Azalee McAlpine

Maximum value of Contiguous Subsequence

1. M(j): max sum over all windows ending at j

2. M(j) = max{M(j=1)+A[j], A[j]}

3. Max (M(i)) for i in [1..L]

Making Change

1. M(j): minimum number of coins required to make change of money j.

2. M(j) = min {M(j-vi)} + 1

3. M(j)

Longest Increasing Subsequence

1. L(j) longest strictly increasing subsequence ending at position j

2. L(j) = Max, i < j, A[i] < A[j] for {L(i)} + 1

3. max{L(j)}

Balanced Partition

1. P(i, J) = 1 if some subset of A1 to Ai has a sum of j. Else its zero. I can run from 1 to n and j can run 0 to n\*k

2. P(i, j) = max{P(i-1, j), P(i-1, j-Ai)}

3. S = (summation of all Ai) /2

Box Stacking

1. H(j): tallest stack of boxes with box j on top

2. H(j) = if i<j, wi>wj, di>dj max{H(i)} +h(j)

3. max{H(j)}

Building Bridges

1. X(i): index of corresponding city on northern bank

2. L(j) = Max, i < j, A[i] < A[j] for {L(i)} + 1

3. max{L(j)}

Integer 0/1 Knapsack Problem

1. M(i, j) : optimal value for filling exactly a capacity j knapsack with some subset of items 1...i

2. M(i, j) = max{M(i-1, j), M(i-1, j-si) + vi}

3. max {M(n, j)}

Edit Distance

1. T(i,j): minimum cost to transform A[i...j] into B[1...j]

2. T(i, j) = min Cd + T(i-1, j)

min T(i, j-1) + Ci

if A[i} = B[j] min T(i-1, j-1)

if A[i} != B[j] min T(i-1, j-1) +Cr

3. T(i, j)

Counting Boolean Parenthesizations

1. T(i, j) : number of ways to parenthesize symbols I...j such that this subexpression evaluates to true.

F(i, j) : number of ways to parenthesize symbols I...j such that this subexpression evaluates to false.

2. T(i, j) = i<=k<=j-1 Summation

T(i, k) T(k+1, j)

total(i,k) total(k+1,j) – F(i,k)F(k+1,j)

T(i,k)F(k+1,j) + F(i,k)T(k+1, j)

3. T(1,n)

Optimal strategy for a game

1. V(i,j) max value we can definitely win if its our turn and only coins vi...vj remain

2. V(i,j) = max{min { v(i+1, j-1), v(i+2, j)}+vi , min { v(i, j-2), v(i+1, j-1)}+vj}

3. Max(min(v))