COT 6401 The Analysis of Algorithms

`Midterm Test`

Open books and notes

Name ——————————————- SSN ——————————————-

1. (25%) Suppose that the function F is defined for all power of 2 and is described by the following recurrence equation and base case:

$$F(n) = n - 2 + 2F(n/2)$$

$$F(1) = 1$$

   (a) Find an *exact solution* for F when n is a power of 2.

   (b) What is the asymptotic order of $F(n)$?

   (c) Demonstrate the correctness of your result by showing $F(16)$.

Answer

   (a) Let $n = 2^k$, then $F(n) = \sum_{i=1}^{k} 2^i(2^{k-i} - 2) + 2^k = 2^k(k-1) + 2 = n(\lg n - 1) + 2$

   (b) $\Theta(n \lg n)$

   (c) $F(16) = 16(\lg 16 - 1) + 2 = 50$

2. (25%) Suppose you have an array of 1234 records in which only a few are out of order and they are not very far from their correct positions. Which sorting algorithm (among merge, insertion, heap, quick, bubble, counting, and radix) would you see to put the whole array in order? Justify your choice.

Answer

Insertion sort: suppose k elements are out of order and, among them, the ith ($i = 1, 2, ..., k$) element is d_i away from its correct position. The complexity, in terms of the number of exchanges, is $\sum_{i=1}^{k} d_i$. (Note that $k \ll n$ and $d_i \ll n$.)

(In the following questions, explain your solutions in **plain English** first and, then, **pseudo code**.)

3. (25%) Design an efficient algorithm (in terms of asymptotic complexity in the worst case) to determine if two students in a class of n students have the same height. What is the complexity of your algorithm? Consider the following two cases:

   (a) The smallest measure is centimeter.

   (b) There is no such a limit (i.e., each measure is precise).

Answer

Case 1: ($\Theta(n)$) Initialize "height slots" to unmark. Measure each student, mark the corresponding height slot, and return *true* when a marked slot is visited.

Case 2: ($\Theta(n \lg n)$) Apply any $\Theta(n \lg n)$ sorting algorithm, scan the sorted array (of heights) from left and right, and return *true* whenever two neighbors are of the same height.

Return *false* if there is no true return at the end for both cases.

Pseudo code

$H(i)$ is the height of student i. $M(j)$ is the mark for height j. *min_height* and *max_height* are pre-determined or can be calculated in linear time.

```
HEIGHT_CASE_ONE(H, A)
1   for j ← min_height to max_height
2       M[j] ← false
3   for i ← 1 to n
4       if M(H(i))
5          then return true
6          else  M(H(i)) ← true
7   return false
```

```
HEIGHT_CASE_TWO(H, A)
1   quicksort (H)
2   for i ← 1 to n − 1
3       if H(i) = H(i + 1)
4          then  return true
5   return false
```

4. (25%) Give an algorithm using dynamic programming to determine how many distinct ways there are to give a cents in change using any coins from among pennies, nickels, dimes, and quarters. For example, there are 6 ways to give 16 cents change: a dime, a nickel, and a penny; a dime and 6 pennies; 3 nickels and a penny; 2 nickels and 6 pennies; one nickel and 11 pennies; and 16 pennies. Demonstrate your solution by showing a step-by-step solution for 12 cents change.

Answer

$N(a,k)$ is the number of distinct ways to give a cents change using the first k denominations (with 1st being the smallest). Let v_k be the value of the kth denomination, i.e., $v_1 = 1$, $v_2 = 5$, $v_3 = 10$, and $v_4 = 25$.

$$\begin{aligned} N(a,k) &= 1 \quad (\textit{if } k = 1 \textit{ or } a = 0) \\ N(a,k) &= N(a,k-1) \quad (\textit{if } a < v_k) \\ N(a,k) &= N(a - v_k, k) + N(a,k-1) \quad (\textit{otherwise}) \end{aligned}$$

Pseudo code

```
CHANGE(a, k)
1   for i ← 0 to a
2       N(i, 1) ← 1
3   for i ← 0 to a
4       for j ← 1 to k
5           if i < v_j
6              then N(i, j) ← N(i, j − 1)
7              else  N(i, j) ← N(i − v_j, j) + N(i, j − 1)
```

For CHANGE$(12, 4)$, we have

row: a col: k	0	1	2	3	4	5	6	7	8	9	10	11	12
1 ($v_1 = 1$)	1	1	1	1	1	1	1	1	1	1	1	1	1
2 ($v_2 = 5$)	1	1	1	1	1	2	2	2	2	2	3	3	3
3 ($v_3 = 10$)	1	1	1	1	1	2	2	2	2	2	4	4	4
4 ($v_4 = 25$)	1	1	1	1	1	2	2	2	2	2	4	4	4

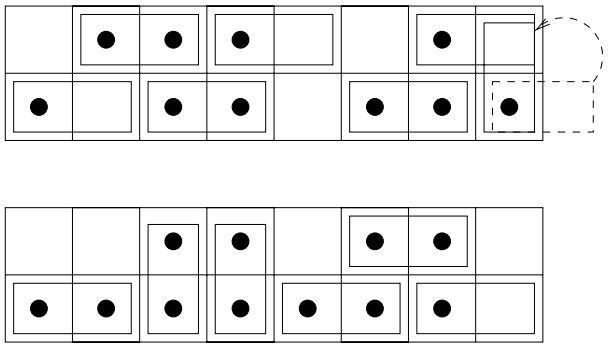
or

$$\begin{aligned} N(12,4) &= N(12,3) = N(2,3) + N(12,2) \\ &= N(2,2) + (N(7,2) + N(12,1)) \\ &= N(2,1) + ([N(2,1) + N(7,1)] + N(12,1)) \\ &= 1 + ([1+1]+1) = 4 \end{aligned}$$

Bonus Question

5. (5 points) Given a $2 \times n$ rectangle of 1×1 squares. Some squares have black bullets and others are blank. Use a minimum number of 1×2 rectangles to *cover* all 1×1 squares with black bullets.

   - Is there a greedy algorithm to solve this problem? If yes, provide such an algorithm.
   - Demonstrate the correctness of your solution by solving the following two samples.



Answer

Two rows are $P(i,1)$ and $P(i,2)$ where column i is labeled from left to right. An h-bar (horizontal bar) at $P(i,k)$ $(k = 1, 2)$ covers $P(i,k)$ and $P(i+1,k)$. A v-bar (vertical bar) at $P[i,*]$ covers $P(i,1)$ and $P(i,2)$.

Exam each column i from left to right. Apply v-bar if two rows (of column i) are both *uncovered blacks* and apply h-bar (at row k) if one row (row k) is an *uncovered black*. If an h-bar starts in the last column, change it to a v-bar.

Optimality is straightforward by induction on the number of columns.

Pseudo code

S (empty initially) is a set of selected 1×1 squares.

```
COVERAGE(P)
1   for i ← 1 to last_column −1
2       case (P(i,1), P(i,2)) of
3            (black, black): S ← S ∪ {(P(i,1), P(i,2))}
4            (black, white): S ← S ∪ {(P(i,1), P(i+1,1))}; P(i+1,1) ← white
5            (white, black): S ← S ∪ {(P(i,2), P(i+1,2))}; P(i+1,2) ← white
6   if P(last_column,1) = true or P(last_column,2) = true
7       then S ← S ∪ {(P(last_column,1), P(last_column,2)}
```