EECS 336: Lecture 6: Introduction to Algorithms

Dynamic Programming (cont) interval pricing

Reading: 6.5

Last Time:

- Approach: isolating previous decisions
- Shortest-paths (Bellman-Ford Alg)

Today:

- interval pricing
- summary of dynamic programming
- comparison to divide and conquer

Example: Interval Pricing

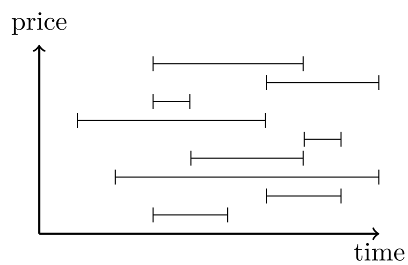
input:

- n customers $S = \{1, ..., n\}$
- T days.
- i's ok days: $I_i = \{s_i, ..., f_i\}$
- i's value: $v_i \in \{1, ..., V\}$

output:

- prices $p[t]$ for day t.
- consumer i buys on day $t_i = argmin_{t \in I_i} p[t]$ if $p[t] \leq v_i$.
- revenue $= \sum_{i \text{that buys}} p[t_i]$.
- goal: maximize revenue.

Example:

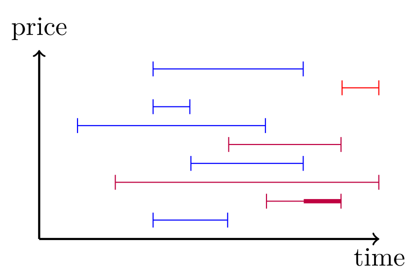


let's use dynamic programming. subproblem?

Question: What is "first decision we can make" to separate into subproblems?

Answer: day and p rice of smallest price.

Example:



Step I: identify subproblem in English

OPT(s, f, p) = "optimal revenue from customers i with intervals $\{s_i, ..., f_i\}$ contained within interval $\{s+1, ..., f-1\}$ with minimum price at least p.

Step II: write recurrence

$OPT(s, f, p)$

$= \max_{t \in \{s+1,...,f-1\}; q \in \{p,...,V\}} \text{Rev}(s, t, f, p)$

$+\text{OPT}(s, t, q)$

$+\text{OPT}(t, f, q)$.

with

> Rev(s, t, f, p) = "the revenue from customers i with intervals $\{s_i, ..., f_i\}$ contained within interval $\{s+1, ..., f-1\}$ with price p."

Step III: value of optimal solution

- optimal interval pricing = OPT$(0, T+1, 0)$

Step IV: base case

- OPT$(s, s+1, p) = 0$.
- OPT$(s, t, V+1) = 0$.

Step V: iterative DP

(exercise)

Correctness

induction

Step VI: Runtime

- precompute $Rev(s,t,f,p) in $O(T^3Vn)$ time.
- size of table: $O(T^2V)$
- cost of combine: $O(TV)$
- total: $O(T^3V(V+n))$

Note: without loss of generality T, V are $O(n)$ so runtime is $O(n^5)$.

Note: can be improved to $O(n^4)$ with slightly better program.

Step VII: implementation

(exercise)

Summary of Dynamic Programming

"divide the problem into small number of subproblems and memoize solution to avoid redundant computation."

Finding Subproblems

- identify a first decision, subproblems for each outcome of decision.
- partition problem, summarize information from one part needed to solve other part.

Subproblem Properties

1. succinct (only a polynomial number of them)
2. efficiently combinable.
3. depend on "smaller" subproblems (avoid infinite loops), e.g.,
    - process elements "once and for all" [today]
    - "measure of progress/size" [coming soon]

Runtime Analysis

runtime = initialization + size of table × cost to combine

Finding Solution

- write DP to identify value of optimal solution.
- traverse memoization table to determine actual solution.