15.2 DP

1) Define the structure

Normally check Palindrome by comparing each end for equality
keep checking for equality in characters
Let str[0...n-1] be a string of length N
Let L[0..n-1][0..n-1] be a two dimensional array of lengths
if str[0] = str[n-1], solve for str[1] = str[n-2]
else solve str[0] = str[n-2] and str[1] = str[n-1]

2) Overlapping subproblems

Each solution can solve for next solution
start from bottom-up, use to solve substrings
Solve subpalindromes of length 1, then 2, ... then n

3) use solution in optimal solution

$L(n, m) = \max(L(n+1, m), L(L(n, m-1))$

```
Subpalindrome(str, n) {
  Let S be an empty string.
  Let A[][] be a symmetric array of size n^2, B[][] is strings
  initialize each symmetric index with 1 in A
  For subn ← 2 to n do
    For i ← 1 to n-subn do
      j ← i + subn - 1
      if str[i] = str[j] & subn = 2
        A[i][j] ← 2, B[i][j] ← str[i] + str[j]
      if str[i] = str[j]
        A[i][j] ← A[i+1][j-1] + 2, B[i][j] ← str[i] + B[i-1][j-1] + str[j]
      else
        A[i][j] = max(A[i+1][j], A[i][j-1])
        if A[i+1][j] is max
          B[i][j] = B[i+1][j]
        else
          B[i][j] = B[i][j-1]
  Return B[1][n-1]
```

i ← Values up to w that can be obtained

16.2-2 1) Define the structure m[i,w] → maximum value attainable with weight under w.

$m[0,w] = 0$

$m[i,w] = m[i-1,w]$ if $w_i > w$ (when the item is too heavy)

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

2) overlapping subproblems

m[i,w] is an array which can hold previous solutions.

Can do a bottom-up approach.

0-1 knapsack ($\vec{V}$, $\vec{W}$, w, n) where $\vec{V}$ is an array of values, W is an array of weights, w is the maximum weight, n is the # of distinct values

Let m[n,w] be a 2 dimensional array to hold optimal values

```
For i ← 1 to n
    m[0,j] ← 0
For i ← 2 to n do                                   } O(n)
    For j ← 1 to W do                               } O(w)
        if w[i] > j then
            m[i,j] ← m[i-1,j]
        else
            m[i,j] ← max(m[i-1,j], m[i-1,j-w[i]] + V[i])
```

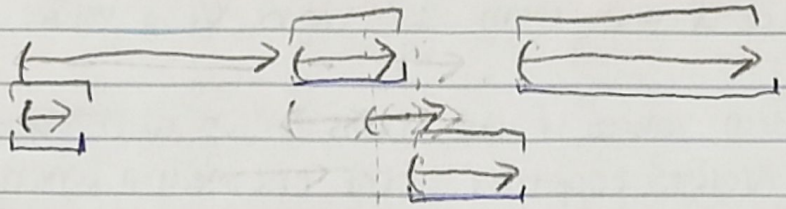
↑ is new weight and value is greater than previous

$T(n) = O(nW)$ time

25.2-2

In the $\Theta(n^3)$ space solution, only the previous matrix is used to find the solution, so in theory, this solution should work. For this algorithm, the d_{ij}^{k} term needs the parameters d_{ij}^{k-1}, d_{ki}^{k-1}, d_{ik}^{k-1} to be unchanged. d_{ij}^{k-1} will remain unchanged. The other terms, d_{ki}^{k-1} and d_{ik}^{k-1}, can be trusted to be maintained because any path with k will have it only once since there is no negative weight cycle. This algorithm should work because of these properties.

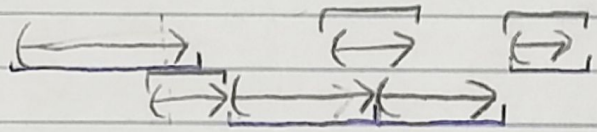
16.1-3 Let S_i be an array with time intervals.

S_0 = 

→ works, however S_2

⌊⌋ → Indicates proposed solution

⌊⌋ → Correct solution

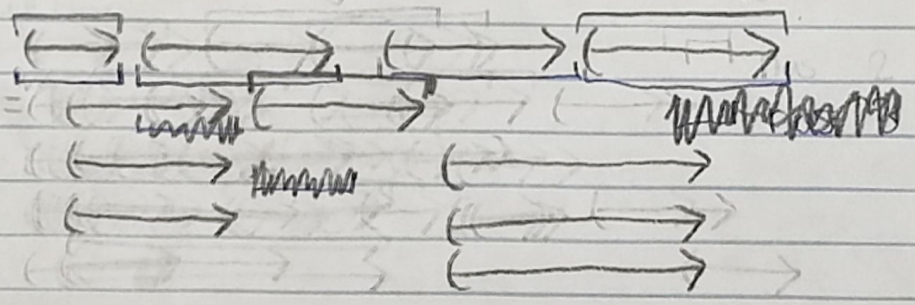
S_1 = 

⌊⌋ has 4

⌐¬ has 3

So the solution to pick the smallest compatible time intervals DOES NOT WORK because a small interval could overlap two longer ones that would produce 2 activities. For this reason, this algorithm is incorrect.

Next, assume the algorithm that selects the activity that overlaps the fewest other remaining activities after adding intervals

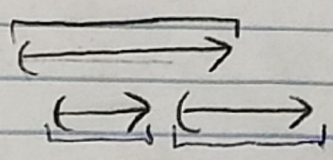
S_2 = 

⌐¬ has 3

⌊⌋ has 4

S_2 shows this algorithm does not work because the best solution could be within overlapping intervals, so the overlaps could be misleading

Next, prove an algorithm the selects least start time compatibility does not work

S_3 = 

⌐¬ has 1

⌐¬ has 2

S_3 does not work because by selecting the earliest time, it could knock of two aligned activities because it does not consider that.

O(1) 0) BASE CASE: If size of values and weights is 1, return $\frac{v_i}{w_i}$

(16.2-6) Fractional knapsack in O(n) time

O(n) { 1) Find the value of each item $\frac{v_i}{w_i}$ where v_i is values and w_i is weight

O(n) { 2) Find the median values of the items which will partition it.

O(n) { 3) Add all the values together after the median which will be Msum.

$T(\frac{n}{2})$ { 4) If, $M > W$, solve for the upperbound
Else, solve on the lowerbound with maximum weight $W - M$ } recursively

O(1) 5) Return M

$T(n) = T(\frac{n}{2}) + O(n) = O(n)$