Dynamic Programming Problems Related to Binomial Coefficients

CIS 407
binomial coefficients

• C(n,k) = “n choose k” = number of different ways to choose k elements from a set of n elements
• base: C(n,0) =1 and C(n,n)=1
• to pick k elements, we consider using the nth element
• we can decide not to use it so now pick k elements from n-1 (a)
• or use it and pick k-1 elements form k-1 (b)
• C(n,k) = (a) + (b) = C(n-1,k)+C(n-1,k-1)
count substrings

• How many ways is string T a subsequence of string S?
• example: S=ABCABC and T=AC, answer is 3
• example: S=AAAAAAAAAAAA and T=AAA, answer is C(10,3)=120
• subproblem: CT(i,j) is the number of ways s_1s_2...s_i can contain t_1t_2...t_j as a subsequence
• note: CT(i,j) <= C(i,j) (think of the locations)
• base case 1: CT(i,j) = 0 if i<j (not possible)
• base case 2: CT(0,0) = 1
• enough base cases?
main recurrence for CT

• look at last character of $s_1s_2...s_i$ and $t_1t_2...t_j$
• does $s_i$ match $t_j$?
• if not (ABCA and AC), remove last character of $S$: compare ABC and AC.
• in this case $CT(i,j) = CT(i-1,j)$
• if they do match (ABCABC and AC) we can decide not to take the match: compare ABCAB to AC
• or take the match: compare ABCAB to A
• both correspond to distinct matchings
• sum up the two choices: $CT(i,j) = CT(i-1,j) + CT(i-1,j-1)$
summary

• CT(i,j) is the number of ways \(s_1s_2...s_i\) can contain \(t_1t_2...t_j\) as a subsequence
• CT(i,j) = 0 if \(i<j\)
• CT(0,0) = 1
• if \(s_i=t_j\), then CT(i,j) = CT(i-1,j) + CT(i-1,j-1)
• else CT(i,j) = CT(i-1,j)
public static int count() {
    int limit;

    for (int i=0; i<=N; i++)
        ct[i][0]=1;

    for (int i=1; i<=N; i++) {
        limit=i;
        if (limit>M)
            limit=M;
        for (int j=1; j<=limit; j++) {
            ct[i][j] = ct[i-1][j];
            if (SA[i-1]==TA[j-1])
                ct[i][j] += ct[i-1][j-1];
        }
    }
    return ct[N][M];
}