Lecture 9: More Dynamic Programming

Tim LaRock

larock.t@northeastern.edu

bit.ly/cs3000syllabus

Business

- Homework 1 is graded
  - If you have asked for clarification and haven't heard back, hold tight! I am getting to it.
- Homework 2 due Tuesday night at midnight Boston time
  - Solutions will be released 8AM Weds; absolutely no late submission without prior permission!
- Midterm 1 Wednesday 8PM through Friday 8PM

Homework 1

Homework 2

- Question 3: Assume you have a function IsMinimumLength() that tells you whether a valid chain C is minimum length in constant time

- Question 4 adjusted to be a bit easier
  - Part a: Write a recurrence for Opt(i,j)
    - More to come on this today
  - Part b: Describe how to fill a dynamic programming table for Opt
  - Part c: Write in pseudocode how to fill the table

Putting edit distance on hold for 1 class!

- We came up with a dynamic programming solution to Subset Sum
- We found a recurrence for Edit Distance, but we still need to develop a dynamic programming solution

$$Edit(i,j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ \min \begin{Bmatrix} Edit(i, j-1) + 1 \\ Edit(i-1, j) + 1 \\ Edit(i-1, j-1) + [A[i] \neq B[j]] \end{Bmatrix} & \text{otherwise} \end{cases}$$

- But…

This week

Today:

- Revist Subset Sum to explain $Opt(i, j)$ solutions
- Introduce and solve the Knapsack Problem

Tomorrow:

- Find a dynamic programming solution for Edit Distance
- Wrap up dynamic programming
- Introduce basic features of graphs to get us started on graph algorithms

Wednesday:

- First half-ish: Continue with graph algorithms
- Second half-ish: Answers to student-submitted questions (form to be sent out this evening)

Thursday:

- No class while midterm exam is out

Subset Sum Recap

$$T(n) = 2T(n-1) + O(1) \le O(2^n)$$

```
SubsetSum(X[1..n], i, T):
  If T = 0:
    return True
  ElseIf T < 0 or i = 0:
    return False
  Else:
    with ← SubsetSum(X, i-1, T – X[i])
    wout ← SubsetSum(X, i-1, T)
    return with OR wout
```

We are given a set of n positive integers $X = \{x_{1,} x_{2,} \ldots, x_n\}$ and a target integer value T. We want to find a subset $\mathrm{Y} \subseteq X$ such that the sum of the elements $\sum_{x_i \in Y} x_i = T$.

Our problem: For a given T and X, does such a Y exist?

Subset Sum Recap

What are our subproblems?

At an arbitrary iteration $1 \leq i \leq n+1$ and $t \leq T$

$$SS(i,t) = \begin{cases} True & if\ t = 0 \\ False & if\ i > n \\ SubSum(i+1,t) & if\ t < X[i] \\ SubSum(i+1,t)\ \vee SubSum(i+1,t-X[i]) & otherwise \end{cases}$$

What data structure can we use for memoization?

We can fill a 2 dimensional array with the following dimensions:
$S[1..n+1, 0..T] = SubSum(i,t)$

Which subproblems depend on each other, and what evaluation order does this imply?

$SubSum(i,t)$ can depend on $SubSum(i+1,t)$ and $SubSum(i+1,t-X[i])$. So we can start at the bottom of the table and work up.

What are the space/time requirements?

Space: $O(nT)$
Time: $O(nT)$

```
FastSubsetSum(X[1..n], T):
  S[n+1,0] ← True
  for t ← 1 to T:
    S[n+1,t] ← False
  for i← n down to 1:
    S[i,0] ← True
    for t ← 1 to X[i]−1:
      S[i,t] ← S[i+1,t]
    for t ← X[i] to T:
      S[i,t] ← S[i+1,t] ∨ S[i+1,t−X[i]]
  return S[1,T]
```

Is FastSubsetSum *always* faster than the recursive version?

No! If $T \gg 2^n$, the recursive version is actually faster!

Subset Sum Recap

What are our subproblems?

At an arbitrary iteration $1 \leq i \leq n+1$ and $t \leq T$

$$SS(i,t) = \begin{cases} True & if\ t = 0 \\ False & if\ i > n \\ SubSum(i+1,t) & if\ t < X[i] \\ SubSum(i+1,t) \vee SubSum(i+1,t-X[i]) & otherwise \end{cases}$$

What data structure can we use for memoization?

We can fill a 2 dimensional array with the following dimensions: $S[1..n+1, 0..T] = SubSum(i,t)$

Which subproblems depend on each other, and what evaluation order does this imply?

$SubSum(i,t)$ can depend on $SubSum(i+1,t)$ and $SubSum(i+1,t-X[i])$. So we can start at the bottom of the table and work up.

What are the space/time requirements?

Space: $O(nT)$
Time: $O(nT)$

```
FastSubsetSum(X[1..n], T):
  S[n+1,0] ← True
  for t ← 1 to T:
    S[n+1,t] ← False
  for i ← n down to 1:
    S[i,0] ← True
    for t ← 1 to X[i]−1:
      S[i,t] ← S[i+1,t]
    for t ← X[i] to T:
      S[i,t] ← S[i+1,t] ∨ S[i+1,t−X[i]]
  return S[1,T]
```

Today: We will reformulate this problem in terms of an *optimal solution* $\mathcal{O}$!

This is where the $Opt(i,j)$ notation in the homework assignment came from.

Subset Sum $Opt(i, t)$ formulation

We are given a set of n positive integers $X = \{x_{1,} x_{2,} \ldots, x_n\}$ and a target integer value T. We want to find a subset $\mathrm{Y} \subseteq X$ such that the sum of the elements $\sum_{x_i \in Y} x_i = T$.

Our problem: For a given T and X, does such a Y exist?

Subset Sum $Opt(i, t)$ formulation

We are given a set of n positive integers $X = \{x_1, x_2, \ldots, x_n\}$ and a target integer value T. We want to find a subset $Y \subseteq X$ such that the sum of the elements $\sum_{x_i \in Y} x_i = T$.

Our problem: For a given T and X, does such a Y exist?

Consider an optimal solution $\mathcal{O}$

- We don't know what the solution is yet, or if it even exists, but we can define our problem in terms of it.

Subset Sum $Opt(i, t)$ formulation

We are given a set of n positive integers $X = \{x_{1,} x_{2,} \dots, x_n\}$ and a target integer value T. We want to find a subset $\mathrm{Y} \subseteq X$ such that the sum of the elements $\sum_{x_i \in Y} x_i = T$.

Our problem: For a given T and X, does such a Y exist?

Consider an optimal solution $\mathcal{O}$

- We don't know what the solution is yet, or if it even exists, but we can define our problem in terms of it.

We know we need to solve the problem for intermediate values of i and t.

Subset Sum $Opt(i, t)$ formulation

We are given a set of n positive integers $X = \{x_{1,} x_{2,} \dots, x_n\}$ and a target integer value T. We want to find a subset $\mathrm{Y} \subseteq X$ such that the sum of the elements $\sum_{x_i \in Y} x_i = T$.

Our problem: For a given T and X, does such a Y exist?

Consider an optimal solution $\mathcal{O}$

- We don't know what the solution is yet, or if it even exists, but we can define our problem in terms of it.

We know we need to solve the problem for intermediate values of i and t.

We can define an optimal solution for a subproblem as:

$$Opt(i, t) = \max_{S} \sum_{j \in S} X[j]$$

Where

- i represents the element under consideration
- t represents a subset weight $t \leq T$ and
- we are taking the maximum over subsets that satisfy $\sum_{j \in S} X[j] \leq t$

Subset Sum $Opt(i,t)$ formulation

We are given a set of n positive integers $X = \{x_{1,} x_{2,} \dots, x_n\}$ and a target integer value T. We want to find a subset $\mathrm{Y} \subseteq X$ such that the sum of the elements $\sum_{x_i \in Y} x_i = T$.

Our problem: For a given T and X, does such a Y exist?

Remember our old friend the T/F table…

T	T	T	T
T	F	T	T
T	F	F	T
T	F	F	F

Consider an optimal solution $\mathcal{O}$

- We don't know what the solution is yet, or if it even exists, but we can define our problem in terms of it.

We know we need to solve the problem for intermediate values of i and t.

We can define an optimal solution for a subproblem as:

$$Opt(i,t) = \max_{S} \sum_{j \in S} X[j]$$

Where

- i represents the element under consideration
- t represents a subset weight $t \leq T$ and
- we are taking the maximum over subsets that satisfy $\sum_{j \in S} X[j] \leq t$

Subset Sum $Opt(i,t)$ formulation

We are given a set of n positive integers $X = \{x_{1,} x_{2,} \ldots, x_n\}$ and a target integer value T. We want to find a subset $\mathrm{Y} \subseteq X$ such that the sum of the elements $\sum_{x_i \in Y} x_i = T$.

Our problem: For a given T and X, does such a Y exist?

Remember our old friend the T/F table…

0	0	0	0
0	1	1	1
0	1	2	3
0	1	2	3

…it will now be an $Opt(i,t)$ table!

Consider an optimal solution $\mathcal{O}$

- We don't know what the solution is yet, or if it even exists, but we can define our problem in terms of it.

We know we need to solve the problem for intermediate values of i and t.

We can define an optimal solution for a subproblem as:

$$Opt(i,t) = \max_{S} \sum_{j \in S} X[j]$$

Where

- i represents the element under consideration
- t represents a subset weight $t \leq T$ and
- we are taking the maximum over subsets that satisfy $\sum_{j \in S} X[j] \leq t$

Subset Sum $Opt(i, t)$ formulation

We are given a set of n positive integers $X = \{x_{1,} x_{2,} \ldots, x_n\}$ and a target integer value T. We want to find a subset $\mathrm{Y} \subseteq X$ such that the sum of the elements $\sum_{x_i \in Y} x_i = T$.

Our problem: For a given T and X, does such a Y exist?

What is our recurrence for $Opt(i, t)$?

$$Opt(i, t) = \max_S \sum_{j \in S} X[j]$$

Where

- i represents the element under consideration
- t represents a subset weight t $\leq T$ and
- we are taking the maximum over subsets that satisfy $\sum_{j \in S} X[j] \leq t$

Subset Sum $Opt(i,t)$ formulation

We are given a set of n positive integers $X = \{x_{1,} x_{2,} \ldots, x_n\}$ and a target integer value T. We want to find a subset $\mathrm{Y} \subseteq X$ such that the sum of the elements $\sum_{x_i \in Y} x_i = T$.

Our problem: For a given T and X, does such a Y exist?

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} \\ \end{cases}$$

$$Opt(i,t) = \max_S \sum_{j \in S} X[j]$$

Where

- i represents the element under consideration
- t represents a subset weight t $\leq T$ and
- we are taking the maximum over subsets that satisfy $\sum_{j \in S} X[j] \leq t$

For each item in the set, is $\mathrm{X[i]} \in \mathcal{O}$?:

Subset Sum $Opt(i,t)$ formulation

We are given a set of n positive integers $X = \{x_{1,} x_{2,} \dots, x_n\}$ and a target integer value T. We want to find a subset $\mathrm{Y} \subseteq X$ such that the sum of the elements $\sum_{x_i \in Y} x_i = T$.

Our problem: For a given T and X, does such a Y exist?

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ & \end{cases}$$

$$Opt(i,t) = \max_S \sum_{j \in S} X[j]$$

Where

- i represents the element under consideration
- t represents a subset weight t $\leq T$ and
- we are taking the maximum over subsets that satisfy $\sum_{j \in S} X[j] \leq t$

For each item in the set, is $\mathrm{X[i]} \in \mathcal{O}$?:

- If $X[i] \notin \mathcal{O}$, then
  $Opt(i,T) = Opt(i-1,T)$

Subset Sum $Opt(i, t)$ formulation

We are given a set of n positive integers $X = \{x_{1,} x_{2,} \dots, x_n\}$ and a target integer value T. We want to find a subset $\text{Y} \subseteq X$ such that the sum of the elements $\sum_{x_i \in Y} x_i = T$.

Our problem: For a given T and X, does such a Y exist?

What is our recurrence for $Opt(i, t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1,t-X[i])) & otherwise \end{cases}$$

$$Opt(i,t) = \max_S \sum_{j \in S} X[j]$$

Where

- i represents the element under consideration
- t represents a subset weight t $\leq T$ and
- we are taking the maximum over subsets that satisfy $\sum_{j \in S} X[j] \leq t$

For each item in the set, is $\text{X[i]} \in \mathcal{O}$?:

- If $X[i] \notin \mathcal{O}$, then

  $Opt(i,T) = Opt(i-1,T)$
- If $\text{X[i]} \in \mathcal{O}$, then

$Opt(i,T) = X[i] + Opt(i-1,T-X[i])$

Subset Sum $Opt(i,t)$ formulation

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1, t - X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]      // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$$Opt(i,t) = \max_S \sum_{j \in S} X[j]$$

Where

- i represents the element under consideration
- t represents a subset weight t $\leq T$ and
- we are taking the maximum over subsets that satisfy $\sum_{j \in S} X[j] \leq t$

For each item in the set, is X[i] ∈ $\mathcal{O}$?:

- If $X[i] \notin \mathcal{O}$, then
  $Opt(i,T) = Opt(i-1,T)$
- If X[i] ∈ $\mathcal{O}$, then

$Opt(i,T) = X[i] + Opt(i-1, T - X[i])$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1,t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]        // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

Previous Solution

$S(1{,}0)$	$S(1{,}1)$	$S(1{,}2)$	$S(1{,}3)$
$S(2{,}0)$	$S(2{,}1)$	$S(2{,}2)$	$S(2{,}3)$
$S(3{,}0)$	$S(3{,}1)$	$S(3{,}2)$	$S(3{,}3)$
$S(4{,}0)$	$S(4{,}1)$	$S(4{,}2)$	$S(4{,}3)$

n (rows), T (columns)

Main differences:

- Instead of $S(i,t)$ (boolean) we have replaced with $Opt(i,t)$ (integer)
- Instead of $n+1 \rightarrow 1$, we are filling from $0 \rightarrow n$

New Solution

$Opt(0{,}0)$	$Opt(0{,}1)$	$Opt(0{,}2)$	$Opt(0{,}3)$
$Opt(1{,}0)$	$Opt(1{,}1)$	$Opt(1{,}2)$	$Opt(2{,}3)$
$Opt(2{,}0)$	$Opt(2{,}1)$	$Opt(2{,}2)$	$Opt(3{,}3)$
$Opt(3{,}0)$	$Opt(3{,}1)$	$Opt(3{,}2)$	$Opt(3{,}3)$

n (rows), T (columns)

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1, t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]      // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

Previous Solution

$S(1,0)$	$S(1,1)$	$S(1,2)$	$S(1,3)$
$S(2,0)$	$S(2,1)$	$S(2,2)$	$S(2,3)$
$S(3,0)$	$S(3,1)$	$S(3,2)$	$S(3,3)$
$S(4,0)$	$S(4,1)$	$S(4,2)$	$S(4,3)$

n (rows), T (columns)

Main differences:

- Instead of $S(i,t)$ (boolean) we have replaced with $Opt(i,t)$ (integer)
- Instead of $n+1 \rightarrow 1$, we are filling from $0 \rightarrow n$

New Solution

$Opt(0,0)$	$Opt(0,1)$	$Opt(0,2)$	$Opt(0,3)$
$Opt(1,0)$	$Opt(1,1)$	$Opt(1,2)$	$Opt(2,3)$
$Opt(2,0)$	$Opt(2,1)$	$Opt(2,2)$	$Opt(3,3)$
$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

n (rows), T (columns)

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1, t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]      // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

n

$Opt(0,0)$	$Opt(0,1)$	$Opt(0,2)$	$Opt(0,3)$
$Opt(1,0)$	$Opt(1,1)$	$Opt(1,2)$	$Opt(2,3)$
$Opt(2,0)$	$Opt(2,1)$	$Opt(2,2)$	$Opt(3,3)$
$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

T

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & \text{if } t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1, t - X[i])) & \text{otherwise} \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]     // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$X = [1,2,3], T = 3$

0	0	0	0
$Opt(1,0)$	$Opt(1,1)$	$Opt(1,2)$	$Opt(2,3)$
$Opt(2,0)$	$Opt(2,1)$	$Opt(2,2)$	$Opt(3,3)$
$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1, t - X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]      // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

	t=0			
	0	0	0	0
i=1	$Opt(1,0)$	$Opt(1,1)$	$Opt(1,2)$	$Opt(2,3)$
	$Opt(2,0)$	$Opt(2,1)$	$Opt(2,2)$	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1, t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i−1,t]      // Exclude item i
      Else:
        S[i,t] ← max(S[i− 1,t], X[i] + S[i−1,t −X[i]]
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

t=0

	0	0	0	0
i=1	0	$Opt(1,1)$	$Opt(1,2)$	$Opt(2,3)$
	$Opt(2,0)$	$Opt(2,1)$	$Opt(2,2)$	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1,t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]      // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

t=1

	0	0	0	0
i=1	0	$Opt(1,1)$	$Opt(1,2)$	$Opt(2,3)$
	$Opt(2,0)$	$Opt(2,1)$	$Opt(2,2)$	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1,t-X[i])) & otherwise \end{cases}$$

$$X = [1,2,3], T = 3$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]     // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

t=1

	0	0	0	0
i=1	0	$Opt(1,1)$	$Opt(1,2)$	$Opt(2,3)$
	$Opt(2,0)$	$Opt(2,1)$	$Opt(2,2)$	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

$$\max(S[0,1] = 0, 1 + S[0, 1-1 = 0] = 1)$$

$$\max(0, 1 + S[0,0] = 1) = 1$$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1, t - X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]     // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]])
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

t=1

	0	0	0	0
i=1	0	1	$Opt(1,2)$	$Opt(2,3)$
	$Opt(2,0)$	$Opt(2,1)$	$Opt(2,2)$	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

$$\max(S[0,1] = 0, 1 + S[0, 1-1=0] = 1)$$

$$\max(0, 1 + S[0,0] = 1) = 1$$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1, t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]      // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$X = [1,2,3], T = 3$

t=2

	0	0	0	0
i=1	0	1	$Opt(1,2)$	$Opt(2,3)$
	$Opt(2,0)$	$Opt(2,1)$	$Opt(2,2)$	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + \ Opt(i-1, t - X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]     // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

t=2

	0	0	0	0
i=1	0	1	$Opt(1{,}2)$	$Opt(2{,}3)$
	$Opt(2{,}0)$	$Opt(2{,}1)$	$Opt(2{,}2)$	$Opt(3{,}3)$
	$Opt(3{,}0)$	$Opt(3{,}1)$	$Opt(3{,}2)$	$Opt(3{,}3)$

$$\max(S[0,2] = 0, 1 + S[0, 2-1 = 1] = 1)$$

$$\max(0, 1 + 0 = 1) = 1$$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1,t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]      // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]]
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

t=2

	0	0	0	0
i=1	0	1	1	$Opt(2,3)$
	$Opt(2,0)$	$Opt(2,1)$	$Opt(2,2)$	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

$$\max(S[0,2] = 0, 1 + S[0, 2-1=1] = 1)$$

$$\max(0, 1+0=1) = 1$$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1,t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]     // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

				t=3
	0	0	0	0
i=1	0	1	1	$Opt(2,3)$
	$Opt(2,0)$	$Opt(2,1)$	$Opt(2,2)$	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + \ Opt(i-1,t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]       // Exclude item i
      Else:
        S[i,t] ← max(S[i −  1,t],  X[i] + S[i − 1,t  − X[i]]
  return S[n,T]
```

$$X\ = [1,2,3], T = 3$$

				t=3
	0	0	0	0
i=1	0	1	1	1
	$Opt(2,0)$	$Opt(2,1)$	$Opt(2,2)$	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1, t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]      // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$X = [1,2,3], T = 3$

	t=0			
	0	0	0	0
	0	1	1	1
i=2	0	$Opt(2,1)$	$Opt(2,2)$	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1,t-X[i])) & otherwise \end{cases}$$

$$X = [1,2,3], T = 3$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i−1,t]      // Exclude item i
      Else:
        S[i,t] ← max(S[i− 1,t], X[i] + S[i−1,t −X[i]]
  return S[n,T]
```

t=1

	0	0	0	0
	0	1	1	1
i=2	0	1	$Opt(2,2)$	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

Subset Sum $Opt(i, t)$ example

What is our recurrence for $Opt(i, t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + \ Opt(i-1,t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]     // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

t=2

	0	0	0	0
	0	1	1	1
i=2	0	1	$Opt(2,2)$	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

$$\max(S[1,2] = 1, 2 + S[1, 2-2 = 0] = 2)$$

$$\max(1, 2) = 2$$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1,t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]     // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

t=2

	0	0	0	0
	0	1	1	1
i=2	0	1	2	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

$$\max(S[1,2] = 1, 2 + S[1, 2-2=0] = 2)$$

$$\max(1,2) = 2$$

Subset Sum $Opt(i, t)$ example

What is our recurrence for $Opt(i, t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1,t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]     // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

				t=3
	0	0	0	0
	0	1	1	1
i=2	0	1	2	$Opt(3,3)$
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

$$\max(S[1,3] = 1, 2 + S[1, 3 - 2 = 1] = 3)$$

$$\max(1, 3) = 3$$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + \ Opt(i-1,t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]     // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]])
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

				t=3
	0	0	0	0
	0	1	1	1
i=2	0	1	2	3
	$Opt(3,0)$	$Opt(3,1)$	$Opt(3,2)$	$Opt(3,3)$

$$\max(S[1,3] = 1, 2 + S[1, 3-2 = 1] = 3)$$

$$\max(1,3) = 3$$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1, t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]      // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

	0	0	0	0
	0	1	1	1
	0	1	2	3
i=3	0	1	2	$Opt(3,3)$

Subset Sum $Opt(i,t)$ example

What is our recurrence for $Opt(i,t)$?

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), X[i] + Opt(i-1,t-X[i])) & otherwise \end{cases}$$

```
OptSubsetSum(X[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]      // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], X[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

$$X = [1,2,3], T = 3$$

	0	0	0	0
	0	1	1	1
	0	1	2	3
i=3	0	1	2	3

$$\max(S[2,3] = 3, 3 + S[2, 3-3 = 0] = 3)$$

$$\max(3,3) = 3$$

Subset Sum Wrap (take 2)

Two different solutions get us to the same result

- First solution used boolean $SubsetSum(i, t)$ to indicate whether a solution existed for values $X[1..i]$ and sum $t \leq T$.
    - If $S[1, T] = True$, we know there is a soultion.

Subset Sum Wrap (take 2)

Two different solutions get us to the same result

- First solution used boolean $SubsetSum(i, t)$ to indicate whether a solution existed for values $X[1..i]$ and sum $t \leq T$.
  - If $S[1, T] = True$, we know there is a soultion.
- Second solution found integer solutions $Opt(i, t)$ for the same subproblems.
  - If $S[n, T] = T$ we know there is a solution.

Subset Sum Wrap (take 2)

Two different solutions get us to the same result

- First solution used boolean $SubsetSum(i, t)$ to indicate whether a solution existed for values $X[1..i]$ and sum $t \leq T$.
  - If $S[1, T] = True$, we know there is a soultion.
- Second solution found integer solutions $Opt(i, t)$ for the same subproblems.
  - If $S[n, T] = T$ we know there is a solution.

Many problems can be viewed from multiple directions in this way

- If both are possible, choosing is a matter of preference
  - Erickson tends to use first method
  - Tardos & Kleinberg use second

Subset Sum Wrap (take 2)

Two different solutions get us to the same result

- First solution used boolean $SubsetSum(i, t)$ to indicate whether a solution existed for values $X[1..i]$ and sum $t \leq T$.
  - If $S[1, T] = True$, we know there is a soultion.
- Second solution found integer solutions $Opt(i, t)$ for the same subproblems.
  - If $S[n, T] = T$ we know there is a solution.

Many problems can be viewed from multiple directions in this way

- If both are possible, choosing is a matter of preference
  - Erickson tends to use first method
  - Tardos & Kleinberg use second
- Either are acceptable in the class (unless otherwise specified) as long as the solution is correct

Extending Subset Sum to the Knapsack Problem

Knapsack Problem

Subset Sum is a special case of a more general problem called the *knapsack problem*

In the knapsack problem, items have both a *weight* $X[i]$ and a *value* $v[i]$

Knapsack Problem

Subset Sum is a special case of a more general problem called the *knapsack problem*

In the knapsack problem, items have both a *weight* $X[i]$ and a *value* $v[i]$

Imagine you are preparing for a long backpacking trip. You have a strict limit W on the amount of weight you can carry in your pack and you want to choose a subset S of items to bring that maximizes the total value $\mathrm{V} = \sum_{j \in S} v[j]$

Knapsack Problem

Subset Sum is a special case of a more general problem called the *knapsack problem*

In the knapsack problem, items have both a *weight* $X[i]$ and a *value* $v[i]$

Imagine you are preparing for a long backpacking trip. You have a strict limit W on the amount of weight you can carry in your pack and you want to choose a subset S of items to bring that maximizes the total value $\mathrm{V} = \sum_{j \in S} v[j]$



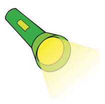


Knapsack Problem

Subset Sum is a special case of a more general problem called the *knapsack problem*

In the knapsack problem, items have both a *weight* $X[i]$ and a *value* $v[i]$

Imagine you are preparing for a long backpacking trip. You have a strict limit W on the amount of weight you can carry in your pack and you want to choose a subset S of items to bring that maximizes the total value $\mathrm{V} = \sum_{j \in S} v[j]$




Knapsack Problem

We want a procedure that will prioritize packing *only* what we need to survive!

For example, in most cases it will probably leave our industrial strength hair dryer behind…

Knapsack Problem

We want a procedure that will prioritize packing *only* what we need to survive!

For example, in most cases it will probably leave our industrial strength hair dryer behind…

Princess Vespa's Hairdryer from Spaceballs

Knapsack Problem

We want to find a subset of items S that maximizes $\sum_{j \in S} v[j]$ with the same constraint that $\sum_{j \in S} X[j] \leq T$.

Knapsack Problem

We want to find a subset of items S that maximizes $\sum_{j \in S} v[j]$ with the same constraint that $\sum_{j \in S} X[j] \leq T$.

Before, we were trying to maximize the *weight* of the subset with no other constraint. We assumed that our summation was over sets S that satisfied the weight constraint.

Knapsack Problem

We want to find a subset of items S that maximizes $\sum_{j \in S} v[j]$ with the same constraint that $\sum_{j \in S} X[j] \leq T$.

Before, we were trying to maximize the *weight* of the subset with no other constraint. We assumed that our summation was over sets S that satisfied the weight constraint.

Keeping that assumption, we already know the subsets S are constrained to those that satisfy the weight constraint, so we can make a very minor modification to solve the knapsack problem!

Same question as before: for each item in the set, is $\mathrm{X}[\mathrm{i}] \in \mathcal{O}$?:

Knapsack Problem

We want to find a subset of items S that maximizes $\sum_{j \in S} v[j]$ with the same constraint that $\sum_{j \in S} X[j] \leq T$.

Before, we were trying to maximize the *weight* of the subset with no other constraint. We assumed that our summation was over sets S that satisfied the weight constraint.

Keeping that assumption, we already know the subsets S are constrained to those that satisfy the weight constraint, so we can make a very minor modification to solve the knapsack problem!

Same question as before: for each item in the set, is $\mathrm{X[i]} \in \mathcal{O}$?:

- If $X[i] \notin \mathcal{O}$, then
  - $Opt(i, T) = Opt(i-1, T)$

$$Opt(i,t) = \begin{cases} Opt(i-1, t) & if\ t < X[i] \\ & \end{cases}$$

Knapsack Problem

We want to find a subset of items S that maximizes $\sum_{j \in S} v[j]$ with the same constraint that $\sum_{j \in S} X[j] \leq T$.

Before, we were trying to maximize the *weight* of the subset with no other constraint. We assumed that our summation was over sets S that satisfied the weight constraint.

Keeping that assumption, we already know the subsets S are constrained to those that satisfy the weight constraint, so we can make a very minor modification to solve the knapsack problem!

Same question as before: for each item in the set, is $\mathrm{X[i]} \in \mathcal{O}$?:

- If $X[i] \notin \mathcal{O}$, then
  - $Opt(i, T) = Opt(i-1, T)$
- If $\mathrm{X[i]} \in \mathcal{O}$, then
  - $Opt(i, T) = v[i] + \ Opt(i-1, T - X[i])$

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), v[i] + Opt(i-1, t - X[i])) & otherwise \end{cases}$$

Knapsack Problem

We want to find a subset of items S that maximizes $\sum_{j \in S} v[j]$ with the same constraint that $\sum_{j \in S} X[j] \leq T$.

Before, we were trying to maximize the *weight* of the subset with no other constraint. We assumed that our summation was over sets S that satisfied the weight constraint.

Keeping that assumption, we already know the subsets S are constrained to those that satisfy the weight constraint, so we can make a very minor modification to solve the knapsack problem!

Same question as before: for each item in the set, is $\mathrm{X[i]} \in \mathcal{O}$?:

- If $X[i] \notin \mathcal{O}$, then
  - $Opt(i, T) = Opt(i-1, T)$
- If $\mathrm{X[i]} \in \mathcal{O}$, then
  - $Opt(i, T) = v[i] + \ Opt(i-1, T - X[i])$

$$Opt(i, t) = \begin{cases} Opt(i-1, t) & if\ t < X[i] \\ \max(Opt(i-1, t), v[i] + Opt(i-1, t - X[i])) & otherwise \end{cases}$$

We can use the same algorithm to solve this!

Knapsack Problem Wrap

We want to find a subset of items S that maximizes $\sum_{j \in S} v[j]$ with the same constraint that $\sum_{j \in S} X[j] \leq T$.

Before, we were trying to maximize the *weight* of the subset with no other constraint. We assumed that our summation was over sets S that satisfied the weight constraint.

Keeping that assumption, we already know the subsets S are constrained to those that satisfy the weight constraint, so we can make a very minor modification to solve the knapsack problem!

$$Opt(i,t) = \begin{cases} Opt(i-1,t) & if\ t < X[i] \\ \max(Opt(i-1,t), v[i] + Opt(i-1, t - X[i])) & otherwise \end{cases}$$

We can use the same algorithm to solve this!

```
OptKnapsack(X[1..n], v[1..n], T):
  for t ← 0 to T:
    S[0,t] ← 0
  for i ← 1 to n:
    for t ← 0 to T:
      if t < X[i]:
        S[i,t] ← S[i − 1,t]      // Exclude item i
      Else:
        S[i,t] ← max(S[i − 1,t], v[i] + S[i − 1,t − X[i]]
  return S[n,T]
```

Wrap-up of today

Subset Sum can be solved in (at least) two ways using dynamic programming

Knapsack Problem is a more general version of Subset Sum that adds a notion of *value* to each element

Knapsack can be solved in almost the exact same way as Subset Sum, just maximizing value rather than weight

This week

Tomorrow:

- Find a dynamic programming solution for Edit Distance
- Wrap up dynamic programming
- Introduce basic features of graphs to get us started on graph algorithms

Wednesday:

- First half-ish: Continue with graph algorithms
- Second half-ish: Answers to student-submitted questions (form to be sent out this evening)

Thursday:

- No class while midterm exam is out