Sweet Tapas

4. Bucky Badger is having dinner at an upscale tapas bar, where he will order many small plates. There are N plates of food on the menu, where information for plate i is given by a triple of non-negative integers (v_i, c_i, s_i): the plate's volume v_i, calories c_i, and sweetness $s_i \in \{0, 1\}$ (the plate is sweet if $s_i = 1$ and not sweet if $s_i = 0$). Bucky is on a **diet**: he wants to eat no more than C calories during his meal, but wants to fill his stomach as much as possible. He also wants to order exactly $S < N$ sweet plates, without purchasing the same dish twice.

   Describe an $O(NCS)$-time algorithm to find the maximum volume of food Bucky can eat given his diet.

Solution:

10 20 40 | 80

40 —

60 40 20 —

70 60 50 40 30 20 10 —

BC:

$dp[0][c][0] = 0$

when 0 plates left

$dp[0][c][s] = -\infty$

not exactly s plates then

move to next plate

this many cals left now

$\{0,1\}$

$$dp[i][c][s] = \max\left(v_i + dp[i-1][c-c_i][s-s_i],\ dp[i-1][c][s]\right)$$

$i = 0 \ldots N$

take

don't take

total maximum volume of food

Calories $\le C$ and order only $S < N$ sweet plates

Name: ______________________ Wisc id: ______________________

Coin Change (again)

1. *CLRS 3rd edition (p. 446).* Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer. Give an $O(nk)$-time algorithm that makes change for any set of k different coin denomination, assuming one of the coins is a penny.

answer represents fewest amount of coins to total n

Solution:

$\overbrace{1¢ \quad 5¢ \quad 10¢ \quad 25¢}^{k=4}$ $\quad n = 30$

$\overbrace{10¢ \quad 20¢ \quad 30¢}^{k=3}$ $\quad n = 40$

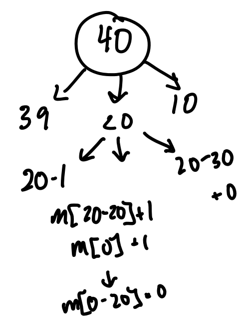
for j from 1 to 30 (k)

$m[40] = \min(m[i - v_j]) + 1$

$m[40] = \min(m[40-1]) + 1 \rightarrow \min[39]+1$

$m[40] = \min[m[40-20]) + 1 \rightarrow \min[20]+1$

$m[40] = \min(m[40-30]) + 1 \rightarrow \min[10]+1$



$v_j = [1, 20, 30]$

$m[40] = \min(m[40 - v_j]) + 1$;

$\min(m[40], m[40 - v_j]) + 1$

min # of coins to achieve i

$$M[i] = \min_{j=1\ldots k}(m[i], m[i - v_j]) + 1$$

k different coin denominations

BC: $i - v_j < 0$

$m[i] = 0$

$O(nk)$

n grids, k coins

the answer will be located at m[n] where n is the goal cents

Coin Collecting

2. Several coins of different value are placed on a $m \times n$ board. Let $c_{i,j} \geq 0$ be the value of the coin placed on grid-(i, j). Note that $c_{i,j} = 0$ implies that there is no coin on grid-(i, j).

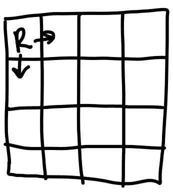
   You placed a robot on the board to collect the coins. The robot starts at the top-left corner of the board (grid-$(0, 0)$), but it can only move right or down. What is the maximum total value of coins the robot can collect?

   left or up

Solution:

n

m



goal grid-(m,n) b/c can't move right or down anymore.
find max from each path of choosing either right or down @ each step.
basing this off of the previous choice that its made so start at g(m,n) go from there

m: 1…i

n: 1…j

left   up

$m[i,j] = \max(m[i,j-1] + c_{i,j} , m[i-1,j] + c_{i,j})$

vs

$m[i,j] = \max(m[i,j-1] , m[i-1,j]) + c_{i,j}$

answer = m[m,n]

BC: edge / corner
m[i,j] = m[i,j]

Knapsack

3. (0-1 Knapsack) Given n items where i-th item has value v_i and weighs w_i, output the maximum value for a knapsack with capacity of W. Note that each item can be chosen at most once.

collect items to be as close to W without exceeding

Solution:

$dp[i][w] = \max(dp[i-1][w] , v_i + dp[i-1][w-w_i])$

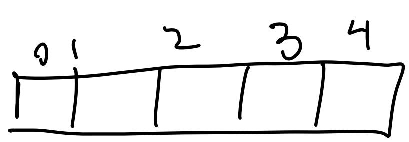
not pick i   take i

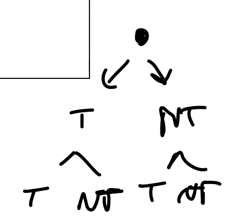
max total val considering 1..i w/ weight ≤ W

iterate through all items & choose if pick v_i or not

answer @

BC:
(1) i=0 (no items left)
M[i] = 0 ← value = 0 for nonexisting item
(2) w = 0 (capacity = 0) or w − wi ≤ 0
M[i] = 0





4. (Unbounded Knapsack) Given n items where i-th item has value v_i and weighs w_i, output the maximum value for a knapsack with capacity of W. Note that each item can be chosen **unlimited times**.

makes it like coin problem

Solution:

n items: $v_1 \quad v_2 \quad v_3 \; \dots \; v_n$
$w_1 \quad w_2 \quad w_3 \; \dots \; w_n$

$m[v, w]$

$$dp[w] = \max_{j=1 \dots n} \left(dp[w], \; v_j + dp[w - w_j] \right)$$

start capacity

max total value w/ width $\le w$

answer $dp[W]$

40
30 20 10

w_1	w_2	w_3	w_n	W

create a 1D array where last item holds the answer

5. (Multidimensional Knapsack) Given n items where i-th item has value v_i, weighs w_i, **and has size** d_i, output the maximum value for a knapsack with capacity of W and size of D. Note that each item can be chosen at most once.

Solution:

don't take take

$$dp[i][w][d] = \max \left(dp[i-1][w][d], \; v_i + dp[i-1][w - w_i][d - d_i] \right)$$

weight $\le w$
size $\le d$

$$dp[0][w][d] = 0$$

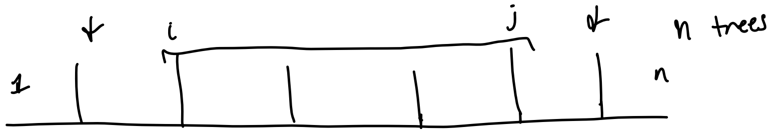
Farmer

6. A farmer has a row of n trees that grow fruits, and he wants to earn some money by cutting down the trees and selling the fruits. He is old so he only cuts down one tree per day. Also, each day he only cuts down the leftmost tree or the rightmost tree.

   Initially, cutting down the i-th tree allows the farmer to earn $v_i > 0$ dollars. As time goes by, a fruit might rot, so cutting it down earlier allows the farmer to earn more! Cutting down the i-th tree on the d-th day allows the farmer to earn $v_i \times (n - d + 1)$ dollars.

   What is the maximum amount of money the farmer could achieve?

Solution:



$v_i * d$ dollars

day 1: 5×1
day 2: 3×2
day 3: 2×3

$$m_{i,j} = \begin{cases} \max\left(m_{i+1,j} + v[i] * age,\ m_{i,j-1} + v[j] * age\right) \\ i = j,\quad v[i] * n \end{cases}$$

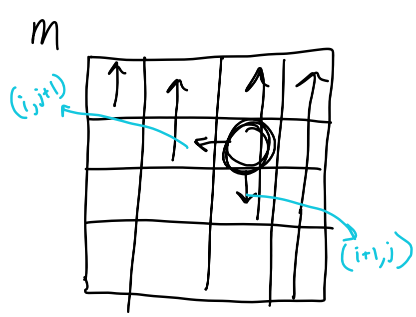
max value achievable considering trees i…j

ith tree

jth tree

$n-(j-i)$

$n-(j-i)$



left → right
bottom → top

$O(n^2) + O(1)$

counter:
5 1 1 1 9 9 4

4×1	5×1
5×2	1×2
1×3	1×3
1×4	1×4
1×5	4×5
26	34

∴ choosing min won't work

Name: ______________________ Wisc id: ______________________

Robber

1. You are a robber that plans to rob n houses along a street. Each house stashes some amount of money $v_i > 0$. However, you cannot rob two adjacent houses since this will alert the police. What is the maximum amount of money you can rob without alerting the police?

Solution:

take ↓ don't take ↓

$$M[i] = \begin{cases} v_0 & i=0 \\ v_1 & i=1 \\ \max(\underline{v_i} + M[i-2],\ M[i-1]) & i = 1 \cdots n \end{cases}$$

↑ max amt of money can rob

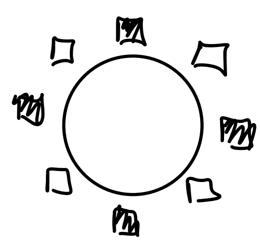
$M : v_1 \ v_2 \ v_3 \ v_4 \ v_5$

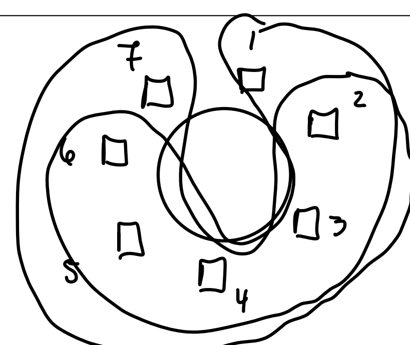
$v_1 + v_3 + v_5$

$M[i-2] + v_i$

2. What if the houses are arranged in a circle?

Solution:





$v_1 \cdots v_n$

next to each other too

max between take $1 - n-1$

and

$2 - n$

Convex Polygon Triangulation

3. You have a convex n-sided polygon where each vertex has an integer value.

   You will triangulate the polygon into $n-2$ triangles. For each triangle, the value of that triangle is the product of the values of its vertices, and the total score of the triangulation is the sum of these values over all $n-2$ triangles in the triangulation.

   Return the smallest possible total score that you can achieve with some triangulation of the polygon.

Solution:

Sweet Tapas

4. Bucky Badger is having dinner at an upscale tapas bar, where he will order many small plates. There are N plates of food on the menu, where information for plate i is given by a triple of non-negative integers (v_i, c_i, s_i): the plate's volume v_i, calories c_i, and sweetness $s_i \in \{0, 1\}$ (the plate is sweet if $s_i = 1$ and not sweet if $s_i = 0$). Bucky is on a **diet**: he wants to eat no more than C calories during his meal, but wants to fill his stomach as much as possible. He also wants to order exactly $S < N$ sweet plates, without purchasing the same dish twice.

   Describe an $O(NCS)$-time algorithm to find the maximum volume of food Bucky can eat given his diet.

Solution:

(v_1, c_1, s_1) $\leq C$

(v_2, c_2, s_2) S can't be 1 for all N plates

max vol 1...i, calories $\leq C$, exactly s sweet plate.

$$dp[i][c][s] = \max\left(d[i-1][c][s],\ v_i + d[i-1][c-c_i][s-s_i]\right)$$

answer = $dp[N][C][S]$

don't take item take item

$S < N$ $S < N$

if $s - s_i < 0$

$d[v_i][c_i][s_i]$

$dp[0][c][0] = 0$

$dp[0][c][s] = -\infty$

not exactly s sweet plates

if

$S = \{0, 1, 1, 0\} = 2$

v_0

LIS

5. Given an integer array, return the length of the longest strictly increasing subsequence. Can you come up with a $O(n \log n)$ solution?

Solution: