Solutions to Quiz 2

Solution 1:

(a) True: Latest start time is optimal. This is the same as earliest finish time, but run in reverse-time order.

(b) (iii) - Interval partitioning is equivalent to coloring an interval graph. The vertices of this graph are the intervals, and two vertices are adjacent if the intervals overlap. Coloring the graph is equivalent to partitioning the intervals into non-overlapping subsets.

(c) All are true except (iv). Claims (i) and (ii) are trivially true. Claim (iii) is true by the corollary to Claim 2. To see why Claim (iv) fails, consider a set of three points at the vertices of an equilateral triangle. After adding any two of these as center, the third point is the same distance from the centers as they are from each other.

(d) $\ln a$: The greedy set-cover's approximation ratio is $\ln |X|$, which in this case is $\ln a$.

Solution 2:

(a) Sort the bottles in increasing order c_i/p_i. Another way to think about this is that every time we put a pill in a bottle, we are paying a fraction of $1/p_i$ of the bottle's overall cost of c_i, and therefore a single pill of bottle i costs us an incremental cost of c_i/p_i. A good strategy is to select bottles so that this incremental "cost per pill" is minimized.

(b) We will show partial optimality, by showing that the first bottle that greedy selects is in some optimal solution, even if it only has one pill. (The general proof is a straightforward extension, based on determining where the optimal first differs from greedy.) It will simplify the proof to assume that all the c_i/p_i ratios are distinct.

Let us assume the bottles are sorted so that c_1/p_1 is the smallest cost-per-pill ratio. Consider any optimal solution O, and suppose towards a contradiction that it does not use bottle 1. Take any other bottle $j \neq 1$ that is in O, and remove one pill from bottle j and put it into bottle 1. Clearly, this is feasible, since bottle 1 was not used in O, so this bottle has room for this pill.

How has the overall cost been affected? Since the pill has been moved from bottle j to bottle 1, the incremental cost for bottle j has decreased by c_j/p_j, and the incremental cost of bottle 1 has increased by c_1/p_1. Thus, the net change in the total cost is

$$\frac{c_1}{p_1} - \frac{c_j}{p_j}.$$

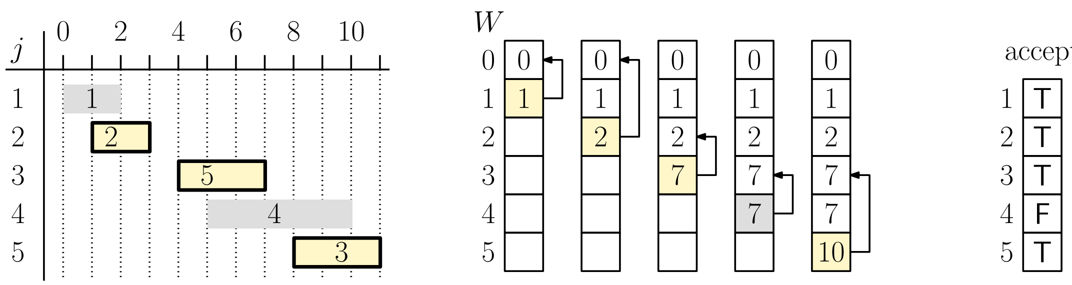
But, since c_1/p_1 has the smallest ratio, this quantity is negative, implying that this change decreases the cost. However, this implies that O is not optimal, a contradiction.

A note about grading: A number of people lost points because their justification was too vague. A typical incorrect justification could be briefly summarized as, "Since c_i/p_i is the cost per pill, it is clearly best to minimize this quantity." While this is a correct assertion, it alone is not a convincing argument. For example, if we altered the cost model so that each bottle that is used must be paid for in full, the greedy algorithm would *not* be optimal. But this argument does not clearly distinguish why greedy works in one case but not in the other. We were looking for an exchange-style argument, like those appearing in lectures and in the homework solutions.

Solution 3: See Fig. 1. (For completeness, we also show the accept array). For example, when computing $W[5]$, we have a choice between

$$\begin{array}{ll} W[4] = 7 & \text{(don't take) or} \\ v[5] + W[\text{prior}[5]] = v[5] + W[3] = 3 + 7 = 10 & \text{(take).} \end{array}$$

The latter is larger so we take the request, setting $W[5] \leftarrow 10$ and accept$[5] = \mathsf{T}$. The final total value is $W[5] = 10$, consisting of the intervals $\{2, 3, 5\}$.



(a) (b) (c)

Figure 1: Weighted interval scheduling.

Solution 4: Define $W(j)$ to be the maximum value, including penalties to schedule the first j intervals. The basis is $W(0) = 0$. Otherwise, for $1 \leq j \leq n$, we have two possibilities, depending on whether we accept or reject the interval. To avoid subscripting out of bounds, let us define $f_0 = 0$.

Interval j is not accepted: If j is not included in the schedule, then we should do the best we can with the remaining $j-1$ requests, which ends at time f_{j-1}. This leaves a gap size $f_j - f_{j-1}$, which is included in the penalty (see Fig. 2(a)). Thus, we have $W(j) = W(j-1) - (f_j - f_{j-1})$.

It is important to account for this penalty correctly. Note, for example, that we do not know which of the prior intervals will be included in the optimal solution for $W(j)$. If a string of intervals are rejected, our approach will effectively add up all of these gaps, and include them in the penalty for $W(j)$.

Interval j is accepted: If we add j to the schedule, we gain v_j units of value, but we cannot take any of the requests following prior(j). This leaves a gap from the finish of prior(j) to the start of j, that is, $s_j - f_{\text{prior}(j)}$ (see Fig. 2(b)). Thus, we have $W(j) = v_j + W(\text{prior}(j)) - (s_j - f_{\text{prior}(j)})$.

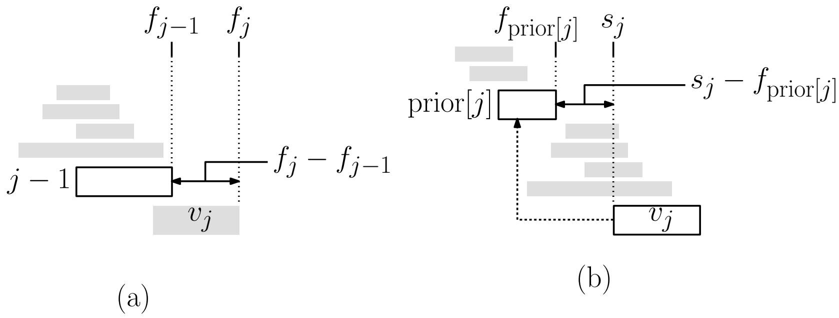


Figure 2: Weighted interval scheduling with penalties.

Clearly, we should take the better of these two options. This yields the following DP forulation:

$$W(j) = \begin{cases} 0 & \text{if } j = 0 \text{ (basis)} \\ \max \left\{ \begin{array}{ll} W(j-1) - (f_j - f_{j-1}) & \text{(reject } j\text{)} \\ v_j + W(\text{prior}(j)) - (s_j - f_{\text{prior}(j)}) & \text{(accept } j\text{)} \end{array} \right\} & \text{if } j > 0 \end{cases}$$

The final optimal answer is $W(n)$.