9

June 2, 2016

1 Greedy Algorithms

1.1 Job Scheduling Problem

Problem: given numbers $l_1 ... l_n > 0$, where l_i is length of job, w_i is weight of the job. Find an ordering to minimize total weighted completion time.

aka. minimize $w_1 l_1 + w_2(l_1 + l_2) + w_3(l_1 + l_2 + l_3) + ...$

Greedy algorithm: we sort by increasing l/w, and greedily take the "best" one.

Why is this optimal?

Proof of correctness: Assume we have an optimal ordering (w_i^*, l_i^*) such that the total weighted completion time is less than our ordering. Then there is an $1 \leq i \leq n$ such that $l_i^*/w_i^* > l_{i+1}^*/w_{i+1}^*$ (otherwise it's the same as our algorithm). Then if we swap i and $i+1$, then the difference between new and old is:

$(l_{i+1}^*)(w_{i+1}^*) + (l_i^* + l_{i+1}^*)(w_i^*) - (l_i^*)(w_i^*) - (l_i^* + l_{i+1}^*)(w_{i+1}^*) = (l_{i+1}^*)(w_i^*) - (l_i^*)(w_{i+1}^*) < 0$. This implies we can further improve the "optimal" solution. Aka. we can't have an optimal solution for which $l_i^*/w_i^* > l_{i+1}^*/w_{i+1}^*$. But all $n!$ permutations will have this property, except for the permutation generated by our greedy algorithm.

1.2 Fractional Knapsack Problem

Problem: given "values" $v_i > 0$, and "weights" $w_i > 0$, maximize $\sum v_i x_i$ such that $\sum w_i x_i \leq W$, $0 \leq x_i \leq 1$.

Greedy algorithm: Sort items by v_i/w_i (value-to-cost ratio), take as much as the one with biggest ratio each time.

Proof of correctness: suppose, in some iteration, $x_j \neq x_j^* \implies x_j^* < x_j$, where x^* is an optimal solution, x is our solution, x_j corresponds to the current biggest v_i/w_i.

Find another item k with $x_k^* > 0$ (if k does not exist, then all other "takings" are 0, and $v_j x_j > v_j x_j^*$). Now we increase x_j^* by δ/w_j and decrease x_k^* by δ/w_k for some small $\delta > 0$ (this is ok as it doesn't change our total weight). Then $\sum v_i x_i^* + \delta(v_j/w_j - v_k/w_k) > \sum v_i x_i^*$.