Lecture 14: Greedy Algorithms

CLRS section 16

Outline of this Lecture

We have already seen two general problem-solving techniques: divide-and-conquer and dynamic-programming . In this section we introduce a third basic technique: the greedy paradigm .

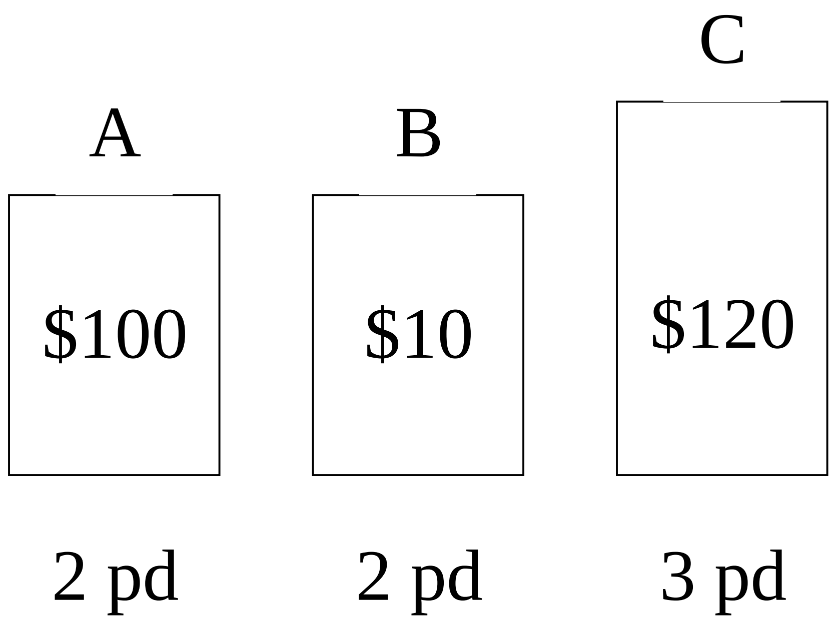
A greedy algorithm for an optimization problem **always makes the choice that looks best at the moment** and adds it to the current subsolution. Examples already seen are Dijkstra's shortest path algorithm and Prim/Kruskal's MST algorithms .

Greedy algorithms don't always yield optimal solutions but, when they do, they're usually the simplest and most efficient algorithms available.

The Knapsack Problem

We review the knapsack problem and see a greedy algorithm for the fractional knapsack. We also see that greedy doesn't work for the 0-1 knapsack (which must be solved using DP).

A thief enters a store and sees the following items:



His Knapsack holds 4 pounds. What should he steal to maximize profit?

1. Fractional Knapsack Problem : Thief can take a fraction of an item.

Solution = +

2 pounds of item A
2 pounds of item C

2 pds A $100	2 pds C $80

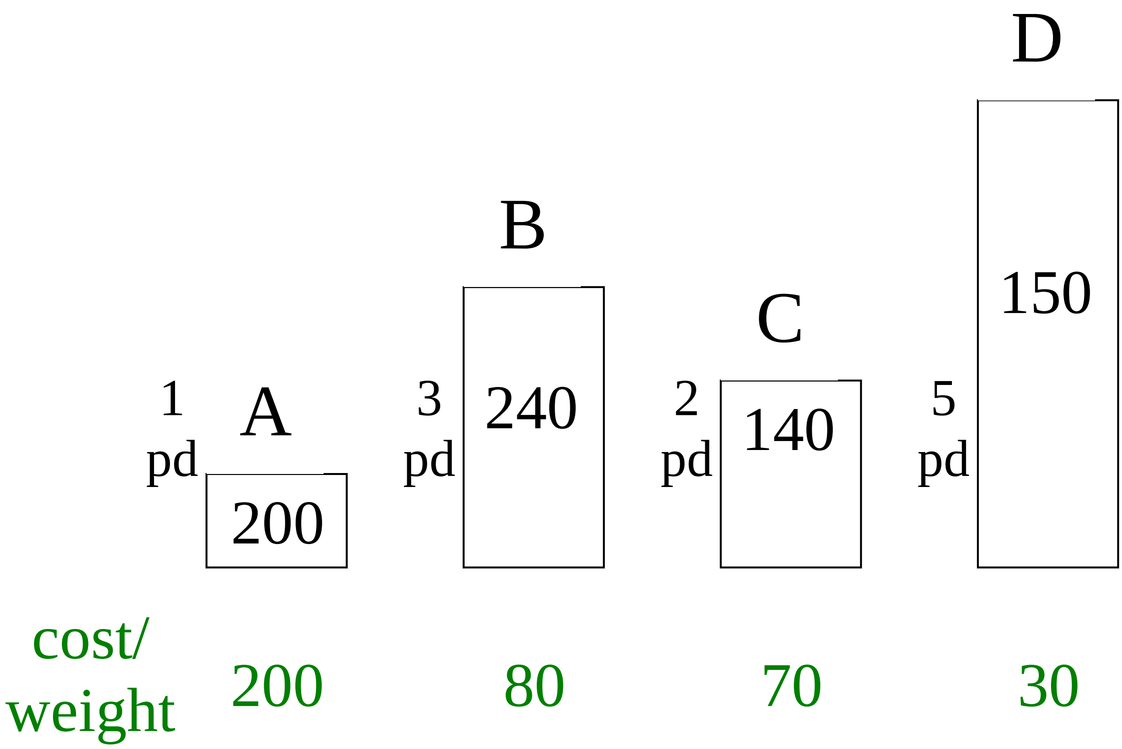
2. 0-1 Knapsack Problem : Thief can only take or leave item. He can't take a fraction.

Solution = 3 pounds of item C

3 pds C $120	

Greedy solution for Fractional Knapsack

Sort items by decreasing cost per pound



If knapsack holds $k = 5$ pds, solution is:

1	pds	A
3	pds	B
1	pds	C

Greedy solution for Fractional Knapsack

General Algorithm $O(nlogn)$.

Given a set of item I:

weight	w_1	w_2	$\ldots$	w_n
cost	c_1	c_2	$\ldots$	c_n

Let P be the problem of selecting items from I, with weight limit K, such that the resulting cost (value) is maximum.

1. Calculate $v_i = \frac{c_i}{w_i}$ for $i = 1, 2, \ldots, n$.

2. Sort the items by decreasing v_i. Let the sorted item sequence be $1, 2, \ldots, i, \ldots n$, and the corresponding v and weight be v_i and w_i respectively.

Greedy solution for Fractional Knapsack

3. Let k be the current weight limit (Initially, $k = K$). In each iteration, we choose item i from the head of the unselected list. If $k >= w_i$, we take item i, and $k = k - w_i$, then consider the next unselected item.

If $k < w_i$, we take *a fraction* f of item i, i.e., we only take $f = \frac{k}{w_i}$ (< 1) of item i, which weights exactly k. Then the algorithm is finished.

Observe that the algorithm may take a fraction of an item, which can only be the last selected item.

We claim that the total cost for this set of items is *an* optimal cost.

Correctness

Let $O = \{o_1, o_2, \ldots, o_j\} \subseteq I$ be the optimal solution of the problem P. Let the greedy solution $G = \{g_1, g_2, \ldots, g_k\} \subseteq I$, where the items are ordered according to the sequence of greedy choices.

The trick of the proof is to show there exist an optimal solution such that it also takes the greedy choice in each iteration. The *first step* is to show there exist an optimal solution such that it selects (a fraction or 1 unit of) item g_1, our first greedy choice. There are two cases to consider.

Suppose G takes 1 unit of g_1 (implies $K \geq w_{g_1}$). If O also takes 1 unit g_1, then we are done. Suppose O does not take 1 unit of g_1. Then we *take away* weight w_{g_1} from O and put 1 unit of g_1 to it, yielding a new solution O'. Observe O' has weight K (the weight constraint). Moreover, since g_1 has the maximum $\frac{c}{w}$, O' is as good as O (or else there is a contradiction). Hence $g_1 \in O'$ is also an optimal solution for P.

Correctness

On the other hand, suppose G takes a fraction f of g_1 (implies $K = f \times w_{g_1}$). If O also takes f unit g_1, then we are done. Suppose O takes *less than* f unit of g_1 (O can't take larger f unit of g_1). Then we *take away* weight $f \times w_{g_1}$ from O and put f unit of g_1 to it, yielding a new solution $O^{'}$. Similarly, $O^{'}$ is as good as O (or else there is a contradiction). Hence f unit of $g_1 \in O^{'}$ is also an optimal solution for P.

Observe that if we have shown an optimal solution $O^{'}$ selects a *fraction* of g_1, we are done with the proof.

Correctness

The *second* step is to show the current problem exhibits *optimal substructure* property. A problem exhibits optimal substructure if an optimal solution to the problem contains within it optimal solutions of subproblems.

In the fractional knapsack problem, we have shown there is an optimal solution $O^{'}$ that selects 1 unit of g_1. After we select g_1, the weight constraint decreases to $K^{''} = K - w_{g_1}$, the item set becomes $I^{''} = I - \{g_1\}$. Let $P^{''}$ be a fractional knapsack problem such that the weight constraint is $K^{''}$, and the item set is I''. Let $O^{''} = O^{'} - \{g_1\}$. To prove the optimal substructure property, we need to show $O^{''}$ is an optimal solution of P'' (an optimal solution to the problem contains within it optimal solution of subproblem).

Correctness

Suppose on the contrary that $O^{''}$ is *not* an optimal solution of P''. Let Q be an optimal solution of P'', which is more valuable that $O^{''}$. Let $R = Q \bigcup \{g_1\}$; observe that R is a feasible selection for P.

On the other hand, the value of $O^{'}$ (an optimal solution for P) equals the value of $O^{''}$ + g_1, which is *less* than the cost of R (since we assume the value of $O^{''} < Q$). Hence we find a selection R which is more valuable than the optimal solution $O^{'}$. A contradiction! Hence $O^{''}$ is an optimal solution for P''.

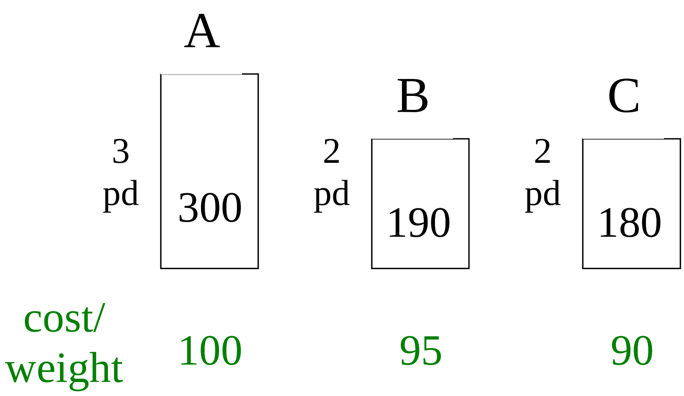
Therefore, after each greedy choice is made, we are left with an problem of the same form as the original problem. Since P'' needs to be solved optimally, we can show there exists an optimal solution for $P^{''}$ which selects g_2.

Inductively, we have shown the greedy solution is *an* optimal solution.

Greedy solution for 0-1 Knapsack Problem?

The 0-1 Knapsack Problem does not have a greedy solution!

Example:



$K = 4$ Solution is item B + item C

Question : Suppose we try to prove the greedy algorithm for 0-1 knapsack problem is correct. We follow exactly the same lines of arguments as fractional knapsack problem. Of course, it must fail. Where is the problem in the proof?