Inventory Planning

Definition

Inventory Planning

Scientifically determine what inventory is needed to achieve service levels, given the existing flows and anticipated supply and demand fluctuations. Calculate and maintain the optimal safety stock and replenishment orders. Project future investment, procurement and warehousing requirements for their business.

Problem Statement

The Rinky Dink Company makes machines that resurface ice rinks. The demand for such products varies from month to month, and so the company needs to develop a strategy to plan its manufacturing given the fluctuating, but predictable, demand.

so we have to minimize the cost of manufacturing the machines of the company to completion the demand for such products which varies from month to month

Algorithm

```
INVENTORY-PLANNING(n, m, c, D, d, h)
  let cost[1..n, 0..D] and make[1..n, 0..D] be new tables
  // Compute cost[n, 0..D] and make[n, 0..D].
  for s = 0 to D
      f = max(d_n − s, 0)
      cost[n, s] = c · max(f − m, 0) + h(s + f − d_n)
      make[n, s] = f
  // Compute cost[1..n − 1, 0..D] and make[1..n − 1, 0..D].
  U = d_n
  for k = n − 1 downto 1
      U = U + d_k
      for s = 0 to D
          cost[k, s] = ∞
          for f = max(d_k − s, 0) to U − s
              val = cost[k + 1, s + f − d_k]
                    + c · max(f − m, 0) + h(s + f − d_k)
              if val < cost[k, s]
                  cost[k, s] = val
                  make[k, s] = f
  print cost[1,0]
  PRINT-PLAN(make, n, d)

PRINT-PLAN(make, n, d)
  s = 0
  for k = 1 to n
      print "For month " k " manufacture " make[k, s] " machines"
      s = s + make[k, s] − d_k
```

Explanation

We first initialize the base cases, which are the cases for month n starting with surplus s, for s=0,…,Ds=0,…,D. If dn>s, it suffices to manufacture dn−s machines, since we need not keep any surplus after month n. If dn≤s, we need not manufacture any machines at all.
We then calculate the total cost for month n as the sum of hiring extra labor c⋅max(f−m,0) and the inventory costs for leftover surplus h(s+f−dn), which can be nonzero if we had started out with a large surplus.
The outer **for** loop of the next block of code runs down from month n−1n−1 to 11, thus ensuring that when we consider month k, we have already solved the sub problems of month k+1k+1.
The next inner **for** loop iterates through all possible values of f as described.

Method Cycle

Solution

For every choice of f for a given month k, the total cost of (k,s) is given by the cost of extra labor (if any) plus the cost of inventory (if there is a surplus) plus the cost of the sub-problem (k+1,s+f−dk).

This value is checked and updated. Finally, the required answer is the answer to the sub-problem (1,0), which appears in cost[1,0].

That is, it is the cheapest way to satisfy all the demands of months 1,…,n when we start with a surplus of 0.

Problem Type

Minimization of cost

Requirements

- The company wishes to design a plan for the next n months.
- for each month, the company knows the demand, that is the number of machines that it will sell.
- the company keeps a full time staff who provide labor to manufacture up to the m machine per month.
- if a company needs to make more than m machines in a given month, it can hire additional part time labor.
- Further more, if, at the end of a month, the company is holding any unsold machines, it must pay inventory cost

Running Time

The running time O(nD2)
The space requirement is O(nD)

We can improve upon the space requirement by noting that we need only store the solution to sub-problems of the next month.
we can construct an algorithm that uses O(n+D) space

Designed by:

Zain Noman Farooq
(15B-058-SE)

Sheikh Ramzan Raza
(15-081-SE)