CSCI 570 Spring 2015 Discussion 7

1. We have a list A of n integers, for some $n = 2^k - 1$, each written in binary. Every number in the range 0 to n is in the list exactly once, except for one. However, we cannot directly access the value of integer A[i] (for any i); instead, we can only access the j^{th} bit of i: $A[i][j]$. Our goal is to find the missing number.

Give an algorithm to find the missing integer that uses O(n) bit accesses.

2. Graduate students get a lot of free food at various events. Suppose you have a schedule of the next n days marked with those days when you get a free dinner, and those days on which you must acquire dinner on your own. On any given day you can buy dinner at the cafeteria for $3. Alternatively, you can purchase one week's groceries for $10, which will provide dinner for each day that week (that day and the six that follow). However, because you don't have a fridge, the groceries will go bad after seven days (including the day of purchase) and any leftovers must be discarded. Due to your very busy schedule, these are your only two options for dinner each night. Your goal is to eat dinner every night while minimizing the money you spend on food.

3. You are in Downtown of a city and all the streets are one-way streets. You can only go east (right) on the east-west (left-right) streets, and you can only go south (down) on the north-south (up-down) streets.

a) In Figure A below, how many unique ways are there to go from the intersection marked S (coordinate (0,0)) to the intersection marked E (coordinate (5,4))? (Hint: You may want to do part (b) first.)

b) Continuing from part (a), formulate the solution to this problem as a dynamic programming problem. Please make sure that you include all the boundary conditions and clearly define your notations you use. (Hint: this is not an optimization problem.)

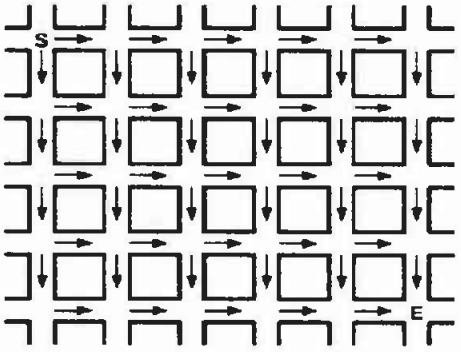


Figure A.

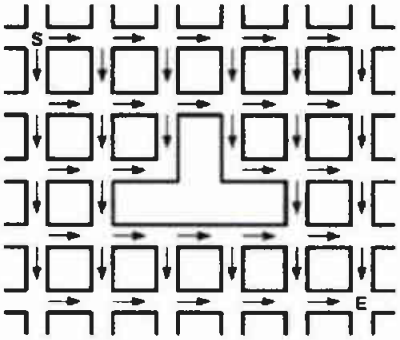


Figure B.

c) Repeat this process with Figure B; be wary of dead ends.

c) Continuing from part (a), in Figure B above, how many unique ways are there to go from the intersection marked S to the intersection marked E? You can solve the problem numerically and directly on the figure (Hint: beware of dead ends.)

1. Select any bit position and count the 0s and 1s in that position (O(n)). There will be n/2 - 1 of one of them, and n/2 of the other. Whichever bit has fewer is the missing bit in that position; mark that in the solution and discard any with the alternate bit (note that, in every other bit position, this discards an equal number of 1s and 0s). Repeat this until the missing integer is found.
   Recurrence: T(n) = T(n/2) + O(n) = O(n).

2. Define OPT(n) to be the cheapest way to eat for days 1..n. It has value OPT(n-1) if day n is free food; otherwise, we have to choose whether to go to the cafeteria that day ($3 + OPT(n-1)) or finish the last of the groceries ($10 + OPT(n-6)). When I give this as a homework question, I allow full credit if they say OPT(n-7) instead; the idea counts here. So, if day n isn't free food, OPT(n) = min(3 + OPT(n-1) , 10 + OPT(n-6))

3.

a) The number of unique ways are shown as follows:

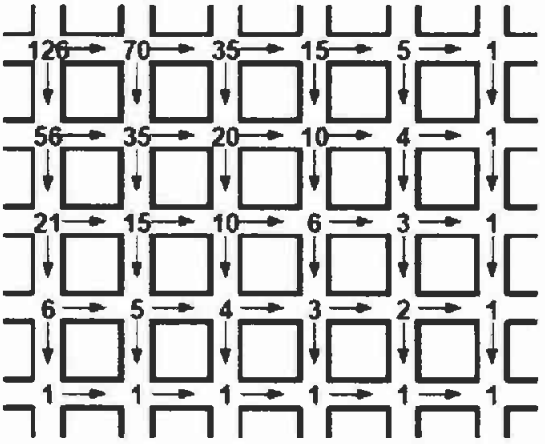


Figure A.

b) Define OPT(i,j) as the number of unique ways from intersection (i,j) to E: (5,4). The recursive relation is :

OPT(i,j) = OPT(i+1, j) + OPT(i,j+1) , for i<5, and j<4

Boundary condition: OPT(5,j) = 1; OPT(i,4) = 1

c) With dead ends, the numerical results of the number of unique ways are shown as follows:

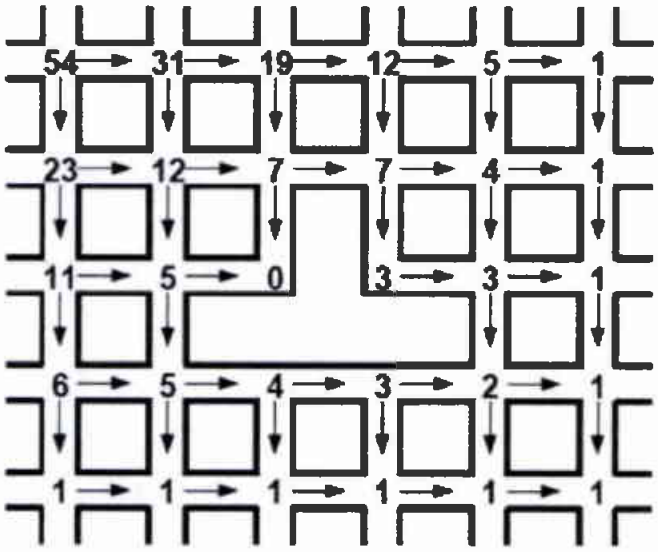
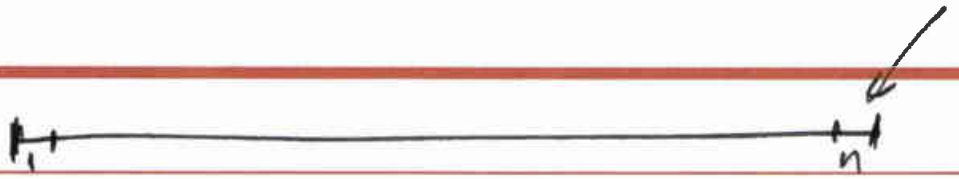


Figure B.

(1) (1) 1 1 . . . 1 1

$$\underbrace{n + n/2 + n/4 + \cdots + 1}_{2n} = O(n)$$

(2)



$OPT(n)$ = value of the opt. sol. for n days (days $1..n$)

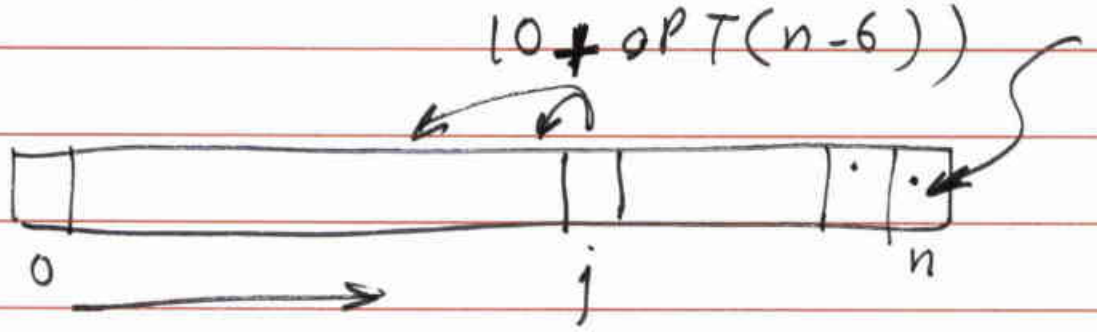
$$OPT(n) = \begin{cases} OPT(n-1) & \text{if there is free dinner on day } \underline{n} \\ \min(3 + OPT(n-1), \\ \quad 10 + OPT(n-6)) \end{cases}$$

FD is an array holding info. on free dinners

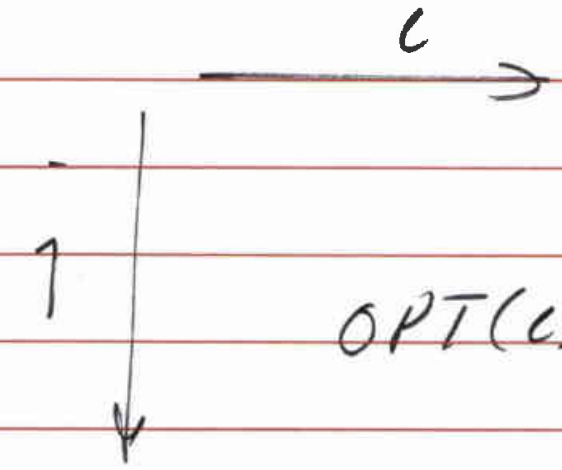
$FD(1..n)$

$$FD(i) = \begin{cases} 0 & \text{if no free dinner} \\ -3 & \text{" there is free dinner} \end{cases}$$

$$OPT(n) = \min(3 + OPT(n-1) + FD(n), \\ 10 + OPT(n-6))$$



(3)



$OPT(i,j) = OPT(i+1,j) +$

$OPT(i,j+1)$