Greedy Solution to the

Fractional Knapsack Problem

There are n items in a store. For i =1,2, . . . , n, item i has weight $w_i > 0$ and worth $v_i > 0$. Thief can carry a maximum weight of W pounds in a knapsack. In this version of a problem the items can be broken into smaller piece, so the thief may decide to carry only a fraction x_i of object i, where $0 \leq x_i \leq 1$.
Item i contribute $x_i w_i$ to the total weight in the knapsack, and $x_i v_i$ to the value of the load.

In Symbol, the fraction knapsack problem can be stated as follows. maximize $^{n}\mathbf{S}_{i=1}\, x_i v_i$ subject to constraint $^{n}\mathbf{S}_{i=1}\, x_i w_i \leq W$

It is clear that an optimal solution must fill the knapsack exactly, for otherwise we could add a fraction of one of the remaining objects and increase the value of the load. Thus in an optimal solution $^{n}\mathbf{S}_{i=1}\, x_i w_i = W$.

Greedy-fractional-knapsack **(*w, v, W*)**

```
FOR i =1 to n
   do x[i] =0
weight = 0
while weight < W
   do i = best remaining item
      IF weight + w[i] ≤ W
         then x[i] = 1
            weight = weight + w[i]
         else
            x[i] = (w - weight) / w[i]
            weight = W
return x
```

Analysis

If the items are already sorted into decreasing order of v_i/w_i, then the while-loop takes a time in $O(n)$; Therefore, the total time including the sort is in $O(n \log n)$.

If we keep the items in heap with largest v_i/w_i at the root. Then

- creating the heap takes $O(n)$ time
- while-loop now takes $O(\log n)$ time
    - since heap property must be restored after the removal of root

Although this data structure does not alter the worst-case, it may be faster if only a small number of items are need to fill the knapsack.

One variant of the 0-1 knapsack problem is when order of items are sorted by increasing weight is the same as their order when sorted by decreasing value.

The optimal solution to this problem is to sort by the value of the item in decreasing order. Then pick up the most valuable item which also has a least weight. First, if its weight is less than the total weight that can be carried. Then deduct the total weight that can be carried by the weight of the item just pick. The second item to pick is the most valuable item among those remaining. Keep follow the same strategy until thief cannot carry more item (due to weight).

Proof

One way to proof the correctness of the above algorithm is to prove the greedy choice property and optimal substructure property. It consists of two steps. First, prove that there exists an optimal solution begins with the greedy choice given above. The second part prove that if A is an optimal solution to the original problem S, then $A - a$ is also an optimal solution to the problem $S - s$ where a is the item thief picked as in the greedy choice and $S - s$ is the subproblem after the first greedy choice has been made. The second part is easy to prove since the more valuable items

have less weight. Note that if $v` / w`$, is not it can replace any other because $w` < w$, but it increases the value because $v` > v$. □

Theorem *The fractional knapsack problem has the greedy-choice property*.

Proof Let the ratio $v`/w`$ is maximal. This supposition implies that $v`/w` \geq v/w$ for any pair (v, w), so $v`v / w > v$ for any ($v$, $w$). Now suppose a solution does not contain the full $w`$ weight of the best ratio. Then by replacing an amount of any other w with more $w`$ will improve the value. □