BLM5106- Advanced Algorithm Analysis and Design

H. İrem Türkmen

Introduction to algorithms TH Cormen, CE Leiserson, RL Rivest, C Stein

HASH FUNCTIONS

- Applications of Hash Functions and Desirable Properties

Dynamic Programming

- Dynamic programming, like the divide-and-conquer method, solves problems by combining the solutions to subproblems.
- A divide-and-conquer algorithm does more work than necessary, repeatedly solving the common subsubproblems.
- A dynamic-programming algorithm solves each subsubproblem just once and then saves its answer in a table, thereby avoiding the work of recomputing the answer every time it solves each subsubproblem.
- We typically apply dynamic programming to ***optimization problems***

Dynamic Programming

- When developing a dynamic-programming algorithm, we follow a sequence of four steps:

1. Define sub problems
2. Relate subproblem solutions
3. Recurse&memorize or Buid DP table bottom up
4. Solve originial problem

Dynamic Programming
Fibonacci Numbers ☺

- Lets see Dynamic Programming Solution

Dynamic Programming

- Longest Palindromic Sequence

Dynamic Programming

- Rod cutting

Rod Cutting: Recursive Solution

```
-- price array p, length n
Cut-Rod(p, n)
    if n = 0 then
        return 0
    end if
    q := MinInt
    for i in 1 .. n loop
        q := max(q, p(i) + Cut-Rod(p, n-i)
    end loop
    return q
```

Rod Cutting: Top Down Memorized Solution

```
MemoizedCutRod(p, n)
    r: array(0..n) := (0 => 0, others =>MinInt)
    return MemoizedCutRodAux(p, n, r)

MemoizedCutRodAux(p, n, r)

if r(n) = MinInt then     -- calculate a new solution?
        q: int := MinInt
        for i in 1 .. n loop
            q := max(q, p(i) + MemoizedCutRodAux(p, n-i, r))
        end loop
    end if
    r(n) := q
end if
return r(n)
```

Rod Cutting: Bottom Up Solution

```
BottomUpCutRod(p, n)
    r: array(0..n)    -- optimal value for rods of length 0..n
    r(0) := 0
    for j in 1 .. n loop
        q := MinInt
        for i in 1 .. j loop  -- Find the max cut position for length j
            q := max(q, p(i) + r(j-i)
        end loop
        r(j) := q
    end loop
    return r(n)
```

Rod Cutting: Finding the Solution
Memorized buttom up (non-recursive)

Let's use the bottom up approach and remember cuts

```
ExtendedBottomUpCutRod(p, n)
    r: array(0..n)    -- optimal value for rods of length 0..n
    s: array(0..n)    -- optimal first cut for rods of length 0..n
    r(0) := 0
    for j in 1 .. n loop
        q := MinInt
        for i in 1 .. j loop  -- Find the max cut position for length j
            if q < p(i) + r(j-i) then
                q := p(i) + r(j-i)
                s(j) := i   -- Remember the value of best so far value of i
            end if
        end loop
        r(j) := q
    end loop
    return r and s

PrintCutRodSolution(p, n)
    (r, s) := ExtendedBottomUpCutRod(p, n)
    while n > 0 loop
        print s(n)
        n := n - s(n)
    end loop
```