Indian Institute of Information Technology Design and Manufacturing,
Kancheepuram, Chennai 600 127, India
An Autonomous Institute under MHRD, Govt of India
http://www.iiitdm.ac.in
COM 209T Design and Analysis of Algorithms -Lecture Notes

Instructor
N.Sadagopan
Scribe:
Dhanalakshmi S

Greedy Algorithms

Objective: This module focuses on yet another paradigm, namely, greedy algorithms.

Greedy algorithms solve optimization problems by making the best choice (local optimum) at each step. We shall look at the knapsack problem in various perspectives and we solve them using greedy technique. Note that a greedy algorithm do not always yield optimal solutions, but a feasible solution. For example, if the given optimization problem is a minimization problem, then the solution obtained by greedy is always greater than or equal to the optimal solution and if it is a maximization problem, then the solution obtained by greedy is always less than or equal to the optimal solution. This calls for a proof of correctness if greedy indeed works. In general,

Heuristics + Proof of Correctness = Greedy Algorithms

1 0/1 Knapsack

Given a set of n objects, say $x_1, x_2, \ldots, x_n$, along with its weight $w_1, w_2, \ldots, w_n$ and profits $p_1, p_2, \ldots, p_n$. The objective is to find $S \subseteq \{x_1, x_2, \ldots, x_n\}$ such that $Profit(S) = \sum_{x_i \in S} p_i$ is maximum subject to the constraint $\sum_{x_i \in S} w_i \leq W$, where W is the capacity of the knapsack. This problem is called as 0/1 knapsack because for all x_i, either $x_i = 1$ (i.e., the object x_i is included in the knapsack) or $x_i = 0$ (i.e., the object x_i is not included in the knapsack). Recall that this problem can be solved using the dynamic programming paradigm in pseudo-polynomial time ($O(nW)$). Note that if $W = 2^n$, then the algorithm runs in exponential in n. We shall now try a greedy approach for the given problem and typically greedy algorithms run in polynomial time.

Greedy with respect to weight:

Step 1: Sort the weights in increasing order, say $w_1 \leq w_2 \leq \ldots \leq w_n$. This step takes $O(n \log n)$ time.
Step 2: Pick the minimum weight (local optimum), which is w_1 in this case. Check $w_1 \leq W$. If so, add x_1 to S.
Step 3: Check $w_1 + w_2 \leq W$. If so, add x_1 and x_2 to S.
Step 4: Proceed this process until we get the least j such that $w_1 + w_2 + \ldots + w_j > W$. Now return $S = \{x_1, \ldots, x_{j-1}\}$.
The Steps 2-4 take $O(n)$. Overall, the above approach takes $O(n \log n)$ time.
The solution obtained by this greedy approach is a feasible solution, since, $\sum_{x_i \in S} w_i \leq W$.

Query: Is the solution obtained by greedy w.r.t weight is optimum ?
Solution: No. Here is a counter example:
The sorted weights and the corresponding profits are as follows
Given the weight of the knapsack is 5, our algorithm picks the first two objects as an output (Since,

Objects	x_1	x_2	x_3	x_4
Weights	1	2	4	5
Profits	2	2	3	5

$w_1 + w_2 + w_3 = 1 + 2 + 4 = 7 > W$ and $w_1 + w_2 = 1 + 2 = 3 \leq W$) and the corresponding profit is $2 + 2 = 4$.

However, the optimum solution is $\{x_4\}$ and the profit is 5. Thus, the greedy with respect to weight does not yield optimum always.
The greedy with respect to weight fails because in some cases the objects chosen by the algorithm gives less profit than any optimum solution. So, let us try another greedy approach which is greedy with respect to profit.

Greedy with respect to profit:

Step 1: Sort the weights in decreasing order, say $p_1 \geq p_2 \geq \ldots \geq p_n$. Let $w_1, w_2, \ldots, w_n$ be the corresponding weights for $p_1 \geq p_2 \geq \ldots \geq p_n$.
Step 2: Pick the maximum profit (local optimum), which is p_1 in this case. Check $w_1 \leq W$. If so, add x_1 to S.
Step 3: Check $w_1 + w_2 \leq W$. If so, add x_1 and x_2 to S.
Step 4: Proceed this process until we get the least j such that $w_1 + w_2 + \ldots + w_j > W$. Now return $S = \{x_1, \ldots, x_{j-1}\}$.

The solution obtained by this greedy approach is a feasible solution, since, $\sum_{x_i \in S} w_i \leq W$.

Query: Is the solution obtained by greedy w.r.t profit is optimum ?
Solution: No. Here is a counter example:
The sorted profits and the corresponding weights are as follows

Objects	x_1	x_2	x_3	x_4
Profits	4	3	2	1
Weights	5	1	4	5

Given the weight of the knapsack is 5, our algorithm outputs $S = \{x_1\}$ (Since, $w_1 + w_2 = 5 + 1 = 6 > W$ and $w_1 = 5 \leq W$) and the corresponding profit is 4. The optimum solution is $\{x_2, x_3\}$ and the profit is 5. Thus, greedy with respect to profit does not give optimum solution always.

Both the greedy with respect to weight and the greedy with respect to profit fail, because in some cases the objects chosen by the algorithm gives less profit than any optimum. So, now let us try yet another greedy approach with respect to profit per unit weight(profit/weight).

Greedy with respect to profit/weight:

Step 1: Sort the weights in decreasing order, say $p_1/w_1 \geq p_2/w_2 \geq \ldots \geq p_n/w_n$.
Step 2: Pick the maximum p_i/w_i (local optimum), which is p_1/w_1 in this case. Check the corresponding $w_1 \leq W$. If so, add x_1 to S.
Step 3: Check $w_1 + w_2 \leq W$. If so, add x_1 and x_2 to S.
Step 4: Proceed this process until we get the least j such that $w_1 + w_2 + \ldots + w_j > W$. Now return $S = \{x_1, \ldots, x_{j-1}\}$.

The solution obtained by this greedy approach is a feasible solution, since, $\sum_{x_i \in S} w_i \leq W$.

Query: Is the solution obtained by greedy w.r.t profit/weight is optimum ?
Solution: No. Here is a counter example:
The sorted weights and the corresponding profits are as follows

Objects	x_1	x_2	x_3
Profits	7	9	6
Weights	3	5	4
Profit/weight	2.33	1.8	1.5

Given the weight of the knapsack is 5. Our algorithm picks the first object as an output (Since, $w_1 + w_2 =$

$3 + 5 = 8 > W$ and $w_1 = 3 \leq W$) and the corresponding profit is 7. The optimum solution is $\{x_2\}$ and the profit is 5. Thus, the above greedy with respect to profit/weight does not give optimum always.

Fine tuning profit/weight:

Now, let us try to change our strategy slightly by picking the objects as follows. Instead of terminating the algorithm when $w_1 + w_2 + \ldots + w_j > w$ for least j, neglect w_j and proceed with w_{j+1}, and check whether the constraint is satisfied or not, terminate this process once all the objects are scanned.

Query: Is the solution obtained by greedy approach 2 w.r.t profit/weight is optimum ?

Solution: No. Here is a counter example:

The sorted profit/weight and the corresponding weights and profits are as follows

Objects	x_1	x_2	x_3	x_4
Profits	7	9	6	1
Weights	3	5	4	1
Profit/weight	2.33	1.8	1.5	1

Given the weight of the knapsack is 5. Our algorithm picks the object x_1 and x_4 as an output (Since, $w_1 + w_2 = 3 + 5 = 8 > W$, neglect w_2 and proceed from w_3, $w_1 + w_3 = 3 + 4 = 7 > W$, neglect w_3 and proceed from w_4, $w_1 + w_4 = 3 + 1 = 4 < W$) and the corresponding profit is 8. The optimum solution is $\{x_2\}$ and the profit is 9. Thus, greedy with respect to profit/weight is not optimum.

2 Variant Knapsack

Given a set of n objects, say $x_1, x_2, \ldots, x_n$, along with its weight $w_1, w_2, \ldots, w_n$ and profits $p_1, p_2, \ldots, p_n$. The objective is to find $S \subseteq \{x_1, x_2, \ldots, x_n\}$ such that $\mid S \mid$ is maximum subject to the constraint $\sum_{x_i \in S} w_i \leq W$, where W is the capacity of the knapsack.

Greedy with respect to weight

Step 1: Sort the weights in increasing order, say $w_1 \leq w_2 \leq \ldots \leq w_n$.

Step 2: Pick the minimum weight (local optimum), which is w_1 in this case. Check $w_1 \leq W$. If so, add x_1 to S.

Step 3: Check $w_1 + w_2 \leq W$. If so, add x_1 and x_2 to S.

Step 4: Proceed this process until we get the least j such that $w_1 + w_2 + \ldots + w_j > W$. Now return $S = \{x_1, \ldots, x_{j-1}\}$.

The solution obtained by this greedy approach is a feasible solution, since, $\sum_{x_i \in S} w_i \leq W$.

Example: The sorted weights and the corresponding profits are as follows

Objects	x_1	x_2	x_3	x_4	x_5
Weights	1	2	4	5	6
Profits	3	5	6	1	8

Given the weight of the knapsack is 10. Our algorithm picks the objects $\{x_1, x_2, x_3\}$ as an output (Since, $w_1 + w_2 + w_3 + w_4 = 1 + 2 + 4 + 5 = 12 > W$ and $w_1 + w_2 + w_3 = 1 + 2 + 4 = 7 < W$) and $\mid S \mid = 3$. Optimum Solutions: $\{x_1, x_2, x_5\}$, $\{x_1, x_3, x_4\}$, $\{x_1, x_2, x_4\}$ and $\{x_1, x_2, x_3\}$. Thus, any optimum solution will pick exactly three objects from $\{x_1, x_2, x_3, x_4, x_5\}$. Interestingly, this greedy strategy gives optimum always which we shall prove next.

Claim: Greedy variant knapsack indeed returns an optimum solution (OPT).

Proof: Let $x_1, x_2, \ldots, x_n$ be the set of objects with weights $w_1, w_2, \ldots, w_n$ and let the weight of the knapsack be W. Sort $w_1, \ldots, w_n$ and rename the corresponding objects as $a_1, \ldots, a_n$. Thus, $a_1 \leq a_2 \leq \ldots \leq a_n$.

Let $A = \begin{pmatrix} a_1 & a_2 & \dots & a_{k-1} & a_k & \dots & a_n \\ 1 & 1 & \dots & 1 & 0 & \dots & 0 \end{pmatrix}$ be the output of the algorithm. i.e., $S = \{a_1, \dots, a_{k-1}\}$ is the solution obtained by the algorithm.
Let $B = \begin{pmatrix} b_1 & b_2 & \dots & b_k & b_{k+1} & \dots & b_n \\ 1 & 1 & \dots & 0 & 1 & \dots & 0 \end{pmatrix}$ be any feasible solution such that $b_1 = a_1, b_2 = a_2, \dots, b_n = a_n$.

Remark: We work with the binary vector (solution vector) of OPT and the algorithm to show that the number of 1's in both are same. It is important to note that the behavior of OPT is unknown to the algorithm, i.e., why did OPT set $b_k = 0$ and $b_{k+1} = 1$, and what is only known to the algorithm is the solution vector output by OPT.
We shall show next that $\sum b_i = \sum a_i$. Our proof consists of two parts. In sub claim-1, we show that $\sum b_i \geq \sum a_i$. Further, in sub claim-2, we present a stronger claim: for any feasible solution B (which includes OPT as well) , $\sum a_i \geq \sum b_i$.

Observation 1: There exists a $k \in \{1, \dots, n\}$ such that $a_i = 1, 1 \leq i \leq k-1$ and $a_j = 0, k \leq j \leq n$.

Thus, to show that the algorithm is OPT, we need to establish $\sum a_i \geq \sum b_i$. Towards this end, we prove the following sub claim.

Sub Claim 1: If B is OPT, then $\sum b_i \geq \sum a_i$.
Since, the algorithm outputs a feasible solution and since, the number of $1's$ in $OPT \geq$ the number of $1's$ in any feasible solution, the above claim is true.

Sub Claim 2: If B is any feasible solution, then $\sum a_i \geq \sum b_i$.
Proof of sub claim 2: Proof is by mathematical induction on m, where m is the number of bits in which A and B differ.
Base Case: $m = 0$. Clearly, $\sum a_i \geq \sum b_i$.
Hypothesis: Assume that the claim is true if they differ in less than m-places, $m \geq 1$.
Induction Step: Let A and B differ in $(m+1)$-places, $m \geq 0$.

Observation 2: Since, A and B differ in $(m+1)$-places there exists j such that $a_j \neq b_j$. Clearly, it can not be the case that such a $j \in \{k, \dots, n\}$. If so, then B is not a feasible solution. This is true because, if B matches with A in the first $(k-1)$ places and differ in $j \in \{k, \dots, n\}$, then clearly B has exceeded the capacity of the knapsack and hence B is infeasible.
Therefore, there exists $j, 1 \leq j \leq k-1$ such that $a_j = 1$ and $b_j = 0$.
Find the least $j \in \{1, \dots, k-1\}$ such that $a_j = 1$ and $b_j = 0$.
Now, set $b_j = 1$ (For example, if $A = (1, 1, 1, 1, 0, 0, \dots, 0)$ and $B = (1, 1, 0, 0, 0, 1, \dots, 0)$, then the modified B is $C = (1, 1, 1, 0, 0, 1, \dots, 0)$). By doing so, the modified solution vector of B, say C, may become infeasible.

Case 1: C is feasible
The number of positions in which A and C differs is m. By the hypothesis, $\sum a_i \geq \sum c_i$. Note that, $\sum c_i \geq \sum b_i$. Thus, $\sum a_i \geq \sum b_i$. Hence, the sub claim is true if C is feasible.

Case 2: C is not feasible
Since C is not feasible, there exists a $j \in \{k, \dots, n\}$ such that $a_j = 0$ and $c_j = 1$. Now, set $c_j = 0$. This implies, A and the modified C, say D, differs in $(m-1)$-places and by the induction hypothesis, $\sum a_i \geq \sum d_i$. Observe that $\sum d_i = \sum b_i$. Thus, $\sum a_i \geq \sum b_i$.

From all the above sub claims, we can conclude that $\sum a_i = \sum b_i$. Hence, the greedy variant knapsack is OPT.

3 Fractional Knapsack

Given a set of n objects, say $x_1, x_2, \ldots, x_n$, along with its weight $w_1, w_2, \ldots, w_n$ and profits $p_1, p_2, \ldots, p_n$. The objective is to find a subset $S \subseteq \{fx_1, fx_2, \ldots, fx_n\}$, such that $Profit(S) = \sum_{x_i \in S} fx_i \cdot p_i$ is maximum subject to the constraint $\sum_{x_i \in S} w_i \leq W$, where fx_i is the fraction of x_i and $0 \leq fx_i \leq 1$ (so far, we are allowed to include x_i or to exclude x_i, now, we can choose a fraction of x_i which varies between 0 and 1), and W is the capacity of the knapsack. Now let us try to establish a greedy approach for the given problem as follows and we will also check whether the solution obtained by the greedy is optimum or not.

Greedy Strategy 1: with respect to weight:

Step 1: Sort the weights in increasing order, say $w_1 \leq w_2 \leq \ldots \leq w_n$.
Step 2: Pick the minimum weight (local optimum), which is w_1 in this case. Check $w_1 \leq W$. If so, add x_1 to S.
Step 3: Check $w_1 + w_2 \leq W$. If so, add x_1 and x_2 to S.
Step 4: Proceed this process until we get the least j such that $w_1 + w_2 + \ldots + w_j > W$. With respect to w_j, calculate $fx_j = \frac{W - \sum_{i=1}^{j-1} w_i}{w_j}$. Now return $S = \{x_1, \ldots, x_{j-1}, fx_j\}$.

The solution obtained by this greedy approach is a feasible solution, since, $\sum_{x_i \in S} (fx_i \cdot w_i) \leq W$.

Example: The sorted weights and the corresponding profits are as follows

Objects	x_1	x_2	x_3
Weights	1	2	5
Profits	1	3	5

Given the weight of the knapsack is 5. Our algorithm first chooses $\{x_1, x_2\}$ (Since, $w_1+w_2+w_3 = 2+3+5 = 10 > W$ and $w_1 + w_2 = 2 + 3 = 5 \leq W$) and then it chooses fx_3 amount of x_3, where, $fx_3 = \frac{5-(1+2)}{5} = \frac{2}{5}$. Thus, the algorithm outputs $(x_1, x_2, x_3) = (1, 1, \frac{2}{5})$ and the corresponding profit is $1 + 3 + \frac{2}{5} \times 5 = 6$, which is same as the optimum.

Query: Is the solution obtained by fractional greedy w.r.t weight is optimum ?
Solution: No. Here is a counter example:
The sorted weights and the corresponding profits are as follows

Objects	x_1	x_2	x_3
Weights	2	3	5
Profits	5	8	15

Given the weight of the knapsack is 5. Our algorithm picks the first two objects as an output (Since, $w_1+w_2+w_3 = 2+3+5 = 10 > W$ and $w_1+w_2 = 2+3 = 5 \leq W$) and the corresponding profit is $5+8 = 13$. The optimum solution is $\{x_3\}$ and the profit is 15. Thus, greedy with respect to weight is not optimum.

Greedy Strategy 2: with respect to profit:

Step 1: Sort profits in decreasing order, say $p_1 \geq p_2 \geq \ldots \geq p_n$. Let $w_1, w_2, \ldots, w_n$ be the corresponding weights for $p_1 \geq p_2 \geq \ldots \geq p_n$.
Step 2: Pick the maximum profit (local optimum), which is p_1 in this case. Check $w_1 \leq W$. If so, add x_1 to S.
Step 3: Check $w_1 + w_2 \leq W$. If so, add x_1 and x_2 to S.
Step 4: Proceed this process until we get the least j such that $w_1 + w_2 + \ldots + w_j > W$. With respect to w_j, calculate $fx_j = \frac{W - \sum_{i=1}^{j-1} w_i}{w_j}$. Now return $S = \{x_1, \ldots, x_{j-1}, fx_j\}$.

The solution obtained by this greedy approach is a feasible solution, since, $\sum_{x_i \in S} (fx_i \cdot w_i) \leq W$.

Example: The sorted profits and the corresponding weights are as follows

Objects	x_1	x_2	x_3
Profits	5	1	1
Weights	4	2	1

Given the weight of the knapsack is 5. Our algorithm first chooses $\{x_1\}$ (Since, $w_1 + w_2 = 4 + 2 = 6 > W$ and $w_1 = 4 \leq W$) and then it chooses fx_2 amount of x_2, where, $fx_2 = \frac{5-(4)}{2} = \frac{1}{2} = 0.5$. Thus, the algorithm outputs $(x_1, x_2, x_3) = (1, 1/2, 0)$ and the corresponding profit is $5 + \frac{1}{2} \times 1 = 5.5$, where as the OPT chooses $\{x_1, x_3\}$ and the corresponding profit is 6.

Query: Is the solution obtained by fractional greedy w.r.t weight is optimum ?
Solution: No. The above example is a counter example:

Greedy Strategy 3: with respect to profit/weight:

Step 1: Calculate the profit/weight for all objects and sort it in decreasing order, say $p_1/w_1 \geq p_2/w_2 \geq \ldots \geq p_n/w_n$.
Step 2: Pick the maximum profit/weight (local optimum), which is p_1/w_1 in this case. Check $w_1 \leq W$. If so, add x_1 to S.
Step 3: Check $w_1 + w_2 \leq W$. If so, add x_1 and x_2 to S.
Step 4: Proceed this process until we get the least j such that $w_1 + w_2 + \ldots + w_j > W$. With respect to w_j, calculate $fx_j = \frac{W - \sum_{i=1}^{j-1} w_i}{w_j}$. Now return $S = \{x_1, \ldots, x_{j-1}, fx_j\}$.

The solution obtained by this greedy approach is a feasible solution, since, $\sum_{x_i \in S} (fx_i \cdot w_i) \leq W$.

Example: The sorted order of profit/weight, and the corresponding weights and profits are as follows

Objects	x_1	x_2	x_3	x_4
Profits	7	9	6	1
Weights	3	5	4	1
Profit/weight	2.33	1.8	1.5	1

Given the weight of the knapsack is 5. Our algorithm first picks the object x_1 (Since, $w_1 + w_2 = 3 + 5 = 8 > W$ and $w_1 < W$) and then it picks fx_2, where $fx_2 = \frac{5-3}{5} = \frac{2}{5}$. Thus the algorithm outputs $(x_1, x_2, x_3, x_4) = (1, 2/5, 0, 0)$ and the corresponding profit is $7 + \frac{2}{5} \times 9 = 10.6$, which is same as optimum solution.

Claim: Greedy fractional knapsack indeed returns an optimum solution (OPT).
Proof: Let $x_1, x_2, \ldots, x_n$ be the set of objects with weights $w_1, w_2, \ldots, w_n$ and profits $p_1, p_2, \ldots, p_n$, respectively. Let the weight of the knapsack be W. Sort $p_1/w_1, \ldots, p_n/w_n$ and without loss of generality assume that, $p_1/w_1 \geq \ldots \geq p_n/w_n$.
Let $A =$ be the output of the algorithm and Let B be the solution output by any OPT algorithm.

Observation 1: (with respect to A) There exist a $k \in \{1, \ldots, n\}$ such that $a_j = 1, 1 \leq j \leq k-1$, $a_k, 0 \leq a_k \leq 1$ and $a_l = 0, k+1 \leq l \leq n$.

The objective is to show that profit earned by the algorithm is equal to the profit earned by OPT. Compare A and B. If they are equal, then there is nothing to prove. If $A \neq B$:

- Find least r such that $a_r \neq b_r$, such r exists in the range $\{1, \ldots, k\}$, if not, B is not feasible. For example, $A = (1, 1, 1, 2/3, 0, \ldots, 0)$ and $B = (0, 1, 1/4, 1/7, 1, \ldots, 0)$.

Observation 2: $a_r > b_r$.

- Increase b_r to a_r and accordingly decrease the weights among $(b_{r+1}, \ldots, b_n)$. Let C be the modified configuration of B.

- Due to increase in the value of r^{th} bit (b_r is increased to a_r), there will be an increase in profit at r^{th} bit. Since, the increase at r^{th} bit is compensated appropriately by decreasing the values at $(r+1)^{th}$ bit, $(r+2)^{nd}$ bit, and so on, there will be a decrease in profit at these bits. We shall now calculate the modified profit as follows.

- Profit gained by $C = \sum c_i \cdot p_i = \sum b_i \cdot p_i + (c_r - b_r) \times w_r \times \frac{p_r}{w_r}$
$- (b_{r+1} - c_{r+1}) w_{r+1} \times \frac{p_{r+1}}{w_{r+1}} - (b_{r+2} - c_{r+2}) w_{r+2} \times \frac{p_{r+2}}{w_{r+2}} - \ldots - (b_n - c_n) w_n \times \frac{p_n}{w_n}$,
where, $(c_r - b_r) w_r \times \frac{p_r}{w_r}$ is the increase in profit w.r.t the r^{th} bit and $-(b_{r+1} - c_{r+1}) w_{r+1} \times \frac{p_{r+1}}{w_{r+1}}$ denotes the decrease in profit at $(r+1)^{th}$ bit.

  Since $p_1/w_1 \geq \ldots \geq p_n/w_n$, $\frac{p_r}{w_r} \geq \frac{p_{r+1}}{w_{r+1}} \geq \frac{p_{r+2}}{w_{r+2}} \ldots$. Using this we shall simplify the above expression as,
  $\therefore, \sum c_i \cdot p_i \geq \sum b_i \cdot p_i + (c_r - b_r) \times w_r \times \frac{p_r}{w_r}$
  $- (b_{r+1} - c_{r+1}) w_{r+1} \times \frac{p_r}{w_r} - (b_{r+2} - c_{r+2}) w_{r+2} \times \frac{p_r}{w_r} - \ldots - (b_n - c_n) w_n \times \frac{p_r}{w_r}$

  Thus, $\sum c_i \cdot p_i \geq \sum b_i \cdot p_i + \frac{p_r}{w_r} \left[(c_r - b_r) w_r - \sum_{i=r+1}^{n} (b_i - c_i) \cdot w_i \right]$

  Note that $\left[(c_r - b_r) w_r - \sum_{i=r+1}^{n} (b_i - c_i) \cdot w_i \right] = 0$ as increase at r^{th} bit is compensated with bits in $(r+1), \ldots, n$.

  Therefore, $\sum c_i \cdot p_i \geq \sum b_i \cdot p_i \Rightarrow$ we now have either $\sum c_i \cdot p_i > \sum b_i \cdot p_i$ or $\sum c_i \cdot p_i = \sum b_i \cdot p_i$
  If $\sum c_i \cdot p_i > \sum b_i \cdot p_i$, then it contradicts the optimality of B. Thus, $\sum c_i \cdot p_i = \sum b_i \cdot p_i$.

- Now, check whether $A = C$ or not. If $A = C$, then the proof is done. If $A \neq C$, then repeat the above process till $A = C$. Note that at the end of the first iteration of the above proof, if $A \neq C$, then they will match at first r places and may differ at any of the places in the range $r+1, \ldots, n$. By repeating the above argument (at max n iterations), we eventually see that the behavior of the greedy algorithm and optimum are same.

- Thus, we conclude the output of greedy is indeed optimum.

Acknowledgements: Lecture contents presented in this module and subsequent modules are based on the following text books and most importantly, author has greatly learnt from lectures by algorithm exponents affiliated to IIT Madras/IMSc; Prof C. Pandu Rangan, Prof N.S.Narayanaswamy, Prof Venkatesh Raman, and Prof Anurag Mittal. Author sincerely acknowledges all of them. Special thanks to Teaching Assistants Mr.Renjith.P and Ms.Dhanalakshmi.S for their sincere and dedicated effort and making this scribe possible. Author has benefited a lot by teaching this course to senior undergraduate students and junior undergraduate students who have also contributed to this scribe in many ways. Author sincerely thanks all of them.

References:

1. E.Horowitz, S.Sahni, S.Rajasekaran, Fundamentals of Computer Algorithms, Galgotia Publications.
2. T.H. Cormen, C.E. Leiserson, R.L.Rivest, C.Stein, Introduction to Algorithms, PHI.
3. Sara Baase, A.V.Gelder, Computer Algorithms, Pearson.