BITS, PILANI – K. K. BIRLA GOA CAMPUS

Design & Analysis of Algorithms

(CS F364)

Lecture No.11

Applying Greedy Strategy - ReCap

Steps in designing a greedy algorithm

The optimal substructure property holds

(same as dynamic programming)

Cast the optimization problem as one in which we make a choice and are left with one subproblem to solve.

Prove that there is always an optimal solution to the original problem that makes the greedy choice, so that the greedy choice is always safe.

The Fractional Knapsack Problem

Given: A set **S** of ***n*** items, with each item ***i*** having

w_i - a positive **"weight"**

v_i - a positive **"benefit"**

Goal:

Choose items with maximum total benefit but with weight at most ***W.***

And we are allowed to take fractional amounts

The Fractional Knapsack Problem

Possible Greedy Strategies:

- Pick the items in increasing order of weights
- Pick the items in decreasing order of benefits
- Pick the items by decreasing order of value per pound

Note:

1^{st} two strategies do not give optimal solution

Counterexamples - **Exercise**

Greedy Algorithm

We can solve the **fractional knapsack problem** with a **greedy algorithm**:

- Compute the value per pound ***(vi/wi)*** for each item
- Sort (increasing) the items by value per pound
- Greedy strategy of always taking as much as possible of the item remaining which has highest value per pound

Time Complexity:

If there are ***n*** items, this greedy algorithm takes ***O(nlogn)*** time

Optimal Substructure

Optimal Substructure

Given the problem S with an optimal solution X with value V

Suppose X has fraction f_i of the i-th item.

Let X' = X – {fraction of the i-th item}

Claim:

X' is the optimal solution to the subproblem:

S' = S - {fraction of the i-th item}

& the Knapsack capacity $W - f_i w_i$

Proof: Cut-Paste Argument

Greedy Choice Property

- **Theorem**

Consider a knapsack instance P, and let item 1 be item of highest value density

Then there exists an optimal solution to P that uses as much of item 1 as possible (that is, min(w1, W)).

Proof:

Suppose we have a solution Q that uses weight w <min(w1, W) of item 1.

Let w' = min(w1, W) – w

Greedy Choice Property

Q must contain at least weight w′ of some other item(s), since it never pays to leave the knapsack partly empty.

Construct Q* from Q by removing w′ worth of other items and replacing with w′ worth of item 1

Because item 1 has max value per weight, So Q* has total value at least as big as Q.

Alternate Proof

Alternate Proof:

Assume the objects are sorted in order of cost per pound.

Let v_i be the value for item i and let w_i be its weight.

Let x_i be the fraction of object i selected by greedy and let $V(X)$ be the total value obtained by greedy

Alternate Proof

- Suppose wlog
  $\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \cdots \geq \frac{v_n}{w_n}$
- Let $X = (x_1, \ldots, x_n)$ be the solution found by greedy algorithm
- If all $x_i = 1$, then this solution is optimal
- Otherwise, let j be the smallest index such that $x_j < 1$
- Observe, $x_i = 1, i < j$ & $x_i = 0, i > j$

Alternate Proof

- Also $\sum_{i=1}^{n} x_i w_i = W$
- Let value of X be $V(X) = \sum_{i=1}^{n} x_i v_i$
- Let $Y = (y_1, y_2, \cdots y_n)$ be any feasible solution
- Then $\sum_{i=1}^{n} y_i w_i \leq W$

  & so $\sum_{i=1}^{n} (x_i - y_i) \geq 0$

- Let the value of Y be

  $V(Y) = \sum_{i=1}^{n} y_i v_i$

<u>Claim</u>: $V(X) \geq V(Y)$

Now $V(X) - V(Y)$

$$= \sum_{i=1}^{n} (x_i - y_i) v_i$$

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

Alternate Proof

<u>Case 1</u> : $i < j$

$x_i = 1$ & so $(x_i - y_i) \geq 0$

& $\frac{v_i}{w_i} \geq \frac{v_j}{w_j}$

so $(x_i - y_i)\frac{v_i}{w_i} \geq (x_i - y_i)\frac{v_j}{w_j}$

<u>Case 2</u> : $i > j$

$x_i = 0$ & so $(x_i - y_i) \leq 0$

& $\frac{v_i}{w_i} \leq \frac{v_j}{w_j}$

& so

$(x_i - y_i)\frac{v_i}{w_i} \geq (x_i - y_i)\frac{v_j}{w_j}$

Alternate Proof

<u>Case 3</u> : $i = j$

$\frac{v_i}{w_i} = \frac{v_j}{w_j}$

& so

$(x_i - y_i)\frac{v_i}{w_i} \geq (x_i - y_i)\frac{v_j}{w_j}$

$\therefore$ in every case

$(x_i - y_i)\frac{v_i}{w_i} \geq (x_i - y_i)\frac{v_j}{w_j}$

$\therefore V(X) - V(Y)$

$= \sum_{i=1}^{n} (x_i - y_i)\, w_i \frac{v_i}{w_i}$

$\geq \frac{v_j}{w_j} \sum (x_i - y_i)\, w_i$

≥ 0

$\therefore$ we have proved that no feasible solution can have value greater than $V(X)$

$\therefore X$ is optimal solution.

Minimize time in the system

- **Problem Statement:**

- A single server with ***N*** customers to serve
- Customer i will take time t_i, $1 \le i \le N$ to be served.
- **Goal:** Minimize average time that a customer spends in the system.

where time in system for customer i = total waiting time + t_i

- Since ***N*** is fixed, we try to minimize time spend by all customers to reach our goal

Minimize T = $\sum_{i=1}^{N}$ (time in system for customer i)

Minimize time in the system

Example:

Assume that we have 3 jobs with $t_1 = 5$, $t_2 = 3$, $t_3 = 7$

Order	Total Time In System
1, 2, 3	5 + (5 + 3) + (5 + 3 + 7) = 28
1, 3, 2	5 + (5 + 7) + (5 + 7 + 3) = 32
2, 3, 1	3 + (3 + 7) + (3 + 7 + 5) = 28
2, 1, 3	3 + (3 + 5) + (3 + 5 + 7) = 23 ⇐ optimal
3, 1, 2	7 + (7 + 5) + (7 + 5 + 3) = 34
3, 2, 1	7 + (7 + 3) + (7 + 3 + 5) = 32

Minimize time in the system

Brute Force Solution:

Time Complexity: *N!*

Which is exponential!!

Optimal Substructure Property:

Exercise

Minimize time in the system

Greedy Strategy

At each step, add to the end of the schedule, the customer requiring the **least service time** among those who remain.

So **serve least time consuming customer first.**

Minimize time in the system

Observe

- Let $P = p_1p_2.....p_N$ be any permutation of the integers 1 to N and let $s_i = t_{p_i}$

If customers are served in the order corresponding to P, then

Total time passed in the system by all the customers is

$$T(P) = s_1 + (s_1 + s_2) + (s_1 + s_2 + s_3) + ...$$

$$= ns_1 + (n-1)s_2 + (n-2)s_3 + ...$$

$$= \sum_{k=1}^{N} (n - k + 1)s_k$$

Minimize time in the system

Theorem:

Greedy strategy is optimal.

Proof:

Suppose strategy ***P*** does not arrange the customers in increasing service time.

Then we can find two integers ***a*** & ***b*** with ***a < b*** and $s_a > s_b$

i.e, the ***a-th*** customer is **served before** the ***b-th*** customer even though the former needs more service time than the latter.

Minimize time in the system

Now, we exchange the position of these two customers to obtain a new order of service ***O***

Then

$$T(O) = (n - a + 1)s_b + (n - b + 1)s_a + \sum_{\substack{k=1 \\ k \neq a,b}}^{n} (n - k + 1)\, s_k$$

And $\boldsymbol{T(P) - T(O)}$

$= (n - a + 1)(s_a - s_b) + (n - b + 1)(s_b - s_a)$

$= (b - a)(s_a - s_b)$

$\boldsymbol{> 0}$

i.e., the **new** schedule ***O*** is **better than** the **old** schedule ***P***