10/03/2016

ALGORITHMS

HOME - WORK - 3

Kiran Shettar
UML ID - 01605800

Problem 16.2-5 (page 428)

Describe an efficient algorithm that gives a set $\{x_1, x_2 \ldots x_n\}$ of points on the real line, determines the smallest set of unit-length closed intervals that contains all of the given points. Argue that your algorithm is correct.

Let us consider a set of points on line $S = \{x_1, x_2, x_3, x_4 \ldots x_n\}$

Now, we can consider the sorted points on the line.

Now we can write as

$x_1 < x_2 < x_3 < \ldots < x_n$.

Let $x_1, x_2, x_3 \ldots x_n$ be the sorted points on the line.

Here x_1 is the minimum

$x_1 = \min\{s\}$ & $x_n = \max\{s\}$

Two points, x_i & x_j are at unit distance if $x_j - x_i = 1$ when $i<j$

At each stage, it finds an interval.

Now, let us consider that the set

X has points $\{x_1, x_2 \ldots x_n\}$ & set Y is $\{\phi\}$.

Optimal solution for this is:

1. $Y \leftarrow Y \cup \{[\min(x), \min(x)+1]\}$
2. $X \leftarrow$ for x $\{x : x \leq \min(x)+1\}$
3. if $x = 0$ then return Y
4. else go to line 1.

Proof of correctness:

From the greedy choice, there is an optimal solution containing:

$[\min(x), \min(x)+1]$

Let the optimal solution have a point 'a' which starts before 'x_1' on the sorted order.

As 'x_1' is the smallest, we can replace 'a' with 'x_1'. This way the number of intervals are not increased. Hence we can say that this is still an optimal solution.

The final solution is the union of all the subproblems. And an optimal substructure is important in the greedy algorithm.

Problem 16-1 (a) Page 446

Consider the problem of making change for 'n' cents, using the fewest number of coins. Assume that each coins value is an integer.

(a) Describe a greedy algorithm to make a change consisting of quarters, dimes, nickels & pennies. Prove that your algorithm yeilds an optimal solution.

A greedy algorithm to make a change consisting of quarters.

Let us give the names as follows

quarters $= q$, dime $= d$, nickel $= n$, pennies $= p$.

Case 1: When we have $n=0$, then the optimal solution would be giving back no coins.

Case 2: When $n>0$, we have to see the value of the largest coin whose value is $\leq n$. Let us give the name 'x' for the largest coin value.

To prove that this algorithm yeilds the optimal solution, we should show that this holds a greedy choice property.

For this we can consider the following senarios.

→ scenario 1: $1 \leq n \leq 5$ then $x=1$

→ scenario 2: $5 \leq n \leq 10$ then $x=5$

→ scenario 3: $10 \leq n \leq 25$ then $x=10$

→ scenario 4: $25 \leq n$ then $x=25$.

Here, 'x' represents the maximum number of coins

For scenario 1, this can contain only pennies. For scenario 2, it can contain a ~~pe~~ nickel & others of pennies to make it only 6 coins together or we can give two nickels. But the maximum change that we can give is 'x' = 10. [each is a penny]

So this will be same for the seenario 3 & 4.

Hence, we can say that there's an optimal solution that includes the greedy choice. And we can combine the subproblem & then come to the original problem to produce an optimal substructure.

For our scenario 1, the running time will be O(1) as there are only pennies. And for other scenarios it'll be O(n).

Over all the running time for this algorithm will be O(n) for making change for 'n' cents using the fewest number of coins.