CS570 FALL 2006 FINAL SOLUTION

Problem 1: 1.True 2.True 3.False 4.False 5.True 6.True 7.False 8.Unknown 9.Unknown 10.True

Problem 2:

Solution 1): Assume that the weight of material 1, material 2 and material 3 carried on the cargo plane are x_1,x_2 and x_3 respectively. Clearly $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$. Then the total revenue is $1000 * x_1/2 + 1200 * x_2/1 + 12000 * x_3/3$. As the cargo plane can carry a maximum of 100 tons and a maximum of 60 cubic meters of cargo, we have $x_1 + x_2 + x_3 \leq 100$, $x_1/2 + x_2/1 + x_3/3 \leq 60$; As Material 1 has maximum available amount 40 cubic meters, we have $x_1/2 \leq 40$;As Material 2 has maximum available amount 30 cubic meters, we have $x_2/1 \leq 30$; As Material 3 has maximum available amount 20 cubic meters, we have $x_3/3 \leq 20$; Hence the linear program which optimize revenue is:

$$\max(1000 * x_1/2 + 1200 * x_2/1 + 12000 * x_3/3)$$

$$\text{subject to: } \begin{cases} x_1 \geq 0, \\ x_2 \geq 0, \\ x_3 \geq 0, \\ x_1 + x_2 + x_3 \leq 100, \\ x_1/2 + x_2/1 + x_3/3 \leq 60, \\ x_1/2 \leq 40, \\ x_2/1 \leq 30, \\ x_3/3 \leq 20. \end{cases}$$

Solution 2): Assume that the amount of material 1, material 2 and material 3 carried on the cargo plane are x_1,x_2 and x_3 respectively. Clearly $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$. Then the total revenue is $1000 * x_1 + 1200 * x_2 + 12000 * x_3$. As the cargo plane can carry a maximum of 100 tons and a maximum of 60 cubic meters of cargo, we have $x_1 + x_2 + x_3 \leq 60$, $x_1 * 2 + x_2 * 1 + x_3 * 3 \leq 100$; As Material 1 has maximum available amount 40 cubic meters, we have $x_1 \leq 40$;As Material 2 has maximum available amount 30 cubic meters, we have $x_2 \leq 30$; As Material 3 has maximum available amount 20 cubic meters, we have $x_3 \leq 20$; Hence the linear program which optimize revenue is:

$$\max(1000 * x_1 + 1200 * x_2 + 12000 * x_3)$$

Date: Dec 8, 2006.

$$\text{subject to: } \begin{cases} x_1 \geq 0, \\ x_2 \geq 0, \\ x_3 \geq 0, \\ x_1 + x_2 + x_3 \leq 60, \\ x_1 * 2 + x_2 * 1 + x_3 * 3 \leq 60, \\ x_1 \leq 40, \\ x_2 \leq 30, \\ x_3 \leq 20. \end{cases}$$

Problem 3: The problem is essentially the same as Problem 1 in Chapter 5, which is assigned in HW#5.

We denote $A[1,n], B[1,n]$ as the sorted array in increasing order and $A[k]$ as the $k-th$ element in the array. For brevity of our computation we assume that $n = 2^s$. First we compare $A[n/2]$ to $B[n/2]$. Without loss of generality, assume that $A[n/2] < B[n/2]$, then the elements $A[1], ..., A[n/2]$ are smaller than n elements, that is,$A[n/2], ..., A[n]$ and $B[n/2], ..., B[n]$. Thus $A[1], ..., A[n/2]$ can't be the median of the two databases. Similarly, $B[n/2], ..., B[n]$ can't be the median of the two databases either. Note that m is the median value of A and B if and only if m is the median of $A[n/2], ..., A[n]$ and $B[1], ..., B[n/2]$(We delete the same number of numbers that are bigger than m and smaller than m), that is, the $n/2$ smallest number of $A[n/2], ..., A[n]$ and $B[1], ..., B[n/2]$. Hence the resulting algorithm is:

Algorithm 1 Median-value($A[1,n], B[1,n], n$)
if $n = 1$ then
    return min(A[n],B[n]);
else if $A[n/2] > B[n/2]$
    Median-value($A[1, n/2], B[n/2, n], n/2$)
else if $A[n/2] < B[n/2]$
    Median-value($A[n/2, n], B[1, n/2], n/2$)
end if

For running time, assume that the time for the comparison of two numbers is constant, namely, c, then $T(n) = T(n/2) + c$ and thus $T(n) = c \log n$. Hence our algorithm is $O(\log n)$.

Problem 4: This problem is actually equivalent to solving the Subset Sum problem: Given n items $\{1, , n\}$ and each has a given nonnegative weight a_i, we want to maximize $\sum a_i$ given the condition that $\sum a_i = t$. If the maximum value returned is t, then the output is yes, otherwise false. The recursive relation is:

$$\begin{cases} OPT(i,t) = OPT(i-1,t), \text{if } t < a_i; \\ OPT(i,t) = \max(OPT(i-1,t), a_i + OPT(i-1,t-a_i), \text{otherwise}; \end{cases}$$

Time complexity: Note the we are building a table of solutions M, and we compute each of the values $M[i,t]$ in $O(1)$ time using the previous values, thus the running time is $O(nt)$.

Or The Subset-Sum algorithm given in the textbook is $O(nt)$, hence the running time of our algorithm is $O(nt)$.

Problem 5: This problem can easily be reduced into max flow problem. We denote the M faculty as $\{x_1, ..., x_M\}$ and N courses as $\{y_1, ..., y_N\}$. The reduction is as follows: add source s, sink t in the graph. For each x_i, add edge $s \to x_i$. The capacity lower bound of these edges is 1 and the capacity upper bound is 2.For each

y_i, add edge $y_i \rightarrow t$. The capacity of each edge is 1. For each faculty x_i, if x_i prefer course y_{i_1} and y_{i_2}, add edges $x_i \rightarrow y_{i_1}$ and $x_i \rightarrow y_{i_2}$ in the graph. The capacity of each edge is 1. To find a feasible allocation, we just need to solve the max flow problem in the constructed graph to find a max flow with value N.

Problem 6:

a)To compute the length of the longest path, we just need to find a integer k such that $D(G,k) = Yes$ and $D(G,k+1) = No$. Hence the algorithm is:

$k = 0$;

while $D(G,k) = Yes$

    $k++$;

return k;

b)True. The longest path problem is NP-complete. The proof is as follows: Clearly given a path we can easily check if the length is k in polynomial time, hence the problem is NP. To prove that it is NP-complete, we prove that even the simplest case of longest path problem is NP-complete. The simplest case in our longest path problem is that the weight of each edge is 1. We use a simple reduction from HAMILTON PATH problem: Given an undirected graph, does it have a Hamilton path, i.e, a path visiting each vertex exactly once? HAMILTON PATH is NP-complete. Given an instance $G' =< V', E' >$ for HAMILTON PATH, count the number $|V'|$ of nodes in G' and output the instance $G = G', K = |V'|$ for LONGEST PATH. Obviously, G' has a simple path of length $|V'|$ if and only if G' has a Hamilton path. Therefore, if we can solve LONGEST PATH problem, we can easily solve HAMILTON PATH problem. Hence LONGEST PATH Problem is NP-complete.