Knapsack Problems

Prerequisite modules

- Greedy
- Dynamic Programming
- Recursive Descent

Sample Problem: Tape Recording

Farmer John's favorite hobby is creating a tape containing some of Bessie's favorite music to listen to while she's being milked. The amount of milk that Bessie produces is dependent on the songs that Bessie listens to while being milked.

Given a collection of songs, each represented by a pair of integers, the length of the song (in seconds), the amount of milk Bessie produces while she's listening to that song, and the total amount of time that it takes to milk Bessie, find the set of songs such that their total length is no more than the time it takes to milk Bessie and they maximize Bessie's milk production.

The Abstraction

Given, A collection of objects, each which a size, a value (i.e., weight), and the total `space' available, find the set of objects which maximizes the sum of the value of the set, but whose sum of size is constrained by some limit. The total number/size of any particular item used in the set cannot exceed its availability.

Problem Viewpoint

The general way to view a knapsack problem is that of a bag of limited capacity, which is to be filled while maximizing the value of the objects in it.

For the problem above, the tape which Bessie will listen to while being milked is the ``knapsack,'' while the songs are the ``objects to be placed within the knapsack.''

Three Knapsack Problems

The knapsack problem style has three forms:

- Fractional knapsack problem
  A fractional knapsack problem is one in which you are allowed to place fractional objects in the knapsack. For example, if the objects were crude oil, airplane fuel, and kerosene and your knapsack a bucket, it might make sense to take 0.473 liter of the crude oil, 0.263 liter of the airplane fuel, and 0.264 liter of the kerosene. This is the easiest form of the knapsack problem to solve.
- Integer Knapsack problem
  In integer knapsack problems, only complete objects can be inserted into the knapsack. The example problem is of this form: partial songs aren't allowed.

- Multiple knapsack problem
  In the multiple knapsack problem, more than one knapsack is to be filled. If fractional objects are allowed, this is the same as having one large knapsack with capacity equal to the sum of all the available knapsacks, so this term will only be used to refer to the case of multiple integer knapsacks.

Fractional knapsack problem

The fractional knapsack problem is the easiest of the three to solve, as the greedy solution works:

- Find the object which has the highest ``value density'' (value of object / size).
- If the total amount of capacity remaining exceeds the availability of that object, put all of it in the knapsack and iterate.
- If the amount of capacity is less than the availability of the object, use as much as possible and terminate.
- This algorithm runs in *N log N* since it must sort the objects first based on value density and then put them into the knapsack in decreasing order until the capacity is used. It's normally easier to not sort them but rather just keep finding the highest value density not used each time, which gives a $O(N^2)$ algorithm.

Side note: For problems of this class, it's rare to have both size and availability, as you can do a trivial transformation to have all the objects of size 1, and the availability be the product of the original size and availability (dividing the value by the original size, of course).

Extensions: The value and availability of the objects can be real numbers without a problem in this case. The fractional size issue is also trivial to handle by this algorithm.

Integer knapsack problem

This is slightly more difficult, but is solvable using dynamic programming if the knapsack is small enough.

- Do dynamic programming on the maximum value that a knapsack of each size can have in it.
- Update this array for an object of size S by traversing the array in reverse order (of capacity), and seeing if placing the current object into the knapsack of size K yields a better set than the current best knapsack of size K+S.
- This algorithm runs in *K x N* time, where *K* is the size of the knapsack, and *N* is the sum of availability of objects.
- If the knapsack is too large to allocate this array, recursive descent is the method of choice, as this problem is NP-complete. Of course, recursive descent can run for a very long time in a large knapsack being filled with small objects.

Extensions:

- Fractional values are not a problem; the array just becomes an array of real numbers instead of integers. Fractional availability doesn't affect things, as you can, without loss of generality, truncate the number (if you have 3.5 objects, you can only use 3).
- Fractional size is a pain, as it makes the problem recursive descent.
- If the sizes are all the same, the problem can be solved greedily, picking the objects in decreasing value order until the knapsack is full.

- If the values are all 1.0, then again greedy works, selecting the objects in increasing size order until the knapsack is full.

Multiple knapsack problem

With multiple knapsacks of any size, the state space is too large to use the DP solution from the integer knapsack algorithm. Thus, recursive descent is the method ~~to solve this problem. Extensions:~~

- With recursive descent, extensions are generally easy. Fractional sizes and values are no problem, nor is another evaluation function.
- If the values are all one, then if the maximum number of objects that can be placed in all the knapsacks is n, then there is such a solution which uses the n smallest objects. This can greatly reduce the search time.

Sample Problems

Score Inflation [1998 USACO National Championship]

You are trying to design a contest which has the maximum number of points (<10,000). Given the length of the contest, a group of problems, the problem lengths, and the point value of each problem, find the contest which has the maximum number of points (which satisfies the length constraint).

Analysis: This is an integer knapsack problem, where the knapsack is the contest, the sizes are the length of problems, and the values are the point values. The limit on knapsack (contest) size is small enough that the DP solution will work in memory.

~~Fence Rails [1999 USACO Spring Open]~~

Farmer John is trying to construct a fence around his field. He has installed the posts already, so he knows the length requirement of the rails. The local lumber store has dropped off some boards (up to 50) of varying length. Given the length of the boards from the lumber store, and the lengths of rails that Farmer John needs, calculate the maximum numbers of rails that Farmer John can create.

Analysis: This is a multiple knapsack problem, where the knapsacks are the boards from the store, and the objects are the rails that Farmer John needs. The size of the objects are just the length, and the value is just one.

Since the values are all one, you know that if there is a solution using any K rails, there is a solution using the K smallest rails, which helps the recursive descent solver quite a bit.

Filling your Tank

You're in the middle of Beaver County, at the only city within 100 miles with a gas station, trying to fill up your tank so you can get to Rita Blanca. Fortunately, this town has a couple of gas stations, but they all seem to be almost out of gas. Given the price of gasoline at each station, and the amount of gas each one has, calculate how much gasoline to buy from each station in order to minimize the total cost.

Analysis: This is an fractional knapsack problem, where your knapsack is your gas tank, and the objects are gasoline.

USACO Gateway | Comment or Question