

LESSON 11: GREEDY METHOD

Objective

In this lesson we are going to learn

- Introduction to Greedy Method
- Cassette Filling and how it helps in Greedy method
- General Knapsack problem
- Job Scheduling
- Backtracking
- queen problem

Points to Ponder

- Greedy Method
- General Knapsack problem
- Job Scheduling
- Backtracking
- queen problem

Greedy Method

Greedy method is a method of choosing a subset of the dataset as the solution set those results in some profit.

Consider a problem having n inputs, we are required to obtain the solution which is a series of subsets that satisfy some constraints or conditions. Any subset, which satisfies these constraints, is called a feasible solution. It is required to obtain the feasible solution that maximizes or minimizes the objective function. This feasible solution finally obtained is called optimal solution.

If one can devise an algorithm that works in stages, considering one input at a time and at each stage, a decision is taken on whether the data chosen results with an optimal solution or not. If the inclusion of a particular data results with an optimal solution, then the data is added into the partial solution set. On the other hand, if the inclusion of that data results with infeasible solution then the data is eliminated from the solution set.

The general algorithm for the greedy method is

1. Choose an element e belonging to dataset D .
2. Check whether e can be included into the solution set S if Yes solution set is $s \cup e$.
3. Continue until s is filled up or D is exhausted whichever is earlier.

11.1 Cassette Filling

Consider n programs that are to be stored on a tape of length L . Each program I is of length l_i where i lies between 1 and n . All programs can be stored on the tape iff the sum of the lengths of the programs is at most L . It is assumed that, whenever a program is to be retrieved the tape is initially positioned at the start end.

Let t_j be the time required retrieving program i_j where programs are stored in the order

$$I = i_1, i_2, i_3, \dots, i_n.$$

The time taken to access a program on the tape is called the mean retrieval time (MRT)

$$\text{i.e., } t_j = \sum_{k=1,2,\dots,j} l_k$$

Now the problem is to store the programs on the tape so that MRT is minimized. From the above discussion one can observe that the MRT can be minimized if the programs are stored in an increasing order i.e., $l_1 \leq l_2 \leq l_3, \dots, l_n$.

Hence the ordering defined minimizes the retrieval time. The solution set obtained need not be a subset of data but may be the data set itself in a different sequence.

Illustration

Assume that 3 sorted files are given. Let the length of files A, B and C be 7, 3 and 5 units respectively. All these three files are to be stored on to a tape S in some sequence that reduces the average retrieval time. The table shows the retrieval time for all possible orders.

Order of recording	Retrieval time	MRT
ABC	$7+(7+3)+(7+3+5)=32$	$32/3$
ACB	$7+(7+5)+(7+5+3)=34$	$34/3$
BAC	$3+(3+7)+(3+7+5)=28$	$28/3$
BCA	$3+(3+5)+(3+5+7)=26$	$26/3$
CAB	$5+(5+7)+(5+7+3)=32$	$32/3$
CBA	$5+(5+3)+(5+3+7)=28$	$28/3$

11.2 General Knapsack problem:

Greedy method is best suited to solve more complex problems such as a knapsack problem. In a knapsack problem there is a knapsack or a container of capacity M n items where, each item i is of weight w_i and is associated with a profit p_i . The problem of knapsack is to fill the available items into the knapsack so that the knapsack gets filled up and yields a maximum profit. If a fraction x_i of object i is placed into the knapsack, then a profit $p_i x_i$ is earned. The constrain is that all chosen objects should sum up to M

Illustration

Consider a knapsack problem of finding the optimal solution where, $M=15$, $(p_1, p_2, p_3, \dots, p_7) = (10, 5, 15, 7, 6, 18, 3)$ and $(w_1, w_2, \dots, w_7) = (2, 3, 5, 7, 1, 4, 1)$.

In order to find the solution, one can follow three different strategies.

Strategy 1: non-increasing profit values

Let (a, b, c, d, e, f, g) represent the items with profit $(10, 5, 15, 7, 6, 18, 3)$ then the sequence of objects with non-increasing profit is (f, c, a, d, e, b, g) .

Item chosen for inclusion	Quantity of item included	Remaining space in M	$P_i X_i$
f	1 full unit	$15-4=11$	$18*1=18$
C	1 full unit	$11-5=6$	$15*1=15$
A	1 full unit	$6-2=4$	$10*1=10$
d	4/7 unit	$4-4=0$	$4/7*7=04$

Profit= 47 units

The solution set is $(1, 0, 1, 4/7, 0, 1, 0)$.

Strategy 2: non-decreasing weights

The sequence of objects with non-decreasing weights is (e, g, a, b, f, c, d) .

Item chosen for inclusion	Quantity of item included	Remaining space in M	$P_i X_i$
E	1 full unit	$15-1=14$	$6*1=6$
G	1 full unit	$14-1=13$	$3*1=3$
A	1 full unit	$13-2=11$	$10*1=10$
b	1 full unit	$11-3=8$	$5*1=05$
f	1 full unit	$8-4=4$	$18*1=18$
c	4/5 unit	$4-4=0$	$4/5*15=12$

Profit= 54 units

The solution set is $(1, 1, 4/5, 0, 1, 1, 1)$.

Strategy 2: maximum profit per unit of capacity used (This means that the objects are considered in decreasing order of the ratio P_i/w_i)

a: $P_1/w_1=10/2 = 5$ b: $P_2/w_2=5/3=1.66$ c: $P_3/w_3=15/5 = 3$

d: $P_4/w_4=7/7=1$ e: $P_5/w_5=6/1=6$ f: $P_6/w_6=18/4 = 4.5$

g: $P_7/w_7=3/1=3$

Hence, the sequence is (e, a, f, c, g, b, d)

Item chosen for inclusion	Quantity of item included	Remaining space in M	$P_i X_i$
E	1 full unit	$15-1=14$	$6*1=6$
A	1 full unit	$14-2=12$	$10*1=10$
F	1 full unit	$12-4=8$	$18*1=18$
C	1 full unit	$8-5=3$	$15*1=15$
g	1 full unit	$3-1=2$	$3*1=3$
b	2/3 unit	$2-2=0$	$2/3*5=3.33$

Profit= 511.33 units

The solution set is $(1, 2/3, 1, 0, 1, 1, 1)$.

In the above problem it can be observed that, if the sum of all the weights is $\leq M$ then all $x_i = 1$, is an optimal solution. If we assume that the sum of all weights exceeds M , all x_i 's cannot be one. Sometimes it becomes necessary to take a fraction of some items to completely fill the knapsack. This type of knapsack problems is a general knapsack problem.

11.3 Job Scheduling

In a job-scheduling problem, we are given a list of n jobs. Every job i is associated with an integer deadline $d_i \geq 0$ and a profit $p_i \geq 0$ for any job i , profit is earned if and only if the job is completed within its deadline. A feasible solution with maximum sum of profits is to be obtained now.

To find the optimal solution and feasibility of jobs we are required to find a subset J such that each job of this subset can be completed by its deadline. The value of a feasible solution J is the sum of profits of all the jobs in J .

Steps in finding the subset J are as follows:

- $\sum p_i, i \in J$ is the objective function chosen for optimization measure.
- Using this measure, the next job to be included should be the one which increases $\sum p_i, i \in J$.
- Begin with $J = \emptyset$ and $\sum p_i = 0, i \in J$
- Add a job to J which has the largest profit
- Add another job to this J keeping in mind the following condition:
 - Search for job which has the next maximum profit.
 - See if this job is union with J is feasible or not.
 - If yes go to step (e) and continue else go to (iv)
 - Search for the job with next maximum profit and go to step (b)
- Terminate when addition of no more jobs is feasible.

