CS207 Design and Analysis of Algorithms

Sajith Gopalan

Indian Institute of Technology Guwahati

sajith@iitg.ac.in

January 26, 2022

Dynamic Programming

Dynamic Programming

- Particularly useful for optimization problems. Solution space. Constraints. Many feasible solutions, each of which satisfies the constraints, and has a value. We wish to find one feasible solution that minimises (maximizes) value.
- Steps:
  - Formulate the structure of an optimal solution
  - Recursively define the value of an optimal solution
  - Compute the value of an optimal solution, typically in a bottom-up fashion
  - Construct an optimal solution from computed information
- Works particularly when the following properties hold:
  - Optimal Substructure: an optimal solution to the problem contains within it optimal solutions to subproblems
  - Overlapping subproblems: recursive algorithm for the problem solves the same subproblems repeatedly

The 0-1 Knapsack Problem

- Ali Baba's brother Kassim goes into the cave of the 40 thieves.
- His donkey can carry a weight of at most W.
- He finds n metal ingots arranged in order.
- Ingot i is labelled by its (integer) weight w_i and its (integer) value v_i.
- Kassim's avarice knows no bounds.
- He will carry home the largest possible value.
- We need an algorithm for him to run.
- A trivial algorithm looks at each of the 2^n possible subsets: check if the subset has a total weight of at most W; declare it the "best-so-far" if the sum of its members' values exceeds those of the subsets seen earlier; the "best-so-far" in the end is the best. This takes $O(n.2^n)$ time, because a subset and its complement together cover all n items.

0-1 Knapsack Solution: Recursive formulation

- Look at item n.
- Two possibilities: take it, or leave it
- If we take it, our residual capacity decreases to $W - w_n$.
- If we leave it, our residual capacity remains W.
- Suppose we have solved
  1. the subproblem on $n-1$ items with maximum weight $W - w_n$, and know $A[n-1, W - w_n]$, the maximum value that can be carried off in that case, and
  2. the subproblem on $n-1$ items with maximum weight W, and know $A[n-1, W]$, the maximum value that can be carried off in that case.

0-1 Knapsack Solution: Recursive formulation

- If $A[n-1, W-w_n] + v_n \geq A[n-1, W]$, then the solution of (1) plus item n is our solution; set $A[n, W] = A[n-1, W-w_n] + v_n$
- Otherwise, the solution of (2) is our solution; set $A[n, W] = A[n-1, W] + v_n$

Recursive Binary Knapsack

w, v and B are global arrays; B is initialized to all 0

Algorithm 1 RBK(n, W)

```
if (n = 0 or W = 0) return 0;
a = RBK(n − 1, W − w_n) + v_n;
b = RBK(n − 1, W);
if a ≥ b then
    B[n, W] = 1; return a;
else
    return b;
end if
```

Binary Knapsack Print Solution

Algorithm 2 BKPS(n, W)

if $(n = 0 \text{ or } W = 0)$ **return**;
if $B[n, W]$ **then**
  BKPS$(n - 1, W - w_n)$; **Print** n;
else
  BKPS$(n - 1, W)$;
end if
return

Analysis

- BKPS runs in $O(n)$ time; each call spawns only one recursive call, and that too with a decremented n
- RBK runs in $\Omega(nW)$ time; usually much more, because there are overlapping subproblems that are repeatedly solved
- Not good!

Iterative Binary Knapsack

Algorithm 3 IBK(W, w, v, n)

Initialize 2-D array $A[0 \dots n, 0 \dots W]$ to all zero
for $i = 1$ to n **do**
    for $w = 1$ to W **do**
        $A[i, w] = A[i-1, w]$; $B[i, w] = 0$;
        if $w \geq w_i$ **and** $A[i-1, w] < A[i-1, w-w_i] + v_i$ **then**
            $A[i, w] = A[i-1, w-w_i] + v_i$; $B[i, w] = 1$;
        end if
    end for
end for

Analysis

- IBK runs in $O(nW)$ time
- The total length of the input is $\log W + \sum_{1=1}^{n} w_i + \sum_{1=1}^{n} v_i + \log n = \Omega(n + \log W)$
- IBK can be an exponential time algorithm in the worst case
- Not for nothing was Kassim still undecided when the thieves returned!
- 0-1 Knapsack is an NP-complete problem; a polynomial time algorithm is not known

Memoized Binary Knapsack

A and B are global $(n+1) \times (W+1)$ arrays; B is initialized to all 0; A to all -1

Algorithm 4 MBK(n, W)

if $(A[n, W] = -1)$ **then**
    if $(n = 0$ or $W = 0)$ **then**
        $A[n, W] = 0$
    else
        $a =$ MBK$(n-1,\ W - w_n) + v_n$; $b =$ MBK$(n-1,\ W)$;
        if $a \geq b$ **then**
            $A[n, W] = a$; $B[n, W] = 1$;
        else
            $A[n, W] = b$;
        end if
    end if
end if
return $A[n, W]$

Analysis

- MBK runs in $O(nW)$ time
- Asymptotically as fast as IBK

The most precious metal may not feature in the optimum solution

- Suppose $W = 50$, $w_1 = 10$, $w_2 = 20$, $w_3 = 30$, $v_1 = 60$, $v_2 = 100$, $v_3 = 120$
- Items 1, 2 and 3 cost 6, 5, 4 respectively per unit weight
- Item 1 is the most precious metal
- But $\{2, 3\}$ is the optimal solution