CSI 3105 Assignment 4, 2011
Solutions

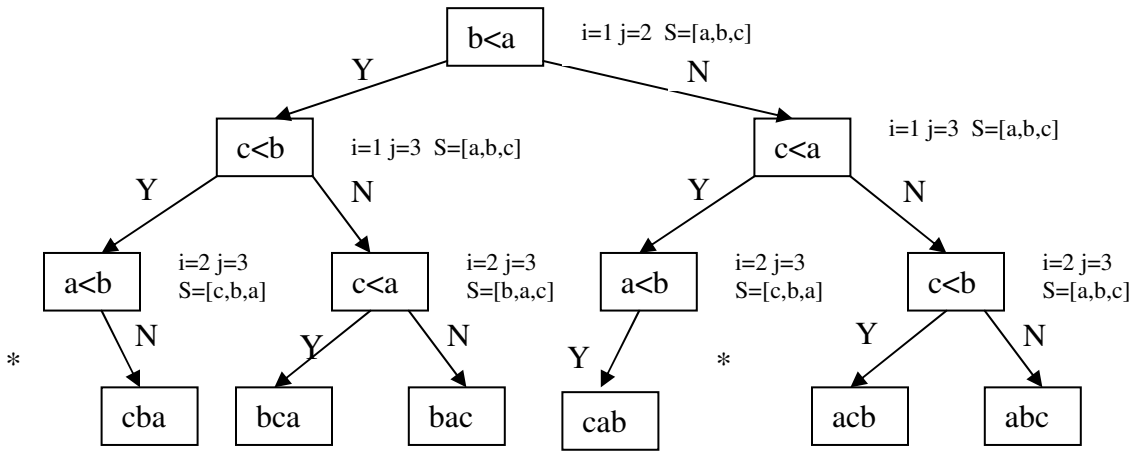
1. [10 marks]
We would need to use something like "- ∞" (i.e. the largest negative number possible in our implementation language) instead of -1 as the indicator in the array *distance* that a node is in the tree. So we could change "distance[near]:= -1" to be "distance[near]:= - ∞". Then change the line "if 0 <= distance[i]<= min" to be "if distance[i] ≠ - ∞ and distance[i]<= min".

2. [10 marks]

Stage	Job Considered	Time Slot For Job	Schedule
1	2	1	2 _ _ _ _
2	3	5	2 _ _ _ 3
3	4	None	2 _ _ _ 3
4	1	4	2 _ _ 1 3
5	7	3	2 _ 7 1 3
6	6	2	2 6 7 1 3
7	5	None	2 6 7 1 3

Final Schedule: 2 6 7 1 3
Profit: 190

3. [10 marks total]
a) [8 marks]
(* indicates a path that will never be taken by the algorithm)



b) [1 mark] W(3) = number of edges in a longest root to leaf path in tree
= depth of tree
= 3

c) [1 mark] fewest comparisons = number of edges in a shortest root to leaf path in tree
= 3.

4.
Form of certificate for the yes answer for graph colouring (there are, of course, different forms you could use for the certificate):

We will assume that the nodes have a predetermined ordering (eg. V1, V2, etc.) and that the colour labels are simply 1, 2, 3, …..etc.

Let string S consist of the colour labels for each node, in the predetermined order for the nodes (i.e. the first colour in the list is for the first node, the second colour for the second node, etc.).

Algorithm A to verify S represents a colouring of G using at most K colours:

i) Make sure the number of colours listed is n (i.e. one for each node).
This can be done in one scan of the string—so $\Theta(n)$.

ii) Make sure the labels are all 1, 2, 3,…,K (i.e. there are no labels in S that don't represent a colour for us). This will also ensure that the number of colours is at most K.
This could be done by simply checking each number is in the range >0 and <=K, so in one scan of S—$\Theta(n)$.

iii) Make sure the colours represent a colouring, i.e. that the colours for the ends of each edge are different colours.
To do this: Assume that G is stored as a list of edges. Now put the colours from S into an array C indexed from 1 to n. For each edge ij in the edge list, we could then simply check that $C[i] \neq C[j]$. The work for all of this would be $\Theta(m)$.

The total complexity for Algorithm A is $\Theta(\max(m,n))$ which is polynomial time as required.

5. [20 marks total
a) 10 marks
Description of transformation algorithm TRAN:

Take each clause in the CNF expression for the SAT problem and add "$\vee z$", where z is a new variable that wasn't previously in the expression.

b) 2 marks

$$(x_1 \vee \overline{x}_2 \vee x_5 \vee z) \wedge (\overline{x}_1 \vee x_3 \vee z) \wedge (x_2 \vee x_3 \vee z)$$

c) 8 marks

i) Proving that a yes answer for the instance y = TRAN(x) implies a yes answer for the original SAT problem.

Suppose y=TRAN(x) is a yes instance of OneF-SAT. Then there is an assignment Ψ of T/F values for the variables in y such that there is at least one T and at least one F in each clause.

Case 1: The value for z in Ψ is F.

In this case there must be at least one T in the rest of each clause of z, so if we take Ψ and remove the value for z we have an assignment for the variables in the original SAT with at least one T in each clause. So x is a yes instance of SAT as required.

Case 2: The value for z in Ψ is T.

In this case there must be at least one F in the rest of each clause of z, so if we take Ψ and remove the value for z and reverse the assignment of all other variables we have an assignment for the variables in the original SAT with at least one T in each clause. So x is a yes instance of SAT as required.

ii) Proving that a yes answer for an instance x of SAT implies a yes answer for the OneF-SAT problem y = TRAN(x):

Suppose we have a yes instance x of SAT. Then there is an assignment of T/F values for the variables in x such that there is at least one T in each clause. Take this assignment and add in the assignment "z=F". Then this is an assignment for the OneF-SAT problem such there is at least one T and one F in each clause. So y is a yes instance of OneF-SAT, as required.