Lindsey Bieda and Joe Frambach
Greedy Algorithm Problems
9.11.2011

8. Consider the following problem. The input consists of n skiers with heights $p_1, \ldots, p_n$, and n skis with heights $s_1, \ldots, sn$. The problem is to assign each skier a ski to to minimize the average difference between the height of a skier and his/her assigned ski. That is, if the ith skier is given the $a(i)$th ski, then you want to minimize:

$$\frac{1}{n}\sum_{i=1}^{n} \mid p_i - s_{a(i)} \mid$$

(a) Consider the following greedy algorithm. Find the skier and ski whose height difference is minimized. Assign this skier this ski. Repeat the process until every skier has a ski. Prove of disprove that this algorithm is correct.

This algorithm does not correctly solve the problem. Consider the following inputs:
skiers = (0.3, 0.3, 0.5, 0.7, 0.7)
ski heights = (0.4, 0.4, 0.4, 0.59, 0.6)

$GRE = 0.09 + 0.1 + 0.1 + 0.1 + 0.3 = 0.59/5 = 0.118$
$OPT = 0.1 + 0.1 + 0.1 + 0.11 + 0.1 = 0.51/5 = 0.102$

(b) Consider the following greedy algorithm. Give the shortest skier the shortest ski, give the second shortest skier the second shortest ski, give the third shortest skier the third shortest ski, etc. Prove of disprove that this algorithm is correct.

Prove that algorithm 8b is correct.

Proof: $\exists$ input $I \ni$ algorithm 8b will produce incorrect output.

$GRE(I) = \{(G_{p_1}, G_{s_1}), \ldots, (G_{p_n}, G_{s_n})\}$
$OPT(I) = \{(O_{p_1}, O_{s_1}), \ldots, (O_{p_n}, O_{s_n})\}$, where $OPT(I)$ agrees with $GRE(I)$ for the most steps and is sorted by increasing O_{p_n}, k
Let k be the first point of disagreement between algorithm 8b's output and the optimal output.
We generate OPT' by swapping (if necessary) O_{p_k} with $O_{p_{k+i}}$, where $O_{p_{k+i}} = G_{p_k}$ and swapping O_{s_k} with $O_{s_{k+i}}$, where $O_{s_{k+i}} = G_{s_k}$. Since, greedy selects skiers and skis from smallest to the largest we know that $G_{p_k} \leq O_{p_k}$ and $G_{s_k} \leq O_{s_k}$, and since OPT is sorted in increasing order for it to disagree with GRE it must not be optimal $\perp$.

12. We consider the following scheduling problem:

INPUT: A collection of jobs $J_1, \ldots, J_n$, where the ith job is a tuple (r_i, x_i) of non-negative integers specifying the release time and size of the job.
OUTPUT: A preemptive feasible schedule for these jobs on one processor that minimizes the total completion time $\sum_{i=1}^{n} C_i$.

A *schedule* specifies for each unit time interval, the unique job that that is run during that time interval. In a *feasible* schedule, every job J_i has to be run for exactly x_i time units after time r_i . The *completion* time C_i for job J_i is the earliest time when J_i has been run for x_i time units. Examples of these basic definitions can be found below.

We consider two greedy algorithms for solving this problem that schedule times in an online fashion, that is the algorithms are of the following form:

$t = 0$
while there are jobs left not completely scheduled
    Among those jobs J_i such that $r_i \leq t$, and that have previously been scheduled for less
        than x_i time units, pick a job J_m to schedule at time t according to some rule;
    increment t

One can get different greedy algorithms depending on the rule for selecting J_m. For each of the following greedy algorithms, prove or disprove that the algorithm is correct. Proofs of correctness must use an exchange argument.

SJF: Pick J_m to be the job with minimal size x_i. Ties may be broken arbitrarily.

SRPT: Let $y_{i,t}$ be the total time that job J_i has been run before time t. Pick J_m to be a job that has minimal remaining processing time, that it, that has minimal $x_i - y_{i,t}$. Ties may be broken arbitrarily.

As an example of SJF and SRPT consider the following instance: $J_1 = (0, 100)$, $J_2 = (10, 10)$ and $J_3 = (1, 4)$. Both SJF and SRPT schedule job J_1 between time 0 and time 1, and job J_3 between time 1 and time 5, when job J_3 completes, and job J_1 again between time 5 and time 10. At time 10, SJF schedules job J_2 because its original size 10 is less than job J_1's original size 100. At time 10, SRPT schedules job J_2 because its remaining processing time 10 is less than job J_1's remaining processing time 94. Both SJF and SRPT schedule job J_2 between time 10 and 20, when J_2 completes, and then job J_1 from time 20 until time 114, which job J_1 completes. Thus for both SJF and SRPT on this instance $C_1 = 114$, $C_2 = 20$ and $C_3 = 5$ and thus both SJF and SRPT have total completion time 139.

Consider the following jobs: $J = [(0, 2), (0, 3), (4, 2)]$.
SJF: J_1 runs from time 0 to 2, completing in $C_1 = 2$. J_2 runs from time 2 to 4. J_3 runs from time 4 to 6, completing in $C_3 = 6$. J_2 runs from time 6 to 7, completing in $C_2 = 7$. Total completion time for SJF is 15.
SRPT: J_1 runs from time 0 to 2, completing in $C_1 = 2$. J_2 runs from time 2 to 5, completing in $C_2 = 5$. J_3 runs from time 5 to 7, completing in $C_3 = 7$. Total completion time is 14.
Therefore, SJF is not optimal since SJF $<$ SRPT.

Thm: SRPT(I) is correct.
Proof: Assume $\exists$ input $I \ni$ SRPT(I) is incorrect.

SRPT(I) returns an ordered list of processes to run at each time interval.
SRPT(I) $= [J_{s(0)}, J_{s(1)}, ..., J_{s(n)}]$, where s(i) represents which job is to run at time i.
OPT(I) $= [J_{o(0)}, J_{o(1)}, ..., J_{o(n)}]$, where o(i) represents which job is to run at time i, and OPT matches SRPT for the most number of steps.
Let k be the first step which OPT(I) disagrees with SRPT(I); OPT(I)[j] == SRPT(I)[j] $\forall j < k$.

Let job $J_{s(k)}$ have completion time $C_{SRPT}(k)$ and let job $J_{o(k)}$ have completion time $C_{OPT}(k)$.
Let job $J_{s(k)}$ have remaining time $R_{SRPT}(k)$ and let job $J_{o(k)}$ have remaining time $R_{OPT}(k)$.
Since SRPT chose $J_{s(k)}$ rather than $J_{o(k)}$, $R_{SRPT}(k) < R_{OPT}(k)$.

Construct OPT' by modifying OPT, allocating the next $R_{SRPT}(k)$ time to Job $J_{s(k)}$, followed by $R_{OPT}(k)$

time to Job $J_{o(k)}$, effectively swapping the positions of the jobs in OPT.

If $C_{SRPT}(k) == C_{OPT}(k)$:
Then OPT can be considered equal to SRPT for step k.

If $C_{SRPT}(k) > C_{OPT}(k)$:
Then Job $J_{s(k)}$ will be able to finish earlier than $J_{o(k)}$ originally did, and Job $J_{o(k)}$ will finish at the same time as $J_{s(k)}$ originally did, since both jobs' cumulative finish times were unchanged by swapping.
Therefore, in this case, $OPT' > OPT$.

If $C_{SRPT}(k) < C_{OPT}(k)$:
Then Job $J_{o(k)}$ will be able to finish earlier than $J_{s(k)}$ originally did, and Job $J_{s(k)}$ will finish at the same time as $J_{o(k)}$ originally did, since both jobs' cumulative finish times were unchanged by swapping.
Therefore, in this case, $OPT' > OPT$.

In all cases, $OPT' > OPT$. $\perp$.