

	Task						
a_i	1	2	3	4	5	6	7
d_i	4	2	4	3	1	4	6
w_i	70	60	50	40	30	20	10

Figure 16.7 An instance of the problem of scheduling unit-time tasks with deadlines and penalties for a single processor.

unit-time tasks with deadlines and penalties for a single processor. The running time is $O(n^2)$ using GREEDY, since each of the $O(n)$ independence checks made by that algorithm takes time $O(n)$ (see Exercise 16.5-2). Problem 16-4 gives a faster implementation.

Figure 16.7 demonstrates an example of the problem of scheduling unit-time tasks with deadlines and penalties for a single processor. In this example, the greedy algorithm selects, in order, tasks a_1, a_2, a_3 , and a_4 , then rejects a_5 (because $N_4(\{a_1, a_2, a_3, a_4, a_5\}) = 5$) and a_6 (because $N_4(\{a_1, a_2, a_3, a_4, a_6\}) = 5$), and finally accepts a_7 . The final optimal schedule is

$\langle a_2, a_4, a_1, a_3, a_7, a_5, a_6 \rangle$,

which has a total penalty incurred of $w_5 + w_6 = 50$.

Exercises

16.5-1

Solve the instance of the scheduling problem given in Figure 16.7, but with each penalty w_i replaced by $80 - w_i$.

16.5-2

Show how to use property 2 of Lemma 16.12 to determine in time $O(|A|)$ whether or not a given set A of tasks is independent.

Problems

16-1 Coin changing

Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer.

- Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.

- b. Suppose that the available coins are in the denominations that are powers of c , i.e., the denominations are c^0, c^1, \dots, c^k for some integers $c > 1$ and $k \geq 1$. Show that the greedy algorithm always yields an optimal solution.
- c. Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. Your set should include a penny so that there is a solution for every value of n .
- d. Give an $O(nk)$ -time algorithm that makes change for any set of k different coin denominations, assuming that one of the coins is a penny.

16-2 Scheduling to minimize average completion time

Suppose you are given a set $S = \{a_1, a_2, \dots, a_n\}$ 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 minimize the average completion time, that is, to minimize $(1/n) \sum_{i=1}^n c_i$. 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 $(5 + 8)/2 = 6.5$. If task a_1 runs first, however, then $c_1 = 3$, $c_2 = 8$, and the average completion time is $(3 + 8)/2 = 5.5$.

- a. Give an algorithm that schedules the tasks so as to minimize 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 minimizes the average completion time, and state the running time of your algorithm.
- b. Suppose now that the tasks are not all available at once. That is, each task cannot start until its **release time** r_i . Suppose also that we allow **preemption**, so that a task can be suspended and restarted at a later time. For example, a task a_i with processing time $p_i = 6$ and release time $r_i = 1$ might start running at time 1 and be preempted at time 4. It might then resume at time 10 but be preempted at time 11, and it might finally resume at time 13 and complete at time 15. Task a_i has run for a total of 6 time units, but its running time has been divided into three pieces. In this scenario, a_i 's completion time is 15. Give an algorithm that schedules the tasks so as to minimize the average completion time in this new scenario. Prove that your algorithm minimizes the average completion time, and state the running time of your algorithm.