Introduction to Computers and Engineering Problem Solving

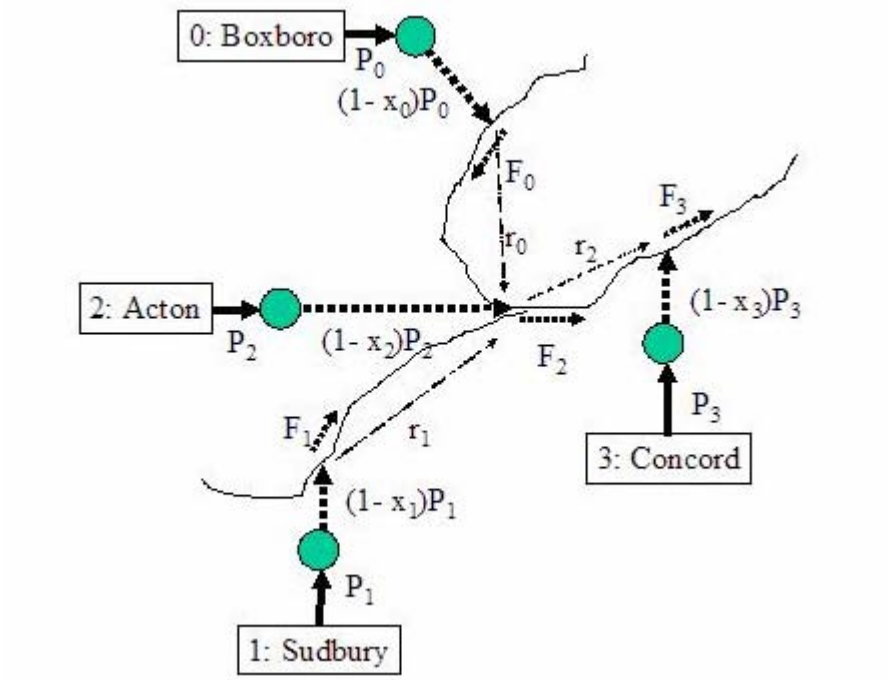
Spring 2005

Problem Set 5: Water quality

Due: 12 noon, Session 19

1. Problem statement

Wastewater and storm runoff discharges from small and large population centers degrade water quality in rivers and lakes. An example watershed with four population centers is shown below:



Each of the four towns produces pollutants at a loading rate of P_i milligrams per day. Each town has a wastewater and stormwater treatment plant that removes a fraction x_i of the pollutants, so that $(1- x_i) P_i$ milligrams per day of pollutants are discharged into the rivers next to the plants.

When the pollutants enter the river, they mix with pollution from upstream towns. The concentration of pollutants downstream of the plant is given by flow conservation:

$$C_{out} = ((1- x_i) P_i + F_{in} C_{in}) / F_{out}$$

Where C_{out} = pollutant concentration (mg/L) downstream of plant

C_{in}= pollutant concentration (mg/L) upstream of plant
F_{in}= flow (L/day) upstream of plant
F_{out}= flow (L/day) downstream of plant

After treatment from the plant, natural biological processes further reduce the pollution level; this is modeled as fraction r_i of pollutants removed.

If we assume that the pollution levels are small above Boxboro and Sudbury, we have the following equations for pollution concentration below each plant (see diagram on previous page):

$C_0 = ((1- x_0)\, P_0) / F_0$ (Boxboro)

$C_1 = ((1- x_1)\, P_1) / F_1$ (Sudbury)

$C_2 = (F_0(1-r_0)C_0 + F_1(1-r_1)C_1 + (1-x_2)P_2)/ F_2$ (Acton)

$C_3 = (F_2(1-r_2)C_2 + (1-x_3)P_3)/ F_3$ (Concord)

There are three types of treatment plant:

	Type 1	**Type 2**	**Type 3**
Unit cost($/mg removed)	$c_1+c_2P_i$	$c_3+c_4P_i^2+c_7A_i$	$c_5+c_6P_i^{1.5}$
Capacity (mg removed/day)	Unlimited	$6 * 10^9$	$4 * 10^9$
Maximum % of pollutants removed	90 if $P_i < 0.5 * 10^9$; 80 otherwise	90	95
Area A_i required (km^2)	1.0	$c_{8+}c_9\text{Capacity}_i$	1.5

where the constants c_1 through c_7 are cost coefficients:
$c_1 = \$2 * 10^{-6}$
$c_2 = \$2 * 10^{-15}$
$c_3 = \$4 * 10^{-6}$
$c_4 = \$1 * 10^{-24}$
$c_5 = \$4 * 10^{-6}$
$c_6 = \$4 * 10^{-19}$
$c_7 = \$5 * 10^{-7}$

and constants c_8 and c_9 are coefficients that determine the plant area:
$c_8 = 0.5$
$c_9 = 5 * 10^{-10}$

Note that the lower case c_i and upper case C_i are different variables.

There is a regulatory maximum that these plants must meet for all points on the river system. Pollution concentration must be less than 20 mg/L at all points. **Your program will not have to compute a solution that meets the regulatory limit; it will only display the result.** The user will rerun the model until a solution is found.

The additional data you need are given below:

Index	City	P_i (mg/day)	C_i (mg/L) if untreated	F_i(L/day)	r_i
0	Boxboro	$1 * 10^9$	100	$1 * 10^7$	50%
1	Sudbury	$2 * 10^9$	40	$5 * 10^7$	65%
2	Acton	$4 * 10^9$	47.27	$1.1 * 10^8$	40%
3	Concord	$2.5 * 10^9$	22.48	$2.5 * 10^8$	0%

Note that plant capacity is higher than pollutant load for all three plant types for these four towns. Don't check capacity in any calculations; this is a simplification.

The column "C_i if untreated" in the table above is for testing. If you set x=0 for all plants, these are the C values that your program should calculate.

2. Problem objective and approach

Write a program to analyze the effectiveness and cost of the four treatment plants on water quality in this watershed. Use inheritance to model the plants. Create an abstract base class TreatmentPlant. Create appropriate subclasses for the three types of plant. Use appropriate abstract, non-abstract or final methods in the subclasses. Use appropriate scope (private, public, package, protected) for data and methods. Give each plant an integer identifier guaranteed to be unique. You may not have a plant type variable in any of your classes; you must use polymorphism only. If a data member does not exist for all subclasses, it may not be present in the base class.

You will find it convenient to create a Town class to hold the town name, P_i, F_i, and r_i. Make the data members private and use get() methods to obtain them.

Find a good way to implement the equations for pollution concentration below each town; this requires some thought. A calculateConcentration() method in the Town or TreatmentPlant class is one option. You need to store data somewhere on which towns or plants are upstream. Also, decide if Town objects have TreatmentPlants, or whether TreatmentPlant objects have Towns (or perhaps both). Spend some time designing your classes before starting to implement them.

You may find it convenient to order the towns in downstream order; you may require the inputs, arrays, etc. to be in a convenient order for your program.

Write a TreatmentPlantTest class with only a main() method to:

1. Accept user inputs on the plant type serving each town. You don't have to check inputs for validity.
2. Accept user inputs for x_i. You don't have to check that x is less than the maximum x for the plant type; this is a simplification.
3. Create the four plants,
4. Output the pollution concentration downstream from each plant along the rivers. Use polymorphism; loop through the plants calling the same methods on all of them to generate the necessary outputs.
5. Output the cost, area, and maximum pollutants removed of each plant. Your getCost() methods must return the total cost, not unit cost.
6. Indicate whether the regulatory standard is met on each river segment.

Make your program convenient to use for repeated runs, so that an analyst can experiment with different combinations of plant types and x_i to achieve the regulatory standard at lowest cost.

There is a way to output the plant type using Java reflection. We cover it later in the term; you don't need to output plant type in this homework. Also, you don't need to output the unique plant ID (it's easy).

Turn In

1. Place a comment with your full name, MIT server username, section, TA name and assignment number at the beginning of all `.java` files in your solution.
2. Place all of the files in your solution in a single zip file.
    a. Do not turn in electronic or printed copies of compiled byte code (`.class` files) or backup source code (`.java~` files)
    b. Do not turn in printed copies of your solution.
3. Submit this single zip file.
4. Your solution is due at noon. Your uploaded files should have a timestamp of no later than noon on the due date.

Penalties

- 30% off if you turn in your problem set after Friday noon but before noon on the following Monday. You have one no-penalty late submission per term for a turn-in after Friday noon and before Monday noon.
- No credit if you turn in your problem set after noon on the following Monday.