Greedy Paradigm, Optimal Sequential Storage, Knapsack Problem

Textbook Reading:

Chapter 6, pp. 242-251

General Paradigm

```
procedure Greedy(S,Solution)
Input:  S (base set)  //it is assumed that there is an associated objective
                      //function f defined on (possibly ordered) subsets of S
Output: Solution  (an ordered subset of S that potentially optimizes the objective
                           function f, or a message that Greedy doesn't even
                           produce a solution (optimal or not))
      PartialSolution ← $\varnothing$  //initialize the partial solution to be empty
      R ← S
      while PartialSolution is not a solution .and. R $\neq \varnothing$ do
            x ← GreedySelect(R)
            R ← R \ {x}
            if PartialSolution $\cup$ {x} is feasible then
                  PartialSolution ← PartialSolution $\cup$ {x}
            endif
      endwhile
      if  PartialSolution is a solution then
            Solution ← PartialSolution
      else
          write("Greedy fails to produce a solution")
      endif
end Greedy
```

Optimal Sequential Storage Order

Suppose we are given a set of objects $b_0, b_1, \ldots b_{n-1}$ arranged sequentially in some order $b_{\pi(0)}, b_{\pi(1)},\ldots, b_{\pi(n-1)}$, where π is some permutation of $\{0,1,\ldots,n-1\}$, such as files stored on a sequential access tape.

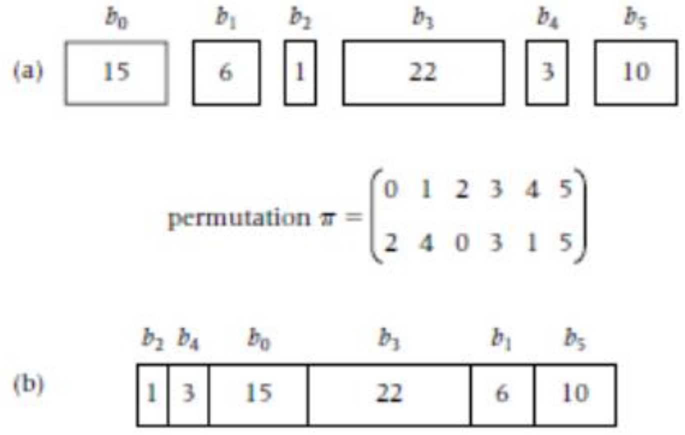
We assume that there is a weight c_i associated with object b_i (for example, the length of a file) that represents the individual cost of processing b_i.

We also assume that to process object $b_{\pi(i)}$, we must first process the objects $b_{\pi(0)}, b_{\pi(1)},\ldots, b_{\pi(i-1)}$ that precede $b_{\pi(i)}$ in the sequential arrangement.

For example, to read the i^{th} file on a sequential tape (assuming that the tape is rewound to the beginning), we must first read the preceding $i - 1$ files on the tape.

Thus the total cost of processing $b_{\pi(i)}$ is $c_{\pi(0)} + c_{\pi(1)} + \ldots + c_{\pi(i)}$.

Below we illustrate this example with six files b_0,b_1,...,b_5 of lengths 15,6,1,22,3,10, respectively. A sample storage order of these files on a tape is given in Figure 6.1b. If we wish to read the file of length 22, we first have to read the files of length 1,3,15 that precede this file on the tape. Thus, reading the file of length 22 results in reading files whose total length is 1 + 3 + 15 + 22



(a)
b_0 15
b_1 6
b_2 1
b_3 22
b_4 3
b_5 10
permutation $\pi = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 0 & 3 & 1 & 5 \end{pmatrix}$
(b)
b_2 1
b_4 3
b_0 15
b_3 22
b_1 6
b_5 10

Formulation of Problem

The **optimal sequential storage** problem for processing objects $b_0, b_1, \ldots, b_{n-1}$ is to find an arrangement (permutation) of the objects $b_{\pi(0)}, b_{\pi(1)}, \ldots, b_{\pi(n-1)}$ that minimizes the average time it takes to process an object.

Let p_i denote the probability that the object b_i is to be processed, $i = 0, \ldots, n - 1$. Given these probabilities and a permutation π, clearly the average total cost of processing an object is given by

$$AveCost(\pi) = p_{\pi(0)}(c_{\pi(0)}) + p_{\pi(1)}(c_{\pi(0)} + c_{\pi(1)}) + \cdots + p_{\pi(n-1)}(c_{\pi(0)} + \cdots + c_{\pi(n-1)}).$$

We first consider the special case where each object is equally likely to be processed, so that $p_i = 1/n$, $i = 1,\ldots, n$. Then

$$\begin{aligned} AveCost(\pi) &= \frac{1}{n}[(c_{\pi(0)}) + (c_{\pi(0)} + c_{\pi(1)}) + \cdots + (c_{\pi(0)} + c_{\pi(1)} + \cdots + c_{\pi(n-1)})] \\ &= \frac{1}{n}\sum_{i=0}^{n-1} (n-i)c_{\pi(i)}. \end{aligned}$$

- Observe that the coefficients of $c_{\pi(i)}$, $i = 0,\ldots,n - 1$, decrease as i increases. Thus, it is intuitively obvious that *AveCost*(π) is minimized when

$$c_{\pi(0)} \leq c_{\pi(1)} \leq \ldots \leq c_{\pi(n-1)}$$

- So that the greedy strategy we use is to select the objects in increasing order of their costs.

Problem with General Probabilities

- Now consider the problem of minimizing $AveCost(\pi)$ as given for a general set of probabilities p_i, $i = 1,...,n$.
- Various greedy strategies come to mind. For example, we might use the same greedy strategy given previously, which give the optimal solution when all the probabilities are equal. However, it is not hard to find examples where this strategy yields suboptimal results.
- We might also arrange the objects in decreasing order of probabilities (placing the objects most likely to be processed near the beginning), but, again, it is not hard to find examples where this strategy fails.
- What works in yielding an optimal solution is arranging the objects in increasing order of the ratios c_i/p_i

Knapsack Problem

- Given n objects $b_0,b_1,\ldots,b_{n-1}$ and a knapsack of capacity C.
- With each object b_i associate a positive weight (or volume) w_i, and a positive value v_i (or profit, cost, and so forth) $i = 0,\ldots, n - 1$.
- We place the objects or fractions of objects in the knapsack, taking care that the capacity of the knapsack is not exceeded.
- If a fraction f_i of object b_i is placed in the knapsack ($0 \leq f_i \leq 1$), then it contributes $f_i v_i$ to the total value in the knapsack.
- The Knapsack problem is to place objects or fractions of objects in the knapsack, without exceeding the capacity, so that total value of the objects in the knapsack is maximized.

More Formal Definition

Given positive real weights $w_i, v_i, i = 0, \dots, n-1$, representing and capacity C, find fractions $f_i, i = 0, \dots, n-1$ that

$$\text{maximize} \sum_{i=0}^{n-1} f_i v_i$$

subject to the constraints:

$$\sum_{i=0}^{n-1} f_i w_i \leq C$$

$$0 \leq f_i \leq 1, i = 0, \dots, n-1.$$

Greedy Method Solution to Knapsack Problem

Density is the ratio of the value/weight, i.e., the density d_i of object b_i is given by

$$d_i = \frac{v_i}{w_i}, i = 0, \dots, n-1.$$

- Sort the objects in **decreasing order of their densities**.
- Keep choosing the object of highest density from the remaining objects and reducing the capacity by its weight until the knapsack is full or an object is encountered that doesn't fit into the knapsack.
- Chose the fraction of the object that doesn't fit so that it fills up the knapsack.

Example

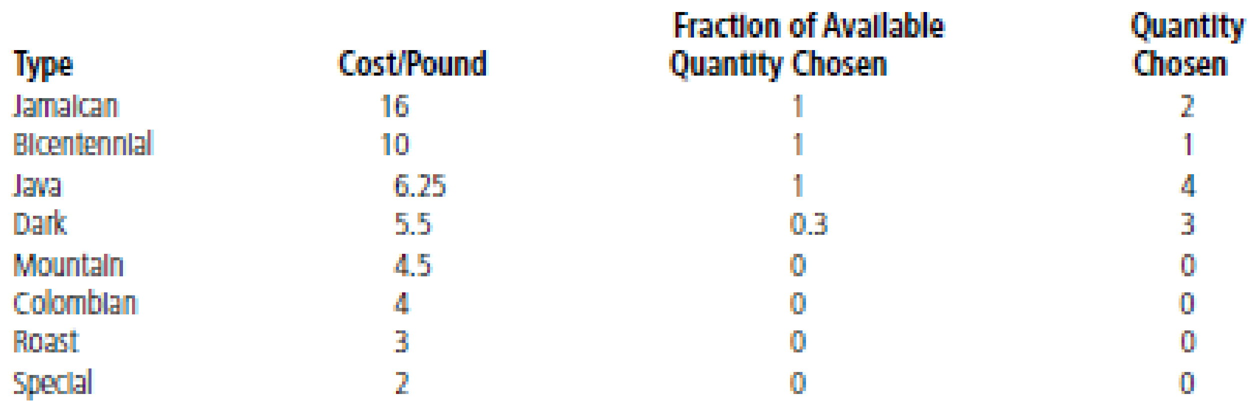
Make the most expensive 10-pound bag, i.e., bag is knapsack with capacity C = 10 and weights and values are given as follows.

Type	Quantity Available In Pounds	Total Cost In Dollars for That Quantity
Colombian	8	32
Jamaican	2	32
Java	4	25
Bicentennial	1	10
Mountain	2	9
Roast	6	18
Dark	10	55
Special	50	100

Coffee Shop Inventory

Solution

Sort in descending order of density, i.e., price and keep choosing as much of the most pricey coffee as possible until coffee bag is full.



Type	Cost/Pound	Fraction of Available Quantity Chosen	Quantity Chosen
Jamaican	16	1	2
Bicentennial	10	1	1
Java	6.25	1	4
Dark	5.5	0.3	3
Mountain	4.5	0	0
Colombian	4	0	0
Roast	3	0	0
Special	2	0	0

Value of most expensive 10 pound bag of coffee is
1×32 + 1×10 + 1×25 + .3×55 = 83.5

PSN. Solve the following instance of the knapsack problem with capacity $C = 15$.

object	b_0	b_1	b_2	b_3	b_4
value	100	20	30	15	66
weight	50	2	10	4	12
density	2	10	3	3.75	5.5

Psuedocode

```
Procedure Knapsack($V[0:n-1]$, $W[0:n-1]$, $C$, $F[0:n-1]$)
Input: $V[0:n-1]$ (an array of positive values)
       $W[0:n-1]$ (an array of positive weights)
       $C$ (a positive capacity)
Output: $F[0:n-1]$ (array of nonnegative fractions)
  Sort the arrays $V[0:n-1]$ and $W[0:n-1]$ in decreasing order of densities, so that
      $V[0]/W[0] \geq V[1]/W[1] \geq \ldots \geq V[n-1]/W[n-1]$
   for $i \leftarrow 0$ to $n-1$do
            $F[i] \leftarrow 0$
       endfor
       $RemainCap \leftarrow C$
       $i \leftarrow 0$
       if $W[0] \leq C$ then
            $Fits \leftarrow$ .true.
       else
            $Fits \leftarrow$ .false.
       endif
       while $Fits$ .and. $i \leq n-1$ do
            $F[i] \leftarrow 1$
            $RemainCap \leftarrow RemainCap - W[i]$
            $i \leftarrow i + 1$
            if $W[i] \leq RemainCap$ then
                 $Fits \leftarrow$ .true.
            else
                 $Fits \leftarrow$ .false.
            endif
       endwhile
       if $i \leq n-1$ then $F[i] \leftarrow RemainCap/W[i]$ endif
end Knapsack
```

Proof of Correctness by Contradiction

- Assume $F = (f_0, f_1, ..., f_{n-1})$ generated by *Knapsack* is suboptimal.
- Consider an optimal solution $Y = (y_0, y_1, ..., y_{n-1})$. Note that $\sum_{i=0}^{n-1} y_i w_i = C$
- Let j denote the first index i such that $f_i \neq y_i$ so that $1 = f_i = y_i$, $0 \leq i \leq j - 1$.
- Choose Y such that j is maximized over all optimal solutions.
- By altering Y, we now construct an optimal solution $Z = (z_0, z_1, ..., z_{n-1})$ that agrees with F in one more initial position. We set $z_i = f_i$, $0 \leq i \leq j$, and for $i > j$ the value of z_i is obtained by suitably decreasing y_i so that $\sum_{i=0}^{n-1} z_i w_i = C$
- Comparing the total values of the solutions Z and Y, we obtain $(z_j - y_j)w_j = \sum_{i=j+1}^{n} (y_i - z_i)w_i.$

$$\sum_{i=0}^{n-1} z_i v_i - \sum_{i=0}^{n-1} y_i v_i = (z_j - y_j)v_j - \sum_{i=j+1}^{n-1} (y_i - z_i)v_i.$$

$$= (z_j - y_j)w_j v_j / w_j - \sum_{i=j+1}^{n-1} (y_i - z_i)w_i v_i / w_i$$

$$\geq \left[(z_j - y_j)w_j - \sum_{i=j+1}^{n-1} (y_i - z_i)w_i\right] v_j / w_j \quad (\text{since } y_i \geq z_i$$

$$\text{and } v_0 / w_0 \geq v_1 / w_1 \geq \cdots \geq v_{n-1} / w_{n-1})$$

$$= 0 \quad .$$

Conclusion of Proof

- We have shown that the value of *Z* is not less than the value of *Y*, so that *Z* is also an optimal solution.
- But *Z* agrees with *F* in the first *j*+1 initial positions, which contradicts the assumption that *Y* is an optimal solution that agrees with *F* in the most initial positions.

Complexity Analysis

Except for the $\Theta(n \log n)$ sorting step, the remainder of *Knapsack* has linear complexity in the worst case.

0/1 Knapsack

A natural variation of the Knapsack problem, called the **0/1 Knapsack problem**, is the same as the Knapsack problem except that fractional objects are not allowed.

Formal Definition of 0/1 Knapsack

Given positive real weights $w_i, v_i, i = 0, \dots, n-1$, representing and capacity C, find fractions $f_i, i = 0, \dots, n-1$ that

$$\text{maximize} \sum_{i=0}^{n-1} f_i v_i$$

subject to the constraints:

$$\sum_{i=0}^{n-1} f_i w_i \leq C$$

$$f_i \in \{0,1\},\ i = 0, \dots, n-1.$$

Approximation to 0/1 Knapsack using Greedy?

Since the greedy solution to the Knapsack Problem chooses whole objects, except for possibly the last object chosen, we can use this to get a solution to the 0/1 knapsack by simply removing the last object if a fraction of it is chosen.

PSN. Show that this greedy solution fails to give a good approximation to the 0/1 Knapsack. In particular, show that for any given ϵ, there are examples where it gives worse than an M-approximation, i.e., the ratio of the value of the an optimal solution to the value of the greedy solution is greater than *M*. For example, if $M = 1000000$, then the value of the optimal solution is more than one million times greater.

What do you call a greedy crab?

Shell fish