CPSC-320 Intermediate Algorithm Design and Analysis Winter 2000

Solutions to Practice Homework # 5

1. Consider the following activities and times:

```
Activities |  A  B  C  D  E  F  G  H  I  J  K
----------------------------------------------
Start      | 11 11 12 12  1  2  3  4  4  5  5
End        |  1  1  2  2  3  4  5  6  6  7  7
```

   The only activity with two conflicts is F. All others have three or more conflicts. If F is scheduled, then the maximum number of activities that can be scheduled is three, e.g. A, F, H. However, A, E, G, K is a larger set of activities that can be scheduled.

2. We can get a 3-sequence LCS algorithm by extending the non-recursive 2-sequence algorithm given in the Cormen et al. text or class lecture notes.

   Once all array entries are computed, for sequences of length n, the array entry Length[n,n,n] contains the length of LCS.

```
function LCS-length(X, Y, Z)
    for i <-- 1 to n do Length[i, 0, 0] <-- 0
    for j <-- 1 to n do Length[0, j, 0] <-- 0
    for k <-- 1 to n do Length[0, 0, k] <-- 0

    for i <-- 1 to n do
        for j <-- 1 to n do
            for k <-- 1 to n do
                if X[i] = Y[j] = Z[k] then
                     Length[i, j, k] <-- Length[i-1, j-1, k-1] + 1
                else if Length[i-1, j, k] >= Length[i, j-1, k] and
                        Length[i-1, j, k] >= Length[i, j, k-1] then
                           Length[i, j, k] <--  Length[i-1, j, k]
                     else if Length[i, j-1, k] >= Length[i-1, j, k] and
                        Length[i, j-1, k] >= Length[i, j, k-1] then
                           Length[i, j, k] <--  Length[i, j-1, k]
                     else Length[i, j, k] <--  Length[i, j, k-1]
    return Length[n,n,n]
```

   The algorithm has a running time of $O(n^3)$.

3. Note that $\text{lcm}(a, b) = ab/\text{gcd(a,b)}$. Therefore, one gcd calculation, one multiplication, and one division is needed to calculate $\text{lcd}(a, b)$. If n is the number of bits of a and b, the time for multiplication and division is $O(n^2)$. The number of steps of the gcd algorithm is $O(n)$, but each step involves a division operation. Hence the total time is $O(n^3)$.

4. Roughly, the idea of the 2SAT algorithm is as follows. Pick any variable, say z. You don't know whether you should set z = true or z = false in order to satisfy the give formula F. But in the case of a 2SAT formula, one can try both possiblities as follows. To try z = true, you set z = true. This satisfies some clauses of F, which can be removed. In the clauses containing $\bar{z}$, the other literal in the clause is now "forced" to be true if there is to be any hope of satisfying the formula.

Continue to set the values of “forced” variables and remove clauses that are satisfied by the truth assignment of that variable until either (i) there are no more “forced” variables, or (ii) a problem arises where a variable is “forced” to be both true and false.

In case (i), you are left with a smaller 2SAT formula. If it is empty, you conclude that the formula is satisfiable and halt. Otherwise, repeat the whole process on the smaller 2SAT formula.

In case (ii) you conclude that setting $z =$ true doesn’t work. You repeat the process on F, this time starting with setting $z =$ false. If again case (ii) is reached, you conclude that it the formula is not satisfiable since it it not possible to find a satisfying assignment either when $z = true$ or when $z = false$.