Greedy design Strategy

$C = \{a_1, a_2, \dots, a_n\}$

$S = \{a \in C \mid C\}$

Making Change — denominations

$C = \{100, 50, 20, 10, 5, 2, 1\}$

Get the minimum no. of coins — Greedy

$S = \{a_1, a_2, a_3, a_4, a_5\}$

... ⊗ optimal

$C = \{6, 4, 3\}$ — $\{\text{greedy}, \text{opt}\}$

$\{2, 2, 2\}$ — $\{\text{opt}, \text{opt}\}$ (Not optimal)

$S = \{3, 2, 1\}$ ⊗ not optimal

$S = \{3, 3\}$ ⊗ optimal

Subset Problem

```
greedyMyst( C  weights) {
    S ← ∅
    while( C ≠ ∅ and ( weight(S) ))
        a ← greedyMove( C )
        C ← C \ {a}
        if (isFeasible( S ∪ {a} ))
            S ← S ∪ {a}
    }
    if ( C ≠ ∅ and w(S) )
        exp "no solution possible"
}
```

```
greedyChange( int amount,  C = {d1, d2, ..., dk} )
    S ← ∅
    sum ← 0
    while( sum ≠ amount ) {
        a ← max { d ∈ C | sum + d ≤ amount }
        if( { d ∈ C | sum + d ≤ amount } = ∅ )
            return ∅
        S ← S ∪ {a}
        sum ← sum + w(a)
    }
    return (S)
}
```

Non optimal for optimal greedy algorithm

(i) Greedy choice property.

$S^* = \{\text{coin}_1, \text{coin}_2, \dots, \text{coin}_k\}$ ⊗ in S_1

$\Rightarrow$ Algorithm takes greedy optimal decision in return, greedy optimal solution.

(ii) Optimal Substructure property

For the overall solution to be optimal the subproblem solution of the solution must be optimal too.

Ex.3: Scan the map of a state to find the shortest path from w_1 to water $(w_1 \to C_n)$

$S \Rightarrow C \xrightarrow{\text{optimal}} \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_n\}$

$C_1 \to C_2$ ✓ $\{C_1 \to C_n\}$

or: $C_1 \to C_2 \to C_3$ (Not best) $\to$

$C_2 \to C_4 \to C_5$ (Not best)

Knapsack Problem

$C = \{a_1, a_2, \dots, a_n\}$

$a_i \in C$, w_i = weight of a_i

v_i = value of a_i

$\boxed{}$ stack of capacity W

$\sum_{i=1}^{n} w_i \le W$

$S = \{a_1, a_2, \dots, a_n\}$ — functional approach

$\sum_{i=1}^{n} v_i$ is bounded

$S = \{a_1, a_2, \dots, a_n\}$ — using $\sum_{i=1}^{n} w_i \le W$ instead of $\sum_{i=1}^{n} v_i$

maximize $\sum_{i=1}^{n} v_i$

subject to $\sum_{i=1}^{n} w_i \le W$

maximize $\sum_{i=1}^{n} v_i$

subject to $\sum_{i=1}^{n} w_i \le W$

or

$v_i \le c \in \{0, 1\}$

or

optimization problem for functional w/σ

Example

Let $w_i \in \{\dots\}$ $\boxed{W=}$

$C = \{a_1, a_2, \dots, a_6\}$

	a_1	a_2	a_3	a_4	a_5	a_6
$w(\cdot)$	10	60	30	50	40	20
$v(\cdot)$	20	30	45	60	50	25
v_i/w_i	2	1.5	1.5	1.2	1.25	1.25

Greedy Strategy: $S = \{a_1, a_2, a_3, a_4, a_5, a_6\}$

(i) Sort the objects in decreasing order of their values.

$C = \{a_1, a_2, a_3, a_4, a_5, a_6\}$

Implicitly Consider the in sorted order of v_i in the non-increasing order

(ii) Sort w_i in the same order.

$S = \{a_1, a_2, \dots, a_6\}$

$S = \{10, 60, 30, 50, 40, 20\}$

$\sigma(C) = 60 + 30 + 50 + 40$

$G = 2 + 1.5 + 1.5 + 1.2 = 7.2$ unit

(i) Sorted the objects in the non-decreasing order of their weight

$\vec{S} = \{a_1, a_2, a_3, a_4, a_5\}$

$\sigma(\vec{S})$, $\sum G$ = unit

(ii) Sorted the values in the non-increasing order of value per unit weight

$\vec{S} = \{a_1, a_2, a_3, a_4, a_5\}$

$\sigma(\vec{S})$, $\sum G$ = unit ✓

```
fractionalKnapsack(v, w, W)
    S[n] ←
    cur ← 0
    while (cur < W) {
        a_i ← pick the object that has the maximum value per unit weight
        if (w(a_i) + cur ≤ W)
            S[i] ← 1
        else {
            S[i] ← (W − cur) / w(a_i)
            cur ← W
        }
    }
    return S
```

$T(n) = O(n \log n)$ ✓

Max Heap $O(n)$

$n = 10^6$, $m = 10^3$, $n \gg m$ — $O\left(\frac{n}{m} + m \log \frac{n}{m}\right)$

Ex. [illegible] — greedy approach produces an optimal solution.

Input: $C = \{a_1, a_2, \dots, a_n\}$

Assume C is arranged in the non-increasing order of value per unit weight

$\frac{v_1}{w_1} \ge \frac{v_2}{w_2} \ge \frac{v_3}{w_3} \ge \dots \ge \frac{v_n}{w_n} \to ①$

$\vec{S} = \{a_1, a_2, \dots, a_n\}$

Until $\sum \vec{S} \le W$ (if this is the first item $\vec{S}_1$ is clearly optimal)

Until $\sum \vec{S} \ge \sum W$ ✓ — $\sum \vec{S} \le W$ Overall

$\vec{S} = \{\underbrace{1, 1, 1, 1}_{\text{order}}, a_i, \dots\}$

$\sum_{i=1}^{n} \vec{S}_i w_i = W \to ②$

$i < j \Rightarrow \vec{S}_i = 1$

$i = j \Rightarrow \vec{S}_i \in (0, 1)$

$i > j \Rightarrow \vec{S}_i = 0$

$v(\vec{S}) = \sum_{i=1}^{n} \vec{S}_i v_i \to ③$

Let $\vec{x} = \{x_1, x_2, \dots, x_n\}$ be an arbitrary feasible solution

$\sum_{i=1}^{n} x_i w_i \le W \to ④$

from ② & ④

$\Rightarrow \sum_{i=1}^{n} (\vec{S}_i - x_i) w_i \ge 0$ — from ⑤

$v(\vec{x}) = \sum_{i=1}^{n} x_i v_i$

$v(\vec{S}) - v(\vec{x}) = \sum_{i=1}^{n} (\vec{S}_i - x_i) v_i$

$= \sum_{i=1}^{n} (\vec{S}_i - x_i) v_i (\frac{v_i}{w_i})$

$i < j \Rightarrow \vec{S}_i = 1 \Rightarrow (\vec{S}_i - x_i) \ge 0$, $\frac{v_i}{w_i} \ge \frac{v_j}{w_j}$ — case

$i > j \Rightarrow \vec{S}_i = 0 \Rightarrow (\vec{S}_i - x_i) \le 0$, $\frac{v_i}{w_i} \le \frac{v_j}{w_j}$ — case

$i = j$, $\frac{v_i}{w_i} = \frac{v_j}{w_j}$ } case ①

$(\vec{S}_i - x_i) \frac{v_i}{w_i} \ge (\vec{S}_i - x_i) \frac{v_j}{w_j} = \frac{v_j}{w_j}$

$v(\vec{S}) - v(\vec{x}) = \sum_{i=1}^{n} (\vec{S}_i - x_i) v_i \frac{v_j}{w_j}$

$\ge \sum_{i=1}^{n} (\vec{S}_i - x_i) w_i \frac{v_j}{w_j}$

$= \frac{v_j}{w_j} \underbrace{\sum_{i=1}^{n} (\vec{S}_i - x_i) w_i}_{\ge 0}$

$\Rightarrow \vec{S}$ defined by the algorithm is optimal