csci-5454-hw8

Jake Krol

October 2024

1 Question 1

1.1 Question 1a

The greedy algorithm, A, described in Q1a can fail to return the minimum subset of classrooms to fill the study time. For example, let $T = 1$, and let the interval set of studyroom availability be $S = \{[0, \frac{1}{2}], [\frac{1}{6}, \frac{5}{6}], [\frac{1}{2}, 1]\}$. The optimal solution, OPT, is $\{[0, \frac{1}{2}], [\frac{1}{2}, 1]\}$ where $|OPT| = 2$. However, the greedy algorithm does not find this solution since it selects the max size interval first, $[\frac{1}{6}, \frac{5}{6}]$. Therefore, both $[0, \frac{1}{2}]$ and $[\frac{1}{2}, 1]$ must be added to reach T total hours of booked studyroom time for a total of $|A| = 3$ rooms.

1.2 Question 1b

The algorithm from Q1a, can be corrected by performing a two level sort on the studyrooms interval set. First, sort ascending studyroom intervals by start time (s), then sort descending by finish time (f) for any start time ties. Then, loop over the array greedily choosing the studyroom with the longest availability, yet only consider rooms with finish time greater than the latest booking.

Algorithm 1 GREEDY-BOOKING(s, f, n)

Requires: Sort s ascending and break ties with f descending
$A = \{a_1\}$ ▷ room set. element subscripts index start-finish intervals
$k = 1$
for $m = 2$ upto n **do**
  if $f[m] \geq f[k]$ **then** ▷ test if finish time occurs after latest booking
    $A = A \cup a_k$ ▷ book room
  end if
end for
return A

1.3 Question 1c

Since the start times are sorted ascending with ties broken by a descending sort of finish times, the first room added is the earliest available room with the longest availability, a_1. Let, t_i denote the time interval covered by studyroom i. a_1 has an interval $[0, t_1]$. If $t_1 = T$, then the min subset is a_1. The remaining subproblem is choosing the optimal $a_i : f[i] > a_1$ and $s[i] \geq s[1]$ to fill the remaining interval $[t_1, T]$. The next greedy choice also creates a subproblem with the same structure: $[t_1 + t_i, T]$. This continues until $t_1 + \ldots \geq T$. Since the problem demonstrates optimal substructure, consider if the a non-greedy choice, $a_j : s[j] = s[1] \wedge f[j] < f[1]$ was made for the initial problem $[0, t_1]$. Since $|a_j| < |a_1|$, this means another room must be booked, a_k, and the solution is $|\{a_j, a_k\}| > OPT = |a_1|$.

2 Question 2

2.1 Question 2a

The algorithm A is not a $\frac{1}{c}$ approximation of the optimal solution since it can be shown that the approximation gets worse as a function of the input size. Since A selects only the item with the best value, the amount of value in the remaining items is unbounded.

Let the item values be $[v_1 = n, v_2 = n-1, v_3 = n-1, \ldots, v_n = n-1]$ with weights $[w_1 = n, w_2 = \frac{1}{n}, w_3 = \frac{1}{n}, \ldots, w_n = \frac{1}{n}]$ and the total weight capacity, $W = n$.

The greedy algorithm chooses the max value item, $v_1 = n$, with weight $w_1 = n$. Meanwhile, the optimal solution chooses all items except item 1: $2 \ldots n$. The value of the optimal solution is $(n-1) \cdot (n-1) = n^2 - 2n + 1$, which is asymptotic to n^2 for large n. Therefore, the approximation ratio between greedy and optimal solutions is $\frac{n}{n^2} = \frac{1}{n}$. The approximation of the greedy solution is not constant and can grow as n increases.

2.2 Question 2b

The greedy algorithm, A', is not a $\frac{1}{c}$ approximation of the optimal solution since it can be shown that the approximation is not within a constant factor of the optimal solution.

Let there be two items with value-weight ratios of $\frac{v_1}{w_1} = \frac{2}{1}$ and $\frac{v_2}{w_2} = \frac{10^9}{10^9}$, and let the total weight capacity be $W = 10^9$. A' first selects $\frac{v_1}{w_1}$ which has the best value-weight ratio and there is no space for item 2 since $10^9 + 1 > W$. The optimal solution is to take item 2 instead.

Therefore, the ratio between greedy and optimal solutions is $\frac{1}{10^9} = \frac{1}{W}$. The approximation is not a constant since this fraction can grow as a function of the weight capacity, W.

2.3 Question 2c

$A'' = max(A, A')$, and we are prompted to show that A'' is a $\frac{1}{2}$ approximation of the optimal solution: $\frac{1}{2} \leq \frac{A''}{OPT} \leq 1$. If the solution is a single max value item which maximizes the weight capacity, then A will capture this as the optimal solution and A'' will also be optimal. However, if a solution includes more than 1 item, then $A' = \sum_{i=1}^{k} v_i \leq OPT_f = (\sum_{i=1}^{k} v_i) + \alpha_{k+1} \cdot v_{k+1}$. The worst case includes only two items $v_k > v_{k+1}$.

$$A' \leq OPT \leq OPT_f = v_k + \alpha_{k+1} \cdot v_{k+1} < (2 \cdot v_k) = 2 \cdot A'$$

Also, $OPT_f \leq A + A'$ since adding A (the most valuable item) to A' is always better than adding a fractional item to A', which is OPT_f.

And, A'' is strictly greater than A and A', so it is also greater than $A + A'$.

$$\begin{aligned} max(A, A') &\geq A' \\ max(A, A') &\geq A \\ max(A, A') + max(A, A') = 2max(A, A') &\geq A + A' \end{aligned}$$

Altogether, the optimal solution is less than the fractional knapsack optimal solution which is less than the sum of A and A' which is less than twice A'' showing that A'' is no worse than a $\frac{1}{2}$ approximation of OPT.

$$OPT \leq OPT_f \leq A + A' \leq 2max(A, A') = 2A''$$