CS2223
HW#6

DUE: Tuesday, December 11

1 (8 points) Do **Problem 8.28** on page 281 of our text. An algorithm is *efficient* if its execution time is a polynomial function of the number of symbols in the input string *x*. (Hint: For each substring $x_i...x_j$ of input string $x_1...x_n$, it would help to know which symbols of $\{a,b,c\}$ could be achieved with appropriate parenthesizations of $x_i...x_j$.)

Solution. We define $n \times n$ array *A* whose entries are members of the power set of $\{a,b,c\}$. Actually, we only compute the values of $A[i, j]$ for which $1 \le i \le j \le n$. For each contiguous substring $x_i...x_j$ of input string $x_1...x_n$, we construct the set $A[i, j]$ such that for each $y \in \{a,b,c\}$, $y \in A[i, j]$ if and only if we can parenthesize $x_i...x_j$ to yield *y*. That is, if $x = abcc$, then *A*[1,3]={*a*,*b*} because $x_1...x_3 = abc$ and the two ways to parenthesize $x_1...x_3 = abc$ are $(ab)c = bc = a$ and $a(bc) = aa = b$.

for $i \leftarrow 1$ **to** n **do** $A[i,i] \leftarrow \{x_i\}$
for $s \leftarrow 1$ **to** n-1 **do**
    for $i \leftarrow 1$ **to** n-s **do**
        $j \leftarrow i + s$
        $A[i, j] \leftarrow \varnothing$
        for $k \leftarrow i$+1 **to** j **do**
            $A[i, j] \leftarrow A[i, j] \cup \{uv | u \in A[i, k-1] \wedge v \in A[k, j]\}$
return $a \in A[1, n]$

2. (5 points) (Extension of **Problem 8.17** of our text) Describe a polynomial time algorithm (a program is not necessary) to test if a graph has a negative cycle, that is, a cycle such that the sum of the lengths of its edges is negative.

Solution A graph has a negative cycle if and only if some vertex *i* lies on a negative cycle.

run *Floyd*'s algorithm
NegativeCycle? ← false
for $i \leftarrow 1$ **to** n **do**
    if $D[i,i]<0$ **then** *NegativeCycle?* ← true
return *NegativeCycle?*

3. (6 points) Let A be an $m \times n$ array of 0s and 1s, and we seek a path from $A[1,1]$ to $A[m,n]$ such that every element on the path has an entry of 1 and one entry can follow another on the path if one is above the other or below the other or left of the other or right of the other. For example, in array A[7,7]

1	1	1	1	0	0	0
1	0	0	1	0	0	1
0	1	0	1	1	0	1
0	0	1	1	0	0	1
1	1	1	0	0	1	1
0	1	0	0	0	1	0
1	1	1	1	1	1	1

a path is

(1,1), (1,2), (1,3), (1,4), (2,4), (3,4), (4,4), (4,3), (5,3), (5,2), (6,2), (7,2), (7,3), (7,4), (7,5), (7,6), (7,7).

a Describe an algorithm (a program is not necessary) with execution time in $\Theta(mn)$ to find a path from $A[1,1]$ to $A[m,n]$.

b Describe an algorithm (a program is not necessary) with execution time in $\Theta(mn)$ to find a shortest path from $A[1,1]$ to $A[m,n]$.

Solution ***a*** and ***b***

Given $A[m,n]$, we construct a graph $G = (N, A)$ with vertices $N = \{(i, j) | 1 \le i \le m \wedge 1 \le j \le n\}$ and two vertices (i, j) and (i', j') are adjacent in G if and only if $A[i, j] = A[i', j'] = 1$ and $(i = i' \wedge j = j' \pm 1) \vee (j = j' \wedge i = i' \pm 1)$. Note that G can be constructed from A in time in $\Theta(mn)$.

Next we do a breadth first search in G from vertex (1,1) and we halt when vertex (m,n) is added to the breadth first search tree. The path from (1,1) to (m,n) is a shortest path in G (and in A).