Homework 15

Samuel Pegg

December 2020

1 Page 428 16.2-5

Describe an efficient algorithm that, given a set $x_1, \ldots, x_n$ of points on the real line, determines the smallest set of unit-length closed intervals that contains all of the given points. Argue that your algorithm is correct.

Sort and relabel $x_1, \ldots, x_n$ to obtain

$$x_{i_1} \leq x_{i_2} \leq \cdots \leq x_{i_n}$$

and start with the interval $I_{i_1} = [x_{i_1}, x_{i_1} + 1]$. Let S be the solution set asked for in the problem statement. Initialise $S = \emptyset$, the empty set. Then for all $j \in \{1, \ldots, n\}$, if $x_{i_j} \in S$ already then continue to the next j. Otherwise, update $S = S \cup I_{i_j}$.

The complexity is $O(n \log n)$ for the sorting part of the algorithm and $O(n)$ to find S. Hence the complexity of this algorithm is $O(n \log n)$.

1.1 Greedy Choice

Suppose S is an optimal solution with left-most unit interval $[x, x + 1]$. Clearly, we must have $x \leq x_{i_1}$. So clearly all the points contained in $[x, x+1]$ must be in the reduced interval $[x_{i_1}, x+1]$. Hence they are also contained in $[x_{i_1}, x_{i_1} + 1]$. So we could replace $[x, x+1]$ in S with $[x_{i_1}, x_{i_1} + 1]$ and still get an optimal solution. Thus we can find an optimal solution by removing all the points in $[x_{i_1}, x_{i_1} + 1]$ and solving the resulting sub-problem. Let S' denote the solution to the sub-problem. Then $[x_{i_1}, x_{i_1} + 1] \cup S'$ is an optimal solution to the full problem.

1.2 Optimal Substructure

Let S be the optimal solution and consider $S_s = S - [x_{i_1}, x_{i_1} + 1]$. Suppose $S' \neq S_s$ is an optimal solution to the sub-problem. Then

$$|S'| < |S_s| \implies |S' \cup [x_{i_1}, x_{i_1} + 1]| < |S_s \cup [x_{i_1}, x_{i_1} + 1]| = |S|$$

so $S' \cup [x_{i_1}, x_{i_1} + 1]$ is a solution to the full problem and is better than S. This contradicts the optimality of S, hence our assumption that $S' \neq S_s$ must be incorrect, and hence S_s is the optimal solution to the sub-problem.

2 Page 447 16-2 (a)

Suppose you are given a set $S = a_1, \ldots, a_n$ of tasks, where task a_i requires p_i units of processing time to complete once it has started. You have one computer on which to run these tasks, and the computer can run only one task at a time. Let c_i be the completion time of task a_i, that is, the time at which task a_i completes processing. Your goal is to minimise the average completion time, that is, to minimise $\frac{1}{n}\sum_{i=1}^{n} c_i$. For example, suppose there are two tasks, a_1 and a_2, with $p_1 = 3$ and $p_2 = 5$, and consider the schedule in which a_2 runs first, followed by a_1. Then $c_2 = 5$, $c_1 = 8$, and the average completion time is $\frac{5+8}{2} = 6.5$. If task a_1 runs first, however, then $c_1 = 3$, $c_2 = 8$, and the average completion time is $\frac{3+8}{2} = 5.5$.

Give an algorithm that schedules the tasks so as to minimise the average completion time. Each task must run non-preemptively, that is, once task a_i starts, it must run continuously for p_i units of time. Prove that your algorithm minimises the average completion time, and state the running time of your algorithm.

Sort the tasks by processing time, i.e. we want an ordering $a_{i_1}, \ldots, a_{i_n}$ such that

$$p_{i_1} \leq \cdots \leq p_{i_n}$$

and then perform the tasks in this order. This algorithm is essentially sorting, so the running time is $n \log n$.

2.1 Greedy Choice

The greedy choice is to choose the task with the smallest processing time. Let O be an optimal solution and let t be the first task to run O. Since a_{i_1} is the smallest task, the processing time of t, p_t, satisfies $p_t \leq p_{i_1}$. If $t = a_{i_1}$ then we are done. If $t \neq a_{i_1}$, since t is first in O, we can swap the order of t and a_{i_1} to get a new solution O'. This amounts to reducing the completion times of a_{i-1} and the completion times of all tasks in O' between t and a_{i_1} by the difference in processing times of t and a_{i_1}. Since all other completion times remain the same, the average completion time of O' is less than or equal to the average completion time of O, hence the greedy choice gives an optimal solution.

2.2 Optimal Substructure

Let O be an optimal solution and let a_k be in the optimal solution. Consider the sub-problem $O - \{a_k\}$ and suppose it is not the optimal solution to the sub-problem over $\{a_1, \ldots, a_n\} - \{a_k\}$. Let O' be the optimal solution to the sub problem over $\{a_1, \ldots, a_n\} - \{a_k\}$. Then $O' \cup \{a_k\}$ has a lower cost than O which contradicts the optimality of O. Hence the algorithm above exhibits optimal substructure by the cut and paste method.