Prof. Sergio A. Alvarez
21 Campanella Way, room 569
Computer Science Department
Boston College
Chestnut Hill, MA 02467 USA

http://www.cs.bc.edu/∼alvarez/
alvarez@cs.bc.edu
voice: (617) 552-4333
fax: (617) 552-6790

CS383, Algorithms
Spring 2009
HW9 Solutions

1. You will design a dynamic programming algorithm for the following computational task. The inputs are two arrays. The desired output is the length of the longest subsequence that appears in both of the input arrays. Elements of the shared subsequence need not occupy immediately consecutive locations in each array. For example, for the two arrays $\{12, 3, -20, 2, -8, 17, 40, 2, 30\}$ and $\{-50, 2, -8, 17, 2, 30, 40\}$, the desired output is 5, since the two input arrays share a subsequence of length 5 (e.g., $\{2, -8, 17, 2, 30\}$) but no subsequence of length greater than 5.

   (a) Specify each of the ingredients of a dynamic programming solution that we discussed in class. Explain in detail in each case.

      i. suproblem definition
      ii. objective function, value function (optimized objective function)
      iii. recurrence relation for the value function
      iv. domain of dependence of a generic subproblem
      v. boundary conditions, and
      vi. update order

   ## Solution

      i. Suproblem definition: for each pair (i, j) with $1 \leq i \leq n$ and $1 \leq j \leq m$, consider the subproblem of finding the length of the longest common subsequence of the prefix arrays $a[1...i]$ and $b[1...j]$, without regard for the end position of the subsequence within a and b.
      ii. The objective function is just the subsequence length. The *value* function, i.e., the optimized objective function, is $L(i, j) =$ the maximum length of a common subsequence of the prefixes $a[1...i]$ and $b[1...j]$.
      iii. Recurrence relation for the value function: we will express $L(i, j)$ in terms of the values of L for smaller subproblems. There are several cases. If $a[i]$ and $b[j]$ are the same, then the common element may be appended to the longest common subsequence that ends before index i in a and before index j in b. Otherwise, the longest common subsequence may end at index i in a *or* at index j in b, but not both since $a[i]$ and $b[j]$ are different. The longest subsequence in the latter case is the longest

among three competing alternatives depending on the endpoints within a and b. The final recurrence relation is shown below.

$$L(i,j) = \begin{cases} L(i-1,j-1)+1, & \text{if } a[i]=b[j] \\ \max\{L(i-1,j),\ L(i,j-1),\ L(i-1,j-1)\}, & \text{if } a[i]\neq b[j] \end{cases}$$

One of the three arguments to the max function in the second case can actually be dropped without affecting the solution. Which one?

iv. Domain of dependence of the subproblem at (i_0, j_0): tracing back the dependencies in the recurrence relation, the domain of dependence is clearly the rectangle in the subproblem space consisting of all (i,j) with $0 \leq i < i_0$ and $0 \leq j < j_0$.

v. Boundary conditions: $L(i,0) = 0 = L(0,j)$ for all $0 \leq i \leq n$ and $0 \leq j \leq m$.

vi. Update order: outer loop over $j = 1...m$ and inner loop over $i = 1...n$, or vice-versa. The important point is that the values of $L(i,j)$ over the entire domain of dependence of (i_0, j_0) must have been updated before attempting to update $L(i_0, j_0)$ itself.

(b) Building on your work in 1a, provide detailed pseudocode for a dynamic programming algorithm that solves the computational task described above.

Solution

See Algorithm 1.

Algorithm 1: Longest common subsequence
Input: Arrays $a[1...n]$, $b[1...m]$.
Output: Length of the longest common subsequence of a and b.

```
dpLCS($a[1...n], b[1...m]$)
(1)   foreach $i = 0...n$
(2)      $L(i,0) = 0$
(3)   foreach $j = 0...m$
(4)      $L(0,j) = 0$
(5)   foreach $j = 0...m$
(6)      foreach $i = 0...n$
(7)         if $a[i] = b[j]$
(8)            $L(i,j) = L(i-1,j-1)+1$
(9)         else
(10)           $L(i,j) = \max\{L(i-1,j),\ L(i,j-1),\ L(i-1,j-1)\}$
(11)  return $L(n,m)$
```

(c) Analyze the running time of your algorithm from 1b.

Solution

The running time is dominated by the nested loops. Assuming $O(1)$ time for each individual arithmetic operation, the total time is proportional to the number of passes through the inner loop, i.e., $\Theta(mn)$.

2. A manufacturing plant needs to hire workers to meet a sudden rise in demand. The total budget for salaries is \$402,000. Due to the cost of health benefits, at most 100 workers can be hired. Two companies are offering temporary staff: A offers up to 66 workers who earn a \$5,000 salary and who are expected to produce 60 units each; B offers up to 72 workers who will earn \$3,000 and are expected to produce 50 units each. Because of pre-existing agreements with the two temp companies, at least one out of every seven hires must come from each company. The plant's goal is to determine how many workers to hire from each company to maximize the total number of units produced.

   (a) Cast the factory hiring problem as a linear programming problem in the standard form discussed in class. Specify the variables (and their meanings), the objective function, and all constraints. The objective function must be given as a specific linear combination of the variables. Similarly, each constraint must be expressed as an inequality that provides an upper bound on some specific linear combination of the variables.

Solution

Since we seek to decide how many workers to hire from each company, we use two variables

$$x_A, \quad x_B$$

whose values equal the respective numbers of workers from the two companies. The objective function is the quantity to be maximized, that is, the number of units produced. Since each worker from company A produces 60 units and each worker from company B produces 50 units, the objective function may be written in terms of the selected variables as follows:

$$60x_A + 50x_B$$

We now add the constraints. First, constraints for the maximum numbers of workers available from each company:

$$x_A \leq 66, \; x_B \leq 72$$

Since total salaries cannot exceed 402,000:

$$5000x_A + 3000x_B \leq 402000$$

(this constraint may be simplified by dividing both sides by 1000). Since the total number of workers cannot exceed 100:

$$x_A + x_B \leq 100$$

It must also be stated that the number of hires from each company must be non-negative:

$$x_A \geq 0, \; x_B \geq 0$$

Since at least 1/7 of the workers must come from each company:

$$x_A \geq \frac{1}{/}7(x_A + x_B), \quad x_B \geq \frac{1}{7}(x_A + x_B)$$

The last two constraints simplify to:

$$-6x_A + x_B \leq 0, \quad x_A - 6x_B \leq 0$$

(b) Sketch the feasible region for the linear programming problem from 2a. How many vertices does the feasible region have? Include the equation of each boundary and the numerical coordinates of each of the vertices.

Solution

The feasible region is an irregular hexagon with vertices at the six points

$$(0,0),\ (12,72),\ (28,72),\ (51,49),\ (66,24),\ (66,11)$$

Proceeding through this vertex list in order, the equations of the boundary lines between consecutive vertices are

$$\begin{aligned} x_B &= 6x_A \\ x_B &= 72 \\ x_A + x_B &= 100 \\ 5x_A + 3x_B &= 402 \\ x_A &= 66 \\ x_A &= 6x_B \end{aligned}$$

(c) Explain the steps followed by a greedy simplex search in finding the solution to the linear programming problem in 2a. Provide detailed calculations of all of the relevant values. Give the optimal hiring decisions and the associated number of units produced.

Solution

Greedy hillclimbing search of the vertices (x_A, x_B) of the feasible region, stopping when no neighboring vertex leads to a higher number of units produced. Optimal vertex is $(51, 49)$, with corresponding production level $60(51) + 50(49) = 5510$ units.

(d) Would the solution to the factory hiring problem change if company A's workers produced 100 units each (all other amounts remain equal)? Explain in detail.

Solution

Yes. Exercise.

3. Recall the cargo plane example that we discussed in class (exercise 7.3 in the textbook by Dasgupta et al). A plane has weight capacity 100 tons and volume capacity 60 cubic meters. Three materials are available for loading, in limited amounts:

- $40m^3$ of material m_1, with density 2 tons/m^3 and value \$1,000/m^3
- $30m^3$ of material m_2, with density 1 ton/m^3 and value \$1,200/m^3
- $20m^3$ of material m_3, with density 3 tons/m^3 and value \$12,000/m^3

We wish to maximize the total value of the shipment subject to the above constraints.

(a) Formulate the cargo plane task as a linear programming problem in the standard matrix form described in class (see below). In detail, specify the contents of each of the matrices c, A, b. Your answers should make it clear what the size of each matrix is, and what specific numerical value appears in each (row, column) combination. Explain briefly.

Standard matrix form of LP problem.

$$\min c'x \quad \text{subject to } Ax \leq b,$$

where x is an n-dimensional column vector of unknowns, c is an n-dimensional vector containing the coefficients of the objective function, A is an $m \times n$ matrix describing m linear expressions (one constraint per row) in the n unknowns, and b is an n-dimensional column vector corresponding to the right-hand sides (upper bounds) of the m inequality constraints whose left-hand sides correspond to the rows of A. Note carefully that the standard form allows minimization only and that all constraints must be inequalities of the form described above.

Solution

Let x_i be the volume of material m_i to be loaded, $i = 1, 2, 3$. The objective function is the total value of the shipment, which is:

$$1000x_1 + 1200x_2 + 12000x_3$$

The corresponding vector c is

$$c = \begin{bmatrix} 1000 \\ 1200 \\ 12000 \end{bmatrix}$$

The required volume, weight, and non-negativity constraints are:

$$x_1 + x_2 + x_3 \leq 60$$
$$2x_1 + x_2 + 3x_3 \leq 100,$$

with the corresponding matrices:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \end{bmatrix} \qquad b = \begin{bmatrix} 60 \\ 100 \end{bmatrix}$$

(b) Building on your work in 3a, use Matlab to solve the cargo plane problem. Provide both the coordinates of the optimal vertex, as well as the optimal value of the objective function (both of these should be computed in Matlab). Include a printout of the commands used to enter the data and to compute the solution.

4. Consider the maximum flow problem for the network shown in exercise 7.10 of the textbook, with S as source and T as sink. In this task you will cast this problem as a linear programming problem in standard form and use Matlab to solve it.

   (a) For the maximum flow problem for the network in exercise 7.10 of the textbook, explain carefully how many variables (unknowns) are needed, and what the meaning of each of them is. Label the variables individually as $x_1, ...x_n$ and give the meaning of each in terms of relevant quantities in the network itself.

Solution

The graph for this task has 15 edges. One variable is needed to represent the flow along each edge, hence we need 15 variables:

$$\begin{aligned}
x_1 &= \text{flow } S \to A \\
x_2 &= \text{flow } S \to B \\
x_3 &= \text{flow } S \to C \\
x_4 &= \text{flow } A \to B \\
x_5 &= \text{flow } A \to D \\
x_6 &= \text{flow } A \to E \\
x_7 &= \text{flow } B \to E \\
x_8 &= \text{flow } C \to B \\
x_9 &= \text{flow } C \to F \\
x_{10} &= \text{flow } D \to E \\
x_{11} &= \text{flow } D \to G \\
x_{12} &= \text{flow } E \to F \\
x_{13} &= \text{flow } E \to G \\
x_{14} &= \text{flow } F \to T \\
x_{15} &= \text{flow } G \to T
\end{aligned}$$

(b) For the same network, state what the objective function is. Write the objective function explicitly in terms of appropriate variables from among those that you described in 4a, using the same variable names as before.

Solution

The objective function measures the quantity to be optimized, which in this case is the total flow leaving node S:

$$x_1 + x_2 + x_3$$

Equivalently, the objective function could be taken as the total flow entering node T, which is the same as the flow leaving node S due to the conservation of flow constraints below.

(c) State the necessary inequality constraints for the network flow problem for the above network. Describe how many constraints are needed, and of what types. List the constraints explicitly, using the notation $x_1 ... x_n$ for the variables from 4a.

Solution

All flow variables must be non-negative (15 inequality constraints):

$$x_i \geq 0, \quad i = 1...15$$

Also, each variable must not exceed the capacity of the corresponding edge in the graph (15 inequality constraints):

$$\begin{aligned} x_1 &\leq 6 \\ x_2 &\leq 1 \\ x_3 &\leq 10 \\ x_4 &\leq 2 \\ x_5 &\leq 4 \\ x_6 &\leq 1 \\ x_7 &\leq 20 \\ x_8 &\leq 2 \\ x_9 &\leq 5 \\ x_{10} &\leq 2 \\ x_{11} &\leq 5 \\ x_{12} &\leq 6 \\ x_{13} &\leq 10 \\ x_{14} &\leq 4 \\ x_{15} &\leq 12 \end{aligned}$$

and conservation of flow must be satisfied at each node of the graph other than the source and sink nodes (7 equality constraints over the seven "internal" nodes A, B, C, D, E, F, G):

$$\begin{aligned} x_1 &= x_4 + x_5 + x_6 \\ x_2 + x_4 + x_8 &= x_7 \\ x_3 &= x_8 + x_9 \\ x_5 &= x_{10} + x_{11} \\ x_6 + x_7 + x_{10} &= x_{12} + x_{13} \\ x_9 + x_{12} &= x_{14} \\ x_{11} + x_{13} &= x_{15} \end{aligned}$$

This gives a grand total of 30 inequality constraints and 7 equality constraints. Each of the equality constraints should be split into two inequality constraints, yielding a total of 44 inequality constraints altogether.

(d) Cast the maximum network flow problem for the above network in the standard matrix-vector form described in 3a. Specify the column vectors b and c, and the matrix A in detail (show all of the elements of each, and what position each element occupies). You should retain the same labeling/ordering of variables that you described above in 4a and that you used for the preceding parts of this task.

Solution

The flow maximization objective must be transformed into a minimization. This is easy to accomplish by simply changing the sign of the objective function:

$$\min -x_1 - x_2 - x_3$$

The transpose of the appropriate c vector that represents the above objective function is:

$$c' = [-1\ -1\ \ -1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\,0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\,0]$$

The non-negativity constraints must be changed to the form:

$$-x_i \leq 0,\ \ i = 1...15$$

The 15 capacity constraints are fine in their initial form from the preceding subtask. Each flow equality constraint must be split into two inequalities in the appropriate form.