$. (Notice, you need to find the actual LCS, not only its length.)

Solution:

□