Computer Algorithms in Systems Engineering

Spring 2010

Problem Set 6: Building ventilation design (dynamic programming)

Due: 12 noon, Wednesday, April 21, 2010

Problem statement

Buildings require exhaust systems to remove waste gases from processes conducted in a manufacturing, medical or research facility, as well as those from human activity in the building. The system must handle the required exhaust volume. Air velocity must be within given limits, primarily to limit noise and vibration. Pressure balance must be maintained at junctions between ducts; duct size must fit within the available space, and costs should be minimized while meeting all the other criteria. As an example, smaller ducts are less expensive in capital costs, but require higher fan pressure to move the required air flow, which increases the operating costs.

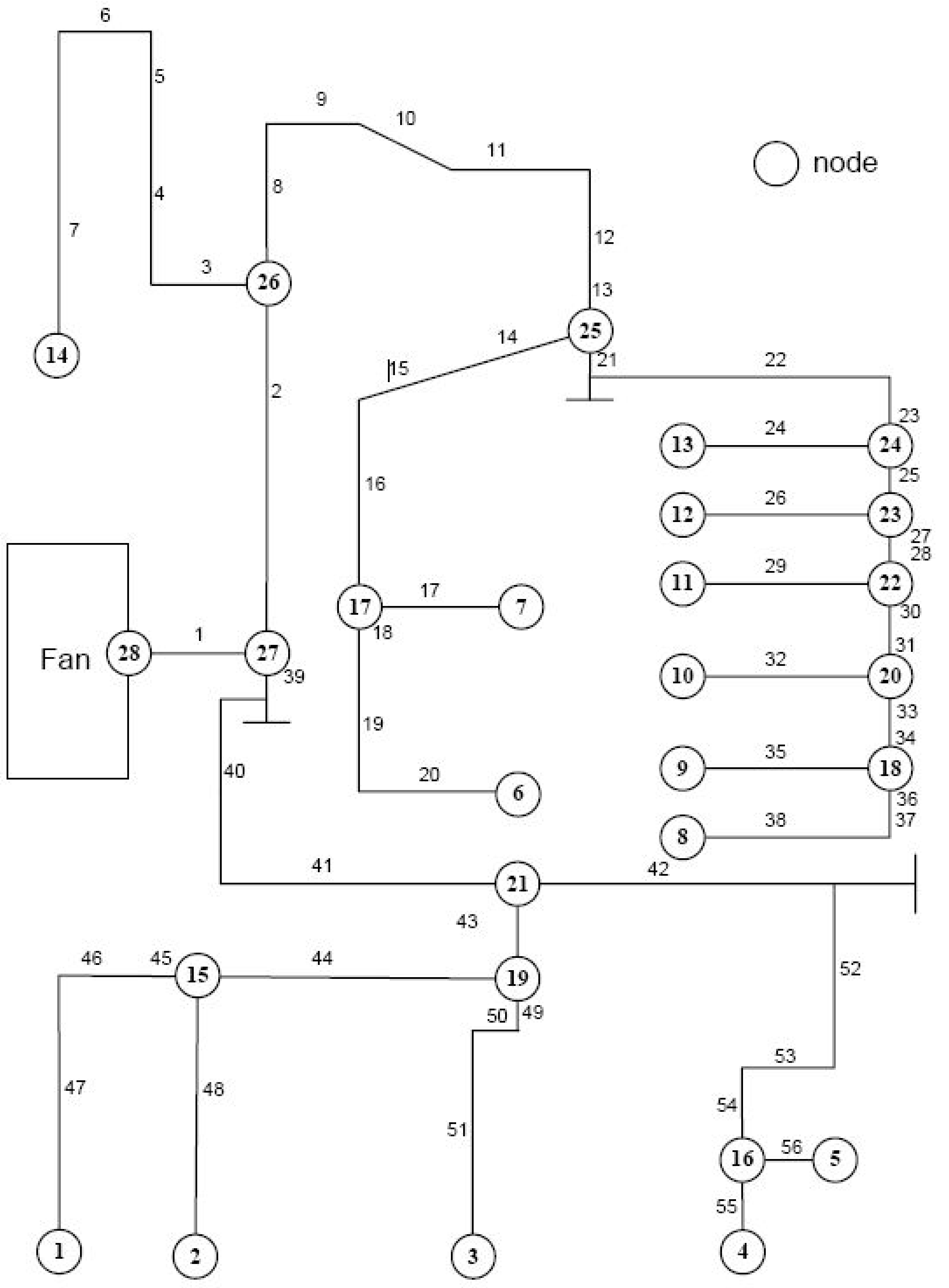
Existing manual design methods either too simple to produce a good design, or are very difficult to use. A dynamic programming approach can be used as the basis of a design tool for engineers or architects. You will write the basic portion of such a tool in this assignment.

A building exhaust system is shown below. There are 56 duct segments and 28 junctions in the network. You are given a tree representation of this network in DuctTestFragment.java. There is only one duct segment between nodes in DuctTestFragment.java; for example, segments 52, 53 and 54 are just one segment whose length is the sum of all three. Nodes 1 through 14 are where the exhaust air enters the system; it then flows through nodes 15 through 27, to the fan (node 28, which you may model separately). There is just a single fan that draws the exhaust volume out from the top of the system. The length of each duct segment and the design of the entire system depend on the building layout.

Nodes 1 through 14 have data on the exhaust volume that enters the system. This is the flow through the duct segment immediately upstream (toward the fan) from the node. The ducts upstream from nodes 15 through 28 have a flow that is the sum of the flows entering from earlier ducts (flow conservation). We assume the ducts are round; each duct (segment between nodes) has a minimum and maximum diameter, and it has a minimum and maximum air velocity. For simplicity, the minimum diameter= 0.3 meters and the maximum diameter= 1.5 meters for all ducts. These size constraints are imposed by the building design. The minimum velocity= 1 m/sec and the maximum velocity= 10 m/sec, except at the duct upstream of node 27 (on the roof), where the maximum velocity= 30 m/sec. If the velocity is too low, granular material may not be carried out through the system; if the velocity is too high, noise and vibration occur.

When two ducts join at a node, the pressure of the incoming air must be the same in both ducts. (Otherwise air would flow from one duct into the other, instead of toward the fan.)

The system design must obey these constraints. The system is to have the minimum cost (capital plus operating) while meeting the constraints.



For each duct segment, your program will select a set of pressure drops from 5 to 100 Pascals, in increments of 5 Pascals. (These are slight pressure differences; 100,000 Pascals= 1 atmosphere.) The maximum total pressure drop at the fan is 100 Pascals. (This resembles a knapsack problem, in that each duct can use a portion of the 100 Pascal limit of pressure drop, and we wish to choose the pressure drops in each segment so that costs are minimized, similar to profit being maximized—think of it as a sign change.) Because pressure is a continuous quantity, we must have the same pressure in both ducts feeding into a node and this is most easily achieved by choosing a fixed set of pressures.

Given a desired pressure drop p (Pa, or Pascals) in a single duct, and input values of air flow Q (m^3/sec) and length L (m) of a duct, the required duct size d (in meters) is given by a simplified version of the usual equation:

$$d = (0.03\ Q^2\ L/p)^{0.2} \qquad (1)$$

Given a duct size d, the capital cost c of a duct is based on its cylindrical area and the cost per m^2 of duct material c_d (\$/m^2):

$$c = \pi\ d\ L\ c_d \qquad (2)$$

The air velocity v (m/sec) in the duct is:

$$v = 4\ Q/\ (\pi\ d^2\ ) \qquad (3)$$

The capital cost of the exhaust system is the sum of the capital costs of all the duct segments (equation 2), multiplied by a capital recovery factor (crf) to amortize those costs to an annual cost. Use crf= 0.14 for this problem.

The operating cost of the exhaust system is the power cost of the fan, which depends on the total pressure difference p (Pa) it must maintain, the air flow Q (m^3/sec), the cost of electricity c_e (\$/kwh) and the efficiency of the fan e (0.64). The equation for operating cost o (\$/yr) is:

$$o = 0.001\ Q_{fan}\ p_{fan}\ c_e\ T/e \qquad (4)$$

where Q_{fan} is the air flow at the fan (m^3/sec), p_{fan} is the pressure drop the fan must maintain (Pa), c_e is the cost of electricity (\$0.60/kwh), T is the number of hours per year that the system operates (8640) and efficiency e= 0.64. The 0.001 factor converts from watt hours to kilowatt hours (kwh).

Assignment

Solve for the minimum cost (fan operating cost plus duct capital cost) exhaust system, subject to duct sizes and air velocities staying between the minimum and maximum constraints for each duct. The figure shows the connections of the ducts, which is another constraint. Last, the pressure of each duct coming into a node must be the same. Again, you will choose a preselected set of pressures from 5 Pascals to 100 Pascals for each duct. To obtain a small pressure change, the duct must be large (increasing capital cost) but the operating cost is lower. Velocity is higher in small ducts; the maximum velocity constraint puts a lower bound on duct size.

Start at nodes 1-14, which have pressure= 0 (atmospheric pressure) at their inlet, and compute the range of pressure changes from 5 to 100 Pa in increments of 5 Pascals. For each pressure value, you can compute the duct diameter, cost and velocity. When ducts join at a node, they must have the same pressure. At the node, the duct exiting the node will have the combined air flow of the two incoming ducts, and its starting pressure will be the pressure of the two incoming ducts. Repeat the calculations until reaching the fan node. At that point, you have a set of pressures that are feasible at the fan. Apply the operating cost and capital cost factors at the fan node to find the cheapest total cost. This will occur at one pressure level at the fan. Then, trace back through all the nodes to find the pressure drops (and duct size, cost and velocity) of each duct in the system. Output the total cost, capital cost, operating cost, and the duct size, velocity, air flow and cost of each duct segment.

You may do this in any way that you wish. A few hints on how the solution was written:

- The exhaust system is a tree, not a graph, and a tree is a natural way to generate and solve the dynamic program. Use a postorder traversal of the tree to make sure you visit each downstream node before each upstream node. This will take you through the stages of the dynamic program in the correct order: each downstream duct will have been computed before any upstream duct is computed.
    - Use the Tree class we covered in lecture earlier, and the postorder traversal that it contains.
    - The Node class has a postorder method as well. Use this to compute the pressure, duct size, velocity and cost of the duct upstream of the node. Keep an array of the pressures and the costs (cumulative). Then, take the pressure of the incoming ducts and add their costs, pressures and flow volumes to those of the upstream duct. (This is very similar to adding an item to a knapsack—you add its weight (pressure, or the scarce resource that has a constraint) and you add its profit (negative of cost, or objective) You will do some 'label correcting' at this point, so keep a predecessor array that records the pressure of the incoming ducts (remember both must be the same) as well as the total pressure. This will let you trace the solution in the backward pass later.
- When you reach the sink (fan) node, you will have a set of pressures and capital costs. Choose the pressure that results in the lowest total cost. Once you've chosen it, you can find its predecessor node, which will tell you the pressure in the ducts that fed into the fan node.
- This starts the backward pass, to trace out the solution. Use a preorder traversal, which visits a parent before either of its children. Your algorithm is guaranteed to have looked at the parent and chosen the predecessor pressure in the child nodes. Each tree node knows its left and right child (see the DuctTestFragment.java code), so you can set the predecessor pressure in the left and right child before processing it later in the traversal. Write out the pressure for each duct as you encounter it. It is probably easiest to recompute the duct size, cost and velocity from the pressure, rather than to store these variables.

- When the all nodes have been preorder traversed, you are done,

You are given DuctTestFragment.java. You may modify it as you wish. The simplest modification is just to add calls to tree.postorder(), tree.fanNode(), and tree.preorder(), and a bit of output. In the solution, there is only one other class, DuctTree.java, which contains a DuctNode inner class. DuctNode needs only preorder(), postorder() and methods to compute the equations given above.

If you are not comfortable with these hints, you may use your own approach. Using a standard dynamic programming graph is awkward, because there is no easy way to order the stages (ducts) so they connect logically. You don't have to use the set-based dynamic programming model; while it could be used, it seems too complex. Using a tree, where each node is a stage and each pressure value is a state, is likely to be the most natural solution, as suggested by the hints.

Turn In

1. Place a comment with your full name, Stellar username, 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 on the 1.204 Web site under the appropriate problem set number. For directions see **How To: Submit Homework** on the 1.204 Web site.
4. Your solution is due at noon. Your uploaded files should have a timestamp of no later than noon on the due date.
5. After you submit your solution, please recheck that you submitted your .java file. If you submitted your .class file, you will receive **zero credit.**

Penalties

- 30 points off if you turn in your problem set after Wednesday (April 21) noon but before noon on Friday (April 23). You have two no-penalty two-day (48 hrs) late submissions per term.
- No credit if you turn in your problem set after noon on the following Friday.