CPSC 320 2024W1: Full Proofs of Greedy Worked Examples

September 27, 2024

In class on September 27, we considered sketches of a proof of optimality for two greedy algorithms, using the greedy stays ahead and exchange argument techniques. We omitted some details in class using sparkle emojis[1] for the sake of clarity: we wanted to focus on the big-picture technique of the proof, without getting bogged down in details of induction or algebra.

In this document, I add the gory details. Note that I intend this document to be an appendix/supplement to the lecture materials, and I do not consider the details of the induction/algebra to be critical to your mastery of the learning goals for this unit (if I thought they were, I would have gone over them in lecture). However, these details may be helpful in reinforcing your understanding of proofs in general, and in helping you formulate your answers for assignment/exam questions.

Greedy Stays Ahead

To illustrate the greedy stays ahead technique, we considered the following problem (from take-home test 2 in the 2023W2 offering):

> On a busy day of the final exam period, several computer science exams are being written in different rooms of OSBO. You have n bundles of exams that some of the invigilating TAs will need to bring back to ICICS for scanning, and each bundle has b_i exam booklets. However, there are a few rules you need to follow:
>
> - Each TA can carry no more than m exams.
> - The exams are expected at the CS building in a particular order, so they need to be transported back in the order listed (bundle 1, bundle 2, ..., bundle n).
> - You can't split any of the exam bundles among multiple TAs, since some exams may end up getting misplaced. (You can assume that no bundle contains more than m exams.)
>
> You want to distribute the exams to TAs such that you transport the exams using as few TAs as possible. For example, suppose you have bundles of size $[40, 40, 150, 25, 100]$ and $m = 200$, then an optimal distribution is $[40, 40], [150, 25], [100]$ – that is, giving bundles 1 and 2 to the first TA, bundles 3 and 4 to a second TA, and bundle 5 to a third TA. (For this instance, the optimal distribution is not unique: other optimal distributions are $[40], [40, 150], [25, 100]$ and $[40, 40], [150], [25, 100]$.)

Greedy algorithm

The optimal greedy solution is to give each TA as many exam booklets as they can carry before giving any booklets to the next TA. Stated more formally: if the sum of all bundles is less than or equal to m, give

[1] No, we will not accept this kind of reasoning on an assignment or exam. Sorry!

all booklets to a single TA. Otherwise, let j be the smallest value such that the first j bundles have a sum greater than m; give $j-1$ bundles to the first TA and recurse on the remaining bundles.

Proof that the greedy algorithm is optimal

We use the following **greedy stays ahead lemma**: The first i TAs in the greedy solution will be carrying at least as many total bundles as the first i TAs in the optimal solution.

We will prove the lemma, and then use this to prove that the greedy algorithm produces an optimal solution.

Proof of lemma: we prove the lemma by induction on i. For the base case $i = 1$, we know the greedy algorithm gives as many bundles to the first TA as possible without exceeding m bundles; this therefore proves the lemma for $i = 1$, because if the optimal solution assigned more bundles to the first TA, that TA would have more than m bundles (which is impossible, because then the solution would be invalid and therefore not optimal).

Now assume the first $i-1$ TAs in the greedy solution are carrying at least as many exam bundles as the first $i-1$ TAs in the optimal solution. Suppose the total number of exam bundles given to the first $i-1$ TAs in the greedy solution is S_{i-1}, and TA i is carrying bundles $[b_{i1}, \ldots, b_{ij}]$. By the inductive hypothesis, we know that the first $i-1$ TAs in the optimal solution are carrying no more than the first S_{i-1} exam bundles in total. Therefore, the first set of bundles assigned to TA i in the optimal solution is no later than b_{i1} (the first bundle assigned to TA i in the greedy algorithm). This means that we cannot assign any bundle beyond b_{ij} to TA i in the optimal solution, because that would necessarily mean TA i has more than m exams (if it were possible to take more bundles beyond b_{ij} when starting with bundle b_{i1}, our greedy algorithm would have given more bundles to TA i). This concludes the proof of the lemma. □

Now that we've proven the lemma, we can conclude the proof that our greedy algorithm uses no more TAs than the optimal solution. Suppose to the contrary that our greedy solution uses k TAs to transport all the exams and an optimal solution uses $j < k$. The first j TAs in the greedy solution are transporting some number of exams less than $\sum_{i=1}^{n} b_i$ – that is, they are transporting some number of exams that is less than the total number to transport (otherwise, the greedy algorithm would have stopped assigning exam booklets after TA j). However, by our stays-ahead lemma, the first j TAs in the optimal solution cannot be transporting more exams than the first j TAs in the greedy solution – which means that the first j TAs in optimal cannot be transporting all $\sum_{i=1}^{n} b_i$ exams. Therefore, the greedy solution uses no more TAs than the optimal solution, which concludes the proof that the greedy solution is in fact optimal.

Exchange Argument

As an example of an exchange argument, we considered the following problem:

> You are an air conditioner mechanic, and your normally temperate coastal city is expecting record-breaking heat this summer. You have agreed to install or repair an air conditioning system for n clients, and each job will take one day to complete. You must decide the order in which to complete the jobs.
> None of your clients want to be hot and uncomfortable in the coming heat wave, so they would all prefer to have their project completed earlier rather than later. In particular, after n days the uncomfortable heat is expected to be over, so nobody is going to be especially happy with being the last client on your list. More precisely, if you complete the job for client i at the end of day j, then client i's satisfaction is $s_i = n - j$.
> Some clients are more important to you than others (perhaps they are paying you more, are more likely to recommend you to others, or are more likely to be repeat customers in the future). You assign each client i a weight w_i. You want to schedule your clients across the n days such that you maximize the weighed sum of satisfaction, that is, $\sum_{1 \le i \le n} w_i s_i$.

Greedy Algorithm

The optimal greedy strategy is to complete the projects in decreasing order of w_i.

Proof that the greedy algorithm is optimal

Let $\mathcal{O}$ denote an optimal solution, with ordered weights given by $o_1, o_2, \ldots, o_n$. The weighted sum of satisfaction for $\mathcal{O}$ is $S(\mathcal{O}) = \sum_{i=1}^{n}(n-i)o_i$.

Let p, q, where $p < q$, denote indices in $\mathcal{O}$ that define an inversion, relative to the order of the greedy ordering $\mathcal{G}$. That is, this is a pair of jobs that appear in a different order than they do in $\mathcal{G}$. Because the greedy strategy is to order jobs by decreasing weight, we must have $o_p \leq o_q$. The weighted sum of satisfaction for $\mathcal{O}$ is

$$S(\mathcal{O}) = (n-1)o_1 + (n-2)o_2 + \ldots + (n-p)o_p + \ldots + (n-q)o_q + \ldots + (n-n)o_n.$$

Let $\mathcal{O}'$ denote the solution obtained after we swap o_p with o_q in $\mathcal{O}$ (i.e., we swap the order so these two elements appear in the same order as they do in $\mathcal{G}$). Then

$$\begin{aligned} S(\mathcal{O}') &= (n-1)o_1 + (n-2)o_2 + \ldots + (n-p)o_q + \ldots + (n-q)o_p + \ldots + (n-n)o_n \\ &= S(\mathcal{O}) + (n-p)(o_q - o_p) + (n-q)(o_p - o_q) \\ &= S(\mathcal{O}) + (q-p)(o_q - o_p) \geq S(\mathcal{O}), \end{aligned}$$

because $q > p$ and $o_q \geq o_p$. Therefore, the weighted satisfaction of $\mathcal{O}'$ is at least as high as the weighted satisfaction of $\mathcal{O}$.

We can proceed in this way, repeatedly swapping inversions between the optimal schedule and the greedy schedule, until the optimal schedule is transformed into the greedy schedule. The weighted satisfaction will not decrease with each swap. Therefore, $S(\mathcal{G}) \geq S(\mathcal{O})$ and the greedy schedule is in fact optimal.