Contents

6.1 0/1 Knapsack

Review: Knapsack (Greedy algorithm)

- Problem:
  - We are given n objects and a knapsack.
  - Object **i** has a weight $\mathbf{w_i}$ and a profit $\mathbf{p_i}$, and the knapsack has a capacity **M**.
  - If a fraction $\mathbf{x_i}$, $0 \le x_i \le 1$, of object i is placed into the knapsack, then the profit of $p_i x_i$ is earned.
  - The objective is to obtain a filling of the knapsack that maximizes the total profit earned.

$$\text{Maximize} \sum_{1 \le i \le n} p_i x_i \text{ subject to } \sum_{1 \le i \le n} w_i x_i \le M$$

  - The solution is a set $(x_1, x_2, \ldots, x_n)$

6.1 0/1 Knapsack

Review: Knapsack (Greedy algorithm)

- Strategy:
  - At each step, we include that object which has the maximum profit per unit of capacity.
  - In the order of the ratio p_i / w_i.
- Example:
  - n = 3
  - M = 20
  - (p_1, p_2, p_3) = (25, 24, 15)
  - (w_1, w_2, w_3) = (18, 15, 10).
  - What is the solution (x_1, x_2, x_3) ?

6.1 0/1 Knapsack

Review: Knapsack (Greedy algorithm)

- Counter example:
  - Can a greedy algorithm solve the following knapsack problem when x_i is only 0 or 1?

- Example:
  - n = 6,
  - M = 100,
  - $(p_1, p_2, p_3, p_4, p_5, p_6) = (40, 35, 18, 4, 10, 2)$,
  - $(w_1, w_2, w_3, w_4, w_5, w_6) = (100, 50, 45, 20, 10, 5)$.

  - What is the solution ?

6.1 0/1 Knapsack

- Principle of optimality
  - The key background of dynamic programming
  - The property of an optimal sequence of decisions is

Whatever the initial state and decision are, the remaining decisions must constitute an optimal decision sequence with regard to the state resulting from the first decision.

6.1 0/1 Knapsack

- Problem
  - Similar to the knapsack problem except that x_i can be either 0 or 1
  - KNAP (1, n, M)

$$\text{maximize} \sum_{i=1}^{n} p_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq M$$

$$x_i = 0 \text{ or } 1$$

6.1 0/1 Knapsack

- Strategy
  - Build a recurrence relation using KNAP (1, n, M)
  - KNAP (1, n, M)
    - If $x_1 = 0$, then KNAP(1, n, M) = KNAP (2, n, M)
    - If $x_1 = 1$, then
      KNAP(1, n, M) = KNAP (2, n, M – w_1)

6.1 0/1 Knapsack

- Strategy
  - Let g_j (y) be the value of an optimal solution to KNAP (j+1, n, y).
  - g_0 (M) is the value of an optimal solution of KNAP (1, n, M).

$$g_0(M) = \max\{g_1(M),\ g_1(M - w_1)\ +\ p_1\}$$

$$g_i(y) = \max\{g_{i+1}(y),\ g_{i+1}(y - w_{i+1})\ +\ p_{i+1}\}$$

6.1 0/1 Knapsack

- Strategy
  - Degenerate case
    - $g_{n-1}(M') = \begin{cases} p_n, & if\ M' \geq w_n \\ 0, & otherwise \end{cases}$
    - $g_i(M') = 0,\ if\ M' < \min(w_{i+1}, \cdots, w_n)$
    - $g_i(M') = \sum_{k=i+1}^{n} p_k,\ \ if\ \sum_{k=i+1}^{n} w_k \leq M'$

6.1 0/1 Knapsack

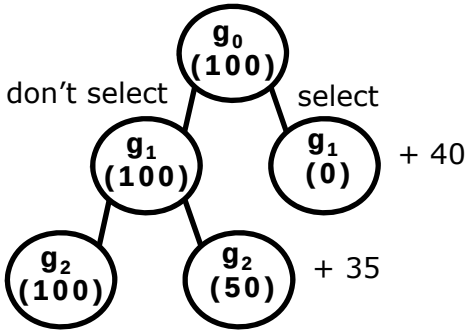
- Example
  - n = 6,
  - M = 100,
  - $(p_1, p_2, p_3, p_4, p_5, p_6) = (40, 35, 18, 4, 10, 2)$,
  - $(w_1, w_2, w_3, w_4, w_5, w_6) = (100, 50, 45, 20, 10, 5)$.

  - What is the solution ?

  - Smart degenerate case?

$$g_i(M) = \sum_{k=i+1}^{n} p_k, if \sum_{k=i+1}^{n} w_k < M$$

6.1 0/1 Knapsack



6.1 0/1 Knapsack

```
int KNAP( int i, int n, int capacity )
{
    if ( capacity < min_w ( i, n ) )
        return 0;

    if ( capacity > sum_w ( i, n ) )
        return sum_p ( i, n );

    return max ( KNAP (i-1, n, capacity-w[i]) + p[i]),
                 KNAP (i-1, n, capacity );
}
```

6.1 0/1 Knapsack

- Time complexity of 0/1 Knapsack problem
  - KNAP (0, n, M) → f(n)
    - KNAP(1, n, M – w_i) + KNAP(1, n, M)
    - f(n-1) + f(n-1)
  - f(n) = 2*f(n-1) + O(1)
  - f(n) = $O(2^n)$

```
int KNAP( int i, int n, int capacity )
{
    if ( capacity < min_w ( i, n ) )
        return 0;

    if ( capacity > sum_w ( i, n ) )
        return sum_p ( i, n );

    return max ( KNAP (i-1, n, capacity-w[i]) + p[i]),
                 KNAP (i-1, n, capacity );
}
```

All about Dynamic Programming

Type	Stepwise approach	Recursive structure	Approach	Time	Specialty
0/1 KNAP	Choosing objects	KNAP(1, n, M) = max {KNAP(2, n, M), KNAP(2, n, M-w_1)+p_1}	Forward	$O(2^n)$	Smart Degenerate Case
Weighted Interval Scheduling					
Multistage Graph					
All Pairs Shortest Path					

6.1 0/1 Knapsack

- 다음 설명 중 옳은 것을 모두 고르시오.

(a) 0/1 knapsack 문제는 greedy algorithm으로 최적의 해를 찾을 수 없다

(b) 0/1 knapsack 문제에 대한 dynamic programming은 backward approach이다.

(c) 0/1 knapsack 문제의 constraint는 knapsack 문제의 constraint와 같다

(d) 0/1 knapsack 문제의 objective function은 knapsack 문제의 objective function과 같다.