CS130B – Data Structures And Algorithms II

Discussion Section Week 6

Written Assignment 3

Due 4:00 PM, May 19, 2017

- Homework box labeled cs130b in HFH 2108

- If late, email homework solutions

- Otherwise, submit a hard copy by the deadline
  - Do not email solutions before deadline unless there is an emergency with documentation

Problem 1

Generalized Coin Change Problem

Statement:
Given C cents and a set D of k coin denominations,

$$D = \{d_1, d_2, \ldots, d_{k-1}, d_k\}$$

return a minimum cardinality set of coins S such that

$$C = \sum_{c_i \in S} c_i$$

Problem 1

Generalized Coin Change Problem

Answer the following:

1. Is greedy approach optimal?
    - If so, prove it
    - Otherwise, provide a counterexample with a specific coin set *D* and cents *C*

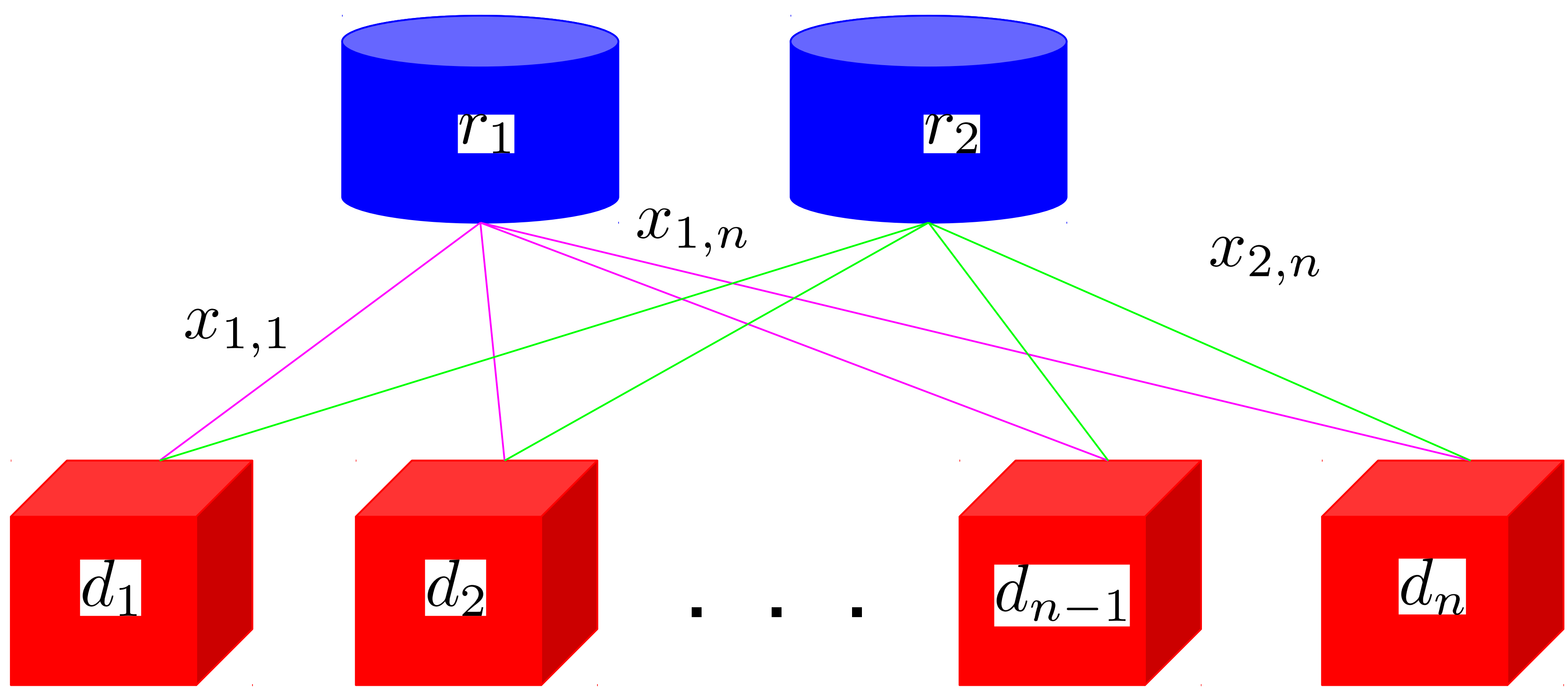
2. Devise an algorithm based on dynamic programming that returns a minimum cardinality set *S*
    - Give the recurrence relation that governs how you pick one solution from another i.e. S(c) = min S(c - <something>) + …
    - Give the time complexity of your algorithm e.g. O(C^2)

Problem 2

Warehouse Problem

$c_{i,j} =$ cost to ship 1 unit from W_i to D_j

$r_i =$ W_i inventory



$x_{i,j} = \#$ units from W_i to D_j $\qquad$ $d_j =$ demand at D_j

Problem 2

Warehouse Problem

Goal: Find $x_{i,j} \geq 0$ for all $1 \leq i \leq 2, 1 \leq j \leq n$

such that the cost $\sum_{i=1}^{2} \sum_{j=1}^{n} c_{i,j} \cdot x_{i,j}$

is minimized.

Problem 2

Warehouse Problem

To start: Let $g_i(x) =$ min cost of shipping to first i destinations when W_1 has x inventory and W_2 has $\sum_{j=1}^{i} d_j - x$ inventory

Solution: $g_i(r_1)$

Problem 2

Warehouse Problem

Answer the following:

1. Give the recurrence relation for $g_i(x)$

2. Give an algorithm (based on your recurrence relation) to find all $x_{i,j}$
   - What are the dimensions of your table of solutions?
   - How do you account for r_2? When infeasible?
   - Infeasible solutions should have cost ∞

Problem 2

Warehouse Problem

Answer the following:

3. Apply algorithm to find $x_{i,j}$ when
   - $r_1 = 10, r_2 = 7$
   - $d_j = [4, 3, 2, 5, 3]$
   - $c_{1,j} = [1, 2, 3, 1, 2], c_{2,j} = [2, 3, 3, 1, 1]$
   - Write down your table of solutions for $g_1(r_1), g_3(r_1), g_5(r_1)$
     - $g_1(r_1)$ and $g_3(r_1)$ are intermediate solutions

Problem 2

Warehouse Problem

Example:

$$r_1 = 7, r_2 = 5 \qquad c_{1,j} = [2, 3, 1]$$

$$d_j = [2, 6, 4] \qquad c_{2,j} = [1, 2, 2]$$

Answer:

$$\text{cost} = 21$$

$$x_{1,j} = [2, 1, 4]$$

$$x_{2,j} = [0, 5, 0]$$

Table of shipping costs in the end:

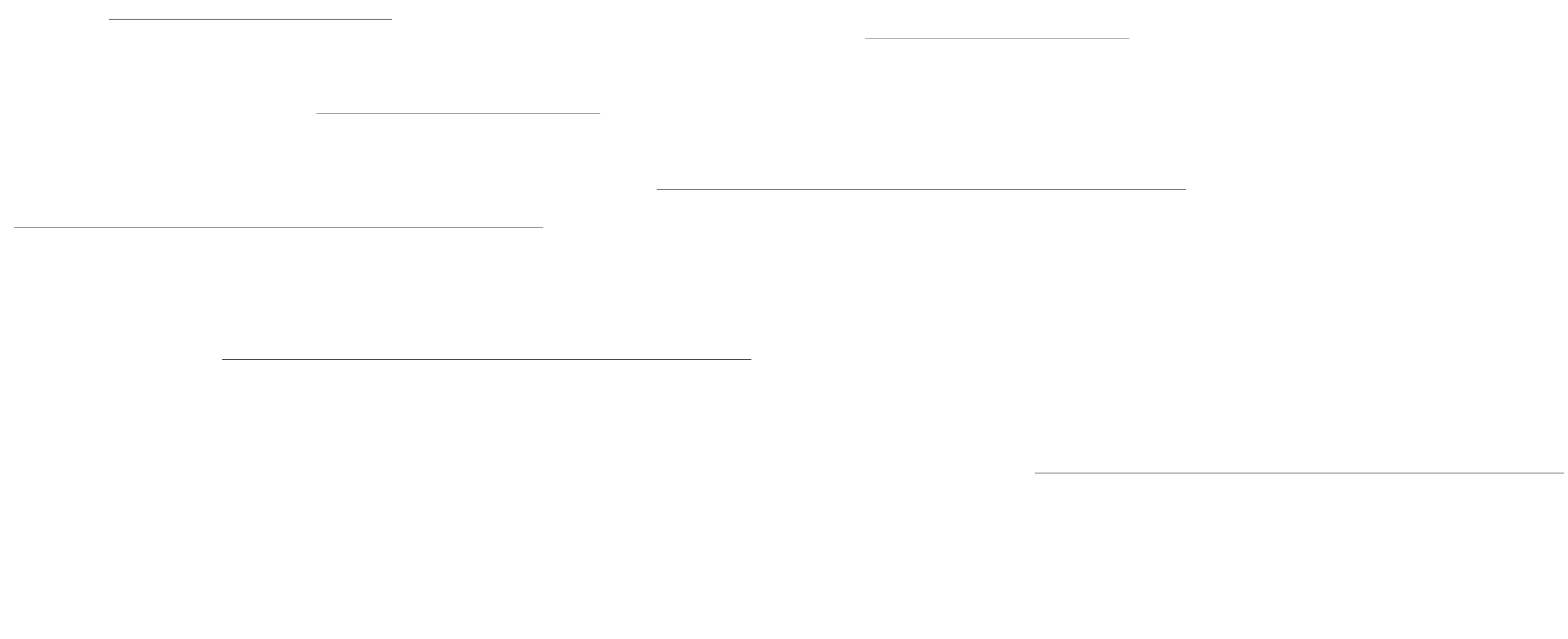
$$\begin{bmatrix} 0 & \dots & \infty \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 21 \end{bmatrix}$$

Dimensions of table?

Problem 3

Max Sum of Non-overlapping Intervals Problem

Given n intervals



Problem 3

Max Sum of Non-overlapping Intervals Problem

Statement:
Given *n* intervals

$$I = \{(s_1, e_1), \ldots, (s_n, e_n)\}$$

find the set *S* of non-overlapping intervals such that the sum of the intervals in *S*

$$\sum_{(s_i, e_i) \in S} e_i - s_i$$

is maximized.

Problem 3

Max Sum of Non-overlapping Intervals Problem

Answer the following:

1. Does greedy approach from homework 2 generate the optimal solution?
    - If so, prove it.
    - Otherwise, provide a counterexample.
2. Devise a dynamic programming algorithm to solve the problem.

Problem 3

Max Sum of Non-overlapping Intervals Problem

Example:

$$I = \{(1,3),(4,6),(6,10),(2,5),(0,5),(7,10)\}$$

Answer:

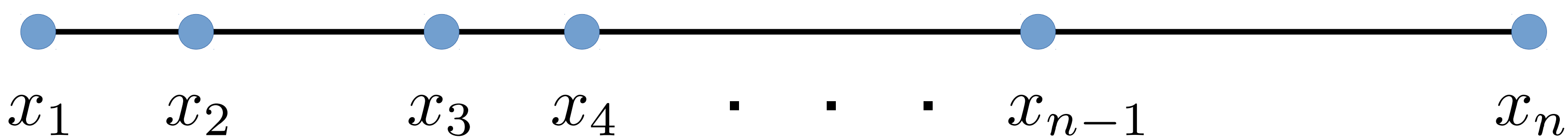
$$S = \{(0,5),(7,10)\}$$

$$\sum_{(s_i,e_i)\in S} e_i - s_i = 8$$

Dynamic Programming

Example Problem: Billboard Constructions

n building sites for billboards



Each site i generates $r_i > 0$ in revenue.

Restriction: no two billboards within 5 miles of each other.

Goal: find max revenue possible from these n sites.

Dynamic Programming

Solution: Billboard Constructions

Given first j site, determine if a billboard should be built at location j or not.

Recurrence:

$$R(1, j) = \max(R(1, j-1), R(1, k) + r_j)$$

where k is the eastmost site that is west of site j and at least 5 miles away.

Dynamic Programming

Algorithm: Billboard Constructions

$$\text{billboardRevenue}(xs, rs, M):$$

$$\quad \#\ \text{Revenue array; index 1 means location 1}$$

$$\quad R[0] = 0, R[1] = r_1$$

$$\quad \text{for } i = 2, \ldots, n:$$

$$\quad\quad R[i] = \max(R[i-1], R[\text{eastmost}(i)] + r_j)$$

$$\quad \text{return } R[n]$$

What modifications are needed to get the actual billboard locations?