Dynamic Programming Recipe

- Define a set of problems, such that
  - base case - easy to solve
  - final case - matches (closely) the final problem we want to solve.
- Write it as a recursion: Solve bigger problem in terms of the smaller problems. (Should be a DAG on the problem instances!)
- Compute the problems on the DAG in the linearized order!

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Find longest increasing sequence of numbers.

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Find longest increasing sequence of numbers.

7 3 5 6 1 2 9 4 8

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Find longest increasing sequence of numbers.

7 3 5 6 1 2 9 4 8

Greedy??

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Find longest increasing sequence of numbers.

7 3 5 6 1 2 9 4 8

Greedy??

(A) Choose first number, delete higher, continue.

(B) Choose lowest number, continue with rest of sequence.

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Find longest increasing sequence of numbers.

7 3 5 6 1 2 9 4 8

Greedy??

(A) Choose first number, delete higher, continue.

(B) Choose lowest number, continue with rest of sequence.

(A) seems bad.

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Find longest increasing sequence of numbers.

7 3 5 6 1 2 9 4 8

Greedy??

(A) Choose first number, delete higher, continue.

(B) Choose lowest number, continue with rest of sequence.

(A) seems bad.
Try method (B):

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Find longest increasing sequence of numbers.

7 3 5 6 1 2 9 4 8

Greedy??

(A) Choose first number, delete higher, continue.

(B) Choose lowest number, continue with rest of sequence.

(A) seems bad.
Try method (B): 1

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Find longest increasing sequence of numbers.

7 3 5 6 1 2 9 4 8

Greedy??

(A) Choose first number, delete higher, continue.

(B) Choose lowest number, continue with rest of sequence.

(A) seems bad.
Try method (B): 1 2

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Find longest increasing sequence of numbers.

$7 \quad 3 \quad 5 \quad 6 \quad 1 \quad 2 \quad 9 \quad 4 \quad 8$

Greedy??

(A) Choose first number, delete higher, continue.

(B) Choose lowest number, continue with rest of sequence.

(A) seems bad.
Try method (B): 1 2 4

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Find longest increasing sequence of numbers.

7 3 5 6 1 2 9 4 8

Greedy??

(A) Choose first number, delete higher, continue.

(B) Choose lowest number, continue with rest of sequence.

(A) seems bad.
Try method (B): 1 2 4 8

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Find longest increasing sequence of numbers.

7 3 5 6 1 2 9 4 8

Greedy??

(A) Choose first number, delete higher, continue.

(B) Choose lowest number, continue with rest of sequence.

(A) seems bad.
Try method (B): 1 2 4 8

7 3 5 6 8 2 9 4 1

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Find longest increasing sequence of numbers.

7 3 5 6 1 $\rightarrow$ 2 $\rightarrow$ 9 4 $\rightarrow$ 8

Greedy??

(A) Choose first number, delete higher, continue.

(B) Choose lowest number, continue with rest of sequence.

(A) seems bad.
Try method (B): 1 2 4 8

7 3 5 6 8 2 9 4 1

Not good!

Longest Increasing Subsequence

Given sequence of numbers: $a_1, a_2, \ldots, a_n$.

Find longest increasing sequence of numbers.

7 3 5 6 1 2 9 4 8

Greedy??

(A) Choose first number, delete higher, continue.

(B) Choose lowest number, continue with rest of sequence.

(A) seems bad.
Try method (B): 1 2 4 8

7 3 5 6 8 2 9 4 1

Not good!

What to do?

Dynamic Programming Solution.

$L(i)$ is length of longest increasing subsequence ending at position i.

Dynamic Programming Solution.

$L(i)$ is length of longest increasing subsequence ending at position i.

Do I know $L(1)$? Is $L(n)$ good enough for the answer?

Dynamic Programming Solution.

$L(i)$ is length of longest increasing subsequence ending at position i.

Do I know $L(1)$? Is $L(n)$ good enough for the answer?($\max_j L(j)$)

Dynamic Programming Solution.

$L(i)$ is length of longest increasing subsequence ending at position i.

Do I know $L(1)$? Is $L(n)$ good enough for the answer?($\max_j L(j)$)

Recursion

$$L(j) = \max_{j<i \,\wedge\, a[j]<a[i]} \{L(j) + 1\}$$

Dynamic Programming Solution.

$L(i)$ is length of longest increasing subsequence ending at position i.

Do I know $L(1)$? Is $L(n)$ good enough for the answer?($\max_j L(j)$)

Recursion

$$L(j) = \max_{j<i \,\wedge\, a[j]<a[i]} \{L(j) + 1\}$$

Think of the DAG?

Dynamic Programming Solution.

$L(i)$ is length of longest increasing subsequence ending at position i.

Do I know $L(1)$? Is $L(n)$ good enough for the answer?($\max_j L(j)$)

Recursion

$$L(j) = \max_{j<i \,\wedge\, a[j]<a[i]} \{L(j)+1\}$$

Think of the DAG? **For** $i = 1, 2, \ldots, n$

$L(i) = 1$

For $j = 1, \ldots, i-1$

if $a[j] < a[i]$

$L(i) = \max(L(i), L(j)+1)$

Dynamic Programming Solution.

$L(i)$ is length of longest increasing subsequence ending at position i.

Do I know $L(1)$? Is $L(n)$ good enough for the answer?($\max_j L(j)$)

Recursion

$$L(j) = \max_{j<i \,\wedge\, a[j]<a[i]} \{L(j) + 1\}$$

Think of the DAG? **For** $i = 1, 2, \ldots, n$

$L(i) = 1$

For $j = 1, \ldots, i-1$

if $a[j] < a[i]$

$L(i) = \max(L(i), L(j) + 1)$

$O(n^2)$ time

Dynamic Programming Solution.

$L(i)$ is length of longest increasing subsequence ending at position i.

Do I know $L(1)$? Is $L(n)$ good enough for the answer?($\max_j L(j)$)

Recursion

$$L(j) = \max_{j<i \,\wedge\, a[j]<a[i]} \{L(j)+1\}$$

Think of the DAG? **For** $i = 1, 2, \ldots, n$

$L(i) = 1$

For $j = 1, \ldots, i-1$

if $a[j] < a[i]$

$L(i) = \max(L(i), L(j)+1)$

$O(n^2)$ time $O(n)$ space.

Dynamic Programming Solution.

$L(i)$ is length of longest increasing subsequence ending at position i.

Do I know $L(1)$? Is $L(n)$ good enough for the answer?($\max_j L(j)$)

Recursion

$$L(j) = \max_{j<i \wedge a[j]<a[i]} \{L(j)+1\}$$

Think of the DAG? **For** $i = 1, 2, \ldots, n$

$L(i) = 1$

For $j = 1, \ldots, i-1$

if $a[j] < a[i]$

$L(i) = \max(L(i), L(j)+1)$

$O(n^2)$ time $O(n)$ space. *Find* longest subsequence?

Dynamic Programming Solution.

$L(i)$ is length of longest increasing subsequence ending at position i.

Do I know $L(1)$? Is $L(n)$ good enough for the answer?($\max_j L(j)$)

Recursion

$$L(j) = \max_{j<i \wedge a[j]<a[i]} \{L(j)+1\}$$

Think of the DAG? **For** $i = 1, 2, \ldots, n$

$L(i) = 1$

For $j = 1, \ldots, i-1$

if $a[j] < a[i]$

$L(i) = \max(L(i), L(j)+1)$

$O(n^2)$ time $O(n)$ space. *Find* longest subsequence?(maintain pointers)

Dynamic Programming Solution.

$L(i)$ is length of longest increasing subsequence ending at position i.

Do I know $L(1)$? Is $L(n)$ good enough for the answer?($\max_j L(j)$)

Recursion

$$L(j) = \max_{j<i \,\wedge\, a[j]<a[i]} \{L(j)+1\}$$

Think of the DAG? **For** $i = 1, 2, \ldots, n$

$L(i) = 1$

For $j = 1, \ldots, i-1$

if $a[j] < a[i]$

$L(i) = \max(L(i), L(j)+1)$

$O(n^2)$ time $O(n)$ space. *Find* longest subsequence?(maintain pointers)

For $i = 1, 2, \ldots, n$

$L(i) = 1$, $\text{prev}(i) = i$

For $j = 1, \ldots, i-1$

if $a[j] < a[i]$

if $L(j)+1 > L(i)$

$L(i) = L(j)+1$; $\text{prev}(i) = j$

Dynamic Programming Solution.

$L(i)$ is length of longest increasing subsequence ending at position i.

Do I know $L(1)$? Is $L(n)$ good enough for the answer?($\max_j L(j)$)

Recursion

$$L(j) = \max_{j<i \,\wedge\, a[j]<a[i]} \{L(j)+1\}$$

Think of the DAG? **For** $i = 1,2,\ldots,n$

$L(i) = 1$

For $j = 1,\ldots,i-1$

if $a[j] < a[i]$

$L(i) = \max(L(i), L(j)+1)$

$O(n^2)$ time $O(n)$ space. *Find* longest subsequence?(maintain pointers)

For $i = 1,2,\ldots,n$

$L(i) = 1$, $\text{prev}(i) = i$

For $j = 1,\ldots,i-1$

if $a[j] < a[i]$

if $L(j)+1 > L(i)$

$L(i) = L(j)+1$; $\text{prev}(i) = j$

Chase prev(j) pointers backwards to construct path!

Dynamic Programming Solution.

$L(i)$ is length of longest increasing subsequence ending at position i.

Do I know $L(1)$? Is $L(n)$ good enough for the answer?($\max_j L(j)$)

Recursion

$$L(j) = \max_{j<i \,\wedge\, a[j]<a[i]} \{L(j)+1\}$$

Think of the DAG? **For** $i = 1, 2, \ldots, n$

$L(i) = 1$

For $j = 1, \ldots, i-1$

if $a[j] < a[i]$

$L(i) = \max(L(i), L(j)+1)$

$O(n^2)$ time $O(n)$ space. *Find* longest subsequence?(maintain pointers)

For $i = 1, 2, \ldots, n$

$L(i) = 1$, $\text{prev}(i) = i$

For $j = 1, \ldots, i-1$

if $a[j] < a[i]$

if $L(j)+1 > L(i)$

$L(i) = L(j)+1$; $\text{prev}(i) = j$

Chase prev(j) pointers backwards to construct path! Similar to finding sp tree.

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Recursion $\equiv$ dynamic programming?

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Recursion $\equiv$ dynamic programming?
    sub problem solutions...

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Recursion $\equiv$ dynamic programming?
  sub problem solutions...built from smaller subproblems.

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Recursion $\equiv$ dynamic programming?
    sub problem solutions...built from smaller subproblems.

Recursive instead of iterative?

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Recursion $\equiv$ dynamic programming?
sub problem solutions...built from smaller subproblems.

Recursive instead of iterative?

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Recursion $\equiv$ dynamic programming?
 sub problem solutions...built from smaller subproblems.

Recursive instead of iterative?

```
def L(i):
    val = 1
   for  j = 1,...,n:
        if  a[j] < a[i]:
            if  L(j) + 1 > val:
                val = L(j) + 1
```

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Recursion $\equiv$ dynamic programming?
sub problem solutions...built from smaller subproblems.

Recursive instead of iterative?

```
def L(i):
    val = 1
  for  j = 1,...,n:
        if  a[j] < a[i]:
            if  L(j) + 1 > val:
                val = L(j) + 1
```

Enumerates all paths in "DAG".

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Recursion $\equiv$ dynamic programming?
sub problem solutions...built from smaller subproblems.

Recursive instead of iterative?

def L(i):
  val = 1
  for $j = 1, \ldots, n$:
    if $a[j] < a[i]$:
      if $L(j) + 1 > val$:
        val = L(j) + 1

Enumerates all paths in “DAG”. Exponential time!!

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Recursion $\equiv$ dynamic programming?
sub problem solutions...built from smaller subproblems.

Recursive instead of iterative?

```
def L(i):
  val = 1
  for  j = 1,...,n:
      if  a[j] < a[i]:
          if  L(j) + 1 > val:
              val = L(j) + 1
```

Enumerates all paths in "DAG". Exponential time!!

Memoization.

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Recursion $\equiv$ dynamic programming?
sub problem solutions...built from smaller subproblems.

Recursive instead of iterative?

```
def L(i):
  val = 1
  for j = 1,...,n:
      if a[j] < a[i]:
          if L(j) + 1 > val:
              val = L(j) + 1
```

Enumerates all paths in “DAG”. Exponential time!!

Memoization.
Answer $L(i)$ same each time, so remember, return.

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Recursion $\equiv$ dynamic programming?
    sub problem solutions...built from smaller subproblems.

Recursive instead of iterative?

```
def L(i):
  val = 1
  for j = 1,...,n:
    if a[j] < a[i]:
      if L(j) + 1 > val:
        val = L(j) + 1
```

Enumerates all paths in "DAG". Exponential time!!

Memoization.
    Answer $L(i)$ same each time, so remember, return.

Only n different arguments provided to $L(\cdot)$.

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Recursion $\equiv$ dynamic programming?
    sub problem solutions...built from smaller subproblems.

Recursive instead of iterative?

```
def L(i):
  val = 1
  for j = 1,...,n:
     if a[j] < a[i]:
        if L(j) + 1 > val:
           val = L(j) + 1
```

Enumerates all paths in "DAG". Exponential time!!

Memoization.
    Answer $L(i)$ same each time, so remember, return.

Only n different arguments provided to $L(\cdot)$.
    Each call takes $O(n)$ time the *first* time.

Dynamic Programming and Recursion.

$L(i)$ - longest increasing subsequence ending at i.
Only need $L(j)$ for $j < i$ to find $L(i)$.

Recursion $\equiv$ dynamic programming?
sub problem solutions...built from smaller subproblems.

Recursive instead of iterative?

def L(i):
  val = 1
  for $j = 1, \ldots, n$:
    if $a[j] < a[i]$:
      if $L(j) + 1 > val$:
        val = L(j) + 1

Enumerates all paths in "DAG". Exponential time!!

Memoization.
Answer $L(i)$ same each time, so remember, return.

Only n different arguments provided to $L(\cdot)$.
Each call takes $O(n)$ time the *first* time.

Same as "iterative".

Edit Distance.

Spell Correction.

Edit Distance.

Spell Correction.

"THEARFTER"

Edit Distance.

Spell Correction.

"THEARFTER"

THEARFTER versus THEREAFTER

Edit Distance.

Spell Correction.

"THEARFTER"

THEARFTER versus THEREAFTER

Find "closest" real word!

Edit Distance.

Spell Correction.

"THEARFTER"

THEARFTER versus THEREAFTER

Find "closest" real word!

Given two words, how far apart are they?

Edit Distance.

Spell Correction.

"THEARFTER"

THEARFTER versus THEREAFTER

Find "closest" real word!

Given two words, how far apart are they?

Best alignment.

Edit Distance.

Spell Correction.

"THEARFTER"

THEARFTER versus THEREAFTER

Find "closest" real word!

Given two words, how far apart are they?

Best alignment.

THEAR– –FTER
THE–REAFTER

Edit Distance.

Spell Correction.

“THEARFTER”

THEARFTER versus THEREAFTER

Find “closest” real word!

Given two words, how far apart are they?

Best alignment.

THEAR– –FTER
THE–REAFTER

Cost: 3.

Edit Distance.

Spell Correction.

"THEARFTER"

THEARFTER versus THEREAFTER

Find "closest" real word!

Given two words, how far apart are they?

Best alignment.

THEAR– –FTER
THE–REAFTER

Cost: 3.
Edit distance. "THEARFTER" to "THEREAFTER" uses one deletion, two insertions.

Edit Distance.

Spell Correction.

"THEARFTER"

THEARFTER versus THEREAFTER

Find "closest" real word!

Given two words, how far apart are they?

Best alignment.

THEAR– –FTER
THE–REAFTER

Cost: 3.
Edit distance. "THEARFTER" to "THEREAFTER" uses one deletion, two insertions.

start with..	THEARFTER
delete position 4.	THERFTER

Edit Distance.

Spell Correction.

"THEARFTER"

THEARFTER versus THEREAFTER

Find "closest" real word!

Given two words, how far apart are they?

Best alignment.

THEAR– –FTER
THE–REAFTER

Cost: 3.
Edit distance. "THEARFTER" to "THEREAFTER" uses one deletion, two insertions.

start with..	THEARFTER
delete position 4.	THERFTER
insert E at position 5	THEREFTER

Edit Distance.

Spell Correction.

"THEARFTER"

THEARFTER versus THEREAFTER

Find "closest" real word!

Given two words, how far apart are they?

Best alignment.

THEAR– –FTER
THE–REAFTER

Cost: 3.
Edit distance. "THEARFTER" to "THEREAFTER" uses one deletion, two insertions.

start with..	THEARFTER
delete position 4.	THERFTER
insert E at position 5	THEREFTER
insert A at position 6	THEREAFTER

Edit Distance.

Spell Correction.

"THEARFTER"

THEARFTER versus THEREAFTER

Find "closest" real word!

Given two words, how far apart are they?

Best alignment.

THEAR– –FTER
THE–REAFTER

Cost: 3.
Edit distance. "THEARFTER" to "THEREAFTER" uses one deletion, two insertions.

start with..	THEARFTER
delete position 4.	THERFTER
insert E at position 5	THEREFTER
insert A at position 6	THEREAFTER

3 steps.

Edit Distance.

Spell Correction.

"THEARFTER"

THEARFTER versus THEREAFTER

Find "closest" real word!

Given two words, how far apart are they?

Best alignment.

THEAR– –FTER
THE–REAFTER

Cost: 3.
Edit distance. "THEARFTER" to "THEREAFTER" uses one deletion, two insertions.

start with..	THEARFTER
delete position 4.	THERFTER
insert E at position 5	THEREFTER
insert A at position 6	THEREAFTER

3 steps. Read operations off alignment.

Edit Distance.

Another alignment.

Edit Distance.

Another alignment.

THEAR–FTER
THEREAFTER

Edit Distance.

Another alignment.

THEAR–FTER
THEREAFTER

Cost 3:

Edit Distance.

Another alignment.

THEAR–FTER
THEREAFTER

Cost 3: Edit sequence: 2 substitutions, one insertion.

Edit Distance.

Another alignment.

THEAR–FTER
THEREAFTER

Cost 3: Edit sequence: 2 substitutions, one insertion.

start with..	THEARFTER
substitute R for A in position 4.	THERRFTER

Edit Distance.

Another alignment.

THEAR–FTER
THEREAFTER

Cost 3: Edit sequence: 2 substitutions, one insertion.

start with..	THEARFTER
substitute R for A in position 4.	THERRFTER
substitute E for R in position 5.	THEREFTER

Edit Distance.

Another alignment.

THEAR–FTER
THEREAFTER

Cost 3: Edit sequence: 2 substitutions, one insertion.

start with..	THEARFTER
substitute R for A in position 4.	THERRFTER
substitute E for R in position 5.	THEREFTER
insert A at position 6.	THEREAFTER

Edit Distance.

Another alignment.

THEAR–FTER
THEREAFTER

Cost 3: Edit sequence: 2 substitutions, one insertion.

start with..	THEARFTER
substitute R for A in position 4.	THERRFTER
substitute E for R in position 5.	THEREFTER
insert A at position 6.	THEREAFTER

Let's see the spell checker in action!

Edit Distance.

Another alignment.

THEAR–FTER
THEREAFTER

Cost 3: Edit sequence: 2 substitutions, one insertion.

start with..	THEARFTER
substitute R for A in position 4.	THERRFTER
substitute E for R in position 5.	THEREFTER
insert A at position 6.	THEREAFTER

Let's see the spell checker in action!

THEARFTER
THEA– –TER

Edit Distance.

Another alignment.

THEAR–FTER
THEREAFTER

Cost 3: Edit sequence: 2 substitutions, one insertion.

start with..	THEARFTER
substitute R for A in position 4.	THERRFTER
substitute E for R in position 5.	THEREFTER
insert A at position 6.	THEREAFTER

Let's see the spell checker in action!

THEARFTER
THEA– –TER

Cost 2:

Edit Distance.

Another alignment.

THEAR–FTER
THEREAFTER

Cost 3: Edit sequence: 2 substitutions, one insertion.

start with..	THEARFTER
substitute R for A in position 4.	THERRFTER
substitute E for R in position 5.	THEREFTER
insert A at position 6.	THEREAFTER

Let's see the spell checker in action!

THEARFTER
THEA– –TER

Cost 2: Edit sequence: 2 deletions.

Dynamic Programming Recipe

- Define a set of problems, such that
  - base case - easy to solve
  - final case - matches (closely) the final problem we want to solve.
- Write it as a recursion: Solve bigger problem in terms of the smaller problems. (Should be a DAG on the problem instances!)
- Compute the problems on the DAG in the linearized order!

Edit Distance.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Edit Distance.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Find edit distance between x and y.

Edit Distance.

Given: $x[1, \ldots, m]$ and $y[1, \ldots n]$.

Find edit distance between x and y.

Subproblems?

Edit Distance.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Find edit distance between x and y.

Subproblems?

How about edit distance between $x[1,\ldots,i]$ and $y[1,\ldots,i]$?

Edit Distance.

Given: $x[1, \ldots, m]$ and $y[1, \ldots n]$.

Find edit distance between x and y.

Subproblems?

How about edit distance between $x[1, \ldots, i]$ and $y[1, \ldots, i]$?

Alignment is fixed.

Edit Distance.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Find edit distance between x and y.

Subproblems?

How about edit distance between $x[1,\ldots,i]$ and $y[1,\ldots,i]$?

Alignment is fixed. There is nothing to do.

Edit Distance.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Find edit distance between x and y.

Subproblems?

How about edit distance between $x[1,\ldots,i]$ and $y[1,\ldots,i]$?

Alignment is fixed. There is nothing to do.

THERAFTER
THEREAFTER
Cost: 6.

Edit Distance.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Find edit distance between x and y.

Subproblems?

How about edit distance between $x[1,\ldots,i]$ and $y[1,\ldots,i]$?

Alignment is fixed. There is nothing to do.

THERAFTER
THEREAFTER
Cost: 6.

Have to search over different alignments!

Edit Distance.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Find edit distance between x and y.

Subproblems?

How about edit distance between $x[1,\ldots,i]$ and $y[1,\ldots,i]$?

Alignment is fixed. There is nothing to do.

THERAFTER
THEREAFTER
Cost: 6.

Have to search over different alignments!

THEAR– –FTER
THE–REAFTER

Edit Distance.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Find edit distance between x and y.

Subproblems?

How about edit distance between $x[1,\ldots,i]$ and $y[1,\ldots,i]$?

Alignment is fixed. There is nothing to do.

THERAFTER
THEREAFTER
Cost: 6.

Have to search over different alignments!

THEAR– –FTER
THE–REAFTER

Subsolution: aligned 5 characters of x to 4 characters of y.

Edit Distance.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Find edit distance between x and y.

Subproblems?

How about edit distance between $x[1,\ldots,i]$ and $y[1,\ldots,i]$?

Alignment is fixed. There is nothing to do.

THERAFTER
THEREAFTER
Cost: 6.

Have to search over different alignments!

THEAR– –FTER
THE–REAFTER

Subsolution: aligned 5 characters of x to 4 characters of y.

Should choose best such mapping!

Edit Distance.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Find edit distance between x and y.

Subproblems?

How about edit distance between $x[1,\ldots,i]$ and $y[1,\ldots,i]$?

Alignment is fixed. There is nothing to do.

THERAFTER
THEREAFTER
Cost: 6.

Have to search over different alignments!

THEAR– –FTER
THE–REAFTER

Subsolution: aligned 5 characters of x to 4 characters of y.

Should choose best such mapping!

Subproblem: "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Edit Distance Dynamic Program.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Edit Distance Dynamic Program.

Given: $x[1, \ldots, m]$ and $y[1, \ldots n]$.

Subproblem:
E(i,j) = "Edit Distance: $x[1, \ldots i]$ with $y[1, \ldots, j]$."

Edit Distance Dynamic Program.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Subproblem:
E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Compute $E(i,j)$?

Edit Distance Dynamic Program.

Given: $x[1, \ldots, m]$ and $y[1, \ldots n]$.

Subproblem:
E(i,j) = "Edit Distance: $x[1, \ldots i]$ with $y[1, \ldots, j]$."

Compute $E(i,j)$?

insertion	deletion	substitution
–	x[i]	x[i]
y[j]	–	y[j]

Edit Distance Dynamic Program.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Subproblem:
E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Compute $E(i,j)$?

insertion	deletion	substitution
–	x[i]	x[i]
y[j]	–	y[j]

Example:

Edit Distance Dynamic Program.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Subproblem:
E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Compute $E(i,j)$?

insertion	deletion	substitution
–	x[i]	x[i]
y[j]	–	y[j]

Example:

x = THEARF –FTER
y = THE–REAFTER

Edit Distance Dynamic Program.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Subproblem:
E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Compute $E(i,j)$?

insertion	deletion	substitution
–	x[i]	x[i]
y[j]	–	y[j]

Example:

x = THEARF –FTER
y = THE–REAFTER

Look at any aligned positions.

Edit Distance Dynamic Program.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Subproblem:
E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Compute $E(i,j)$?

insertion	deletion	substitution
–	x[i]	x[i]
y[j]	–	y[j]

Example:

x = THEARF –FTER
y = THE–REAFTER

Look at any aligned positions.

$E[6,4] = \min(1+E(6,3), 1+E(5,4), 1+E(5,3))$.
since $x[6] \neq x[4]$

Edit Distance Dynamic Program.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Subproblem:
E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Compute $E(i,j)$?

insertion	deletion	substitution
–	x[i]	x[i]
y[j]	–	y[j]

Example:

x = THEARF –FTER
y = THE–REAFTER

Look at any aligned positions.

$E[6,4] = \min(1+E(6,3), 1+E(5,4), 1+E(5,3))$.
since $x[6] \neq x[4]$

$E[3,3] = \min(1+E(3,2), 1+E(2,3), E(2,2))$.

Edit Distance Dynamic Program.

Given: $x[1,\ldots,m]$ and $y[1,\ldots n]$.

Subproblem:
E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Compute $E(i,j)$?

insertion	deletion	substitution
–	x[i]	x[i]
y[j]	–	y[j]

Example:

x = THEARF –FTER
y = THE–REAFTER

Look at any aligned positions.

$E[6,4] = \min(1 + E(6,3), 1 + E(5,4), 1+E(5,3))$.
since $x[6] \neq x[4]$

$E[3,3] = \min(1 + E(3,2), 1 + E(2,3), E(2,2))$. since $x[3] = y[3]$.

Edit Distance: Dynamic Program.

Subproblems:

Edit Distance: Dynamic Program.

Subprograms:

E(i,j) = "Edit Distance: $x[1, \ldots i]$ with $y[1, \ldots, j]$."

Edit Distance: Dynamic Program.

Subproblems:

E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Reccurrence:

Edit Distance: Dynamic Program.

Subproblems:

E(i,j) = “Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$.”

Reccurrence:

if $x[i] \neq y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), 1+E(i-1,j-1))$

Edit Distance: Dynamic Program.

Subprograms:

E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Reccurrence:

if $x[i] \neq y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), 1+E(i-1,j-1))$
if $x[i] = y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), E(i-1,j-1))$

Edit Distance: Dynamic Program.

Subproblems:

E(i,j) = "Edit Distance: $x[1,\dots i]$ with $y[1,\dots,j]$."

Reccurrence:

if $x[i] \neq y[j]$, $E(i,j) = \min(1 + E(i-1,j), 1 + E(i,j-1), 1 + E(i-1,j-1))$
if $x[i] = y[j]$, $E(i,j) = \min(1 + E(i-1,j), 1 + E(i,j-1), E(i-1,j-1))$

$E(0,j) =$

Edit Distance: Dynamic Program.

Subproblems:

E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Reccurrence:

if $x[i] \neq y[j]$, $E(i,j) = \min(1 + E(i-1,j), 1 + E(i,j-1), 1 + E(i-1,j-1))$
if $x[i] = y[j]$, $E(i,j) = \min(1 + E(i-1,j), 1 + E(i,j-1), E(i-1,j-1))$

$E(0,j) =$

(A) 0

(B) 1

Edit Distance: Dynamic Program.

Subproblems:

E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Reccurrence:

if $x[i] \neq y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), 1+E(i-1,j-1))$
if $x[i] = y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), E(i-1,j-1))$

$E(0,j) =$

(A) 0

(B) 1

(C) j

Edit Distance: Dynamic Program.

Subprobles:

E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Reccurrence:

if $x[i] \neq y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), 1+E(i-1,j-1))$
if $x[i] = y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), E(i-1,j-1))$

$E(0,j) =$

(A) 0
(B) 1
(C) j

(C).

Edit Distance: Dynamic Program.

Subproblems:

E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Reccurrence:

if $x[i] \neq y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), 1+E(i-1,j-1))$
if $x[i] = y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), E(i-1,j-1))$

$E(0,j) =$

(A) 0

(B) 1

(C) j

(C). Insert j characters.

Edit Distance: Dynamic Program.

Subproblems:

E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Reccurrence:

if $x[i] \neq y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), 1+E(i-1,j-1))$
if $x[i] = y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), E(i-1,j-1))$

$E(0,j) =$

(A) 0

(B) 1

(C) j

(C). Insert j characters.

$E(i,0) =$

(A) i

(B) 1

(C) 0

Edit Distance: Dynamic Program.

Subproblems:

E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Reccurrence:

if $x[i] \neq y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), 1+E(i-1,j-1))$
if $x[i] = y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), E(i-1,j-1))$

$E(0,j) =$

(A) 0

(B) 1

(C) j

(C). Insert j characters.

$E(i,0) =$

(A) i

(B) 1

(C) 0

(A).

Edit Distance: Dynamic Program.

Subproblems:

E(i,j) = "Edit Distance: $x[1,\ldots i]$ with $y[1,\ldots,j]$."

Reccurrence:

if $x[i] \neq y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), 1+E(i-1,j-1))$
if $x[i] = y[j]$, $E(i,j) = \min(1+E(i-1,j), 1+E(i,j-1), E(i-1,j-1))$

$E(0,j) =$

(A) 0

(B) 1

(C) j

(C). Insert j characters.

$E(i,0) =$

(A) i

(B) 1

(C) 0

(A). Delete i characters.

Dynamic Programming Program.

Make a table to store subproblem solutions: E(i,j)

Dynamic Programming Program.

Make a table to store subproblem solutions: E(i,j)

for i=0,...,m: E(i,0) =i Add i characters.
for j=0,...,n: E(0,j) =j Delete j characters.

for i=0,...,m:
  for j=0,...,n:

$$E(i,j) = \min\{\, E(i-1,j)+1,\; E(i-1,j-1)+1,\; E(i-1,j-1)+\text{diff}(i,j) \,\}$$

Dynamic Programming Program.

Make a table to store subproblem solutions: E(i,j)

for i=0,...,m: E(i,0) =i Add i characters.
for j=0,...,n: E(0,j) =j Delete j characters.

for i=0,...,m:
  for j=0,...,n:

$$E(i,j) = \min \{ E(i-1,j)+1, \quad E(i-1,j-1)+1, \quad E(i-1,j-1)+\mathrm{diff}(i,j) \}$$

Time:

Dynamic Programming Program.

Make a table to store subproblem solutions: E(i,j)

for i=0,. . . ,m: E(i,0) =i Add i characters.
for j=0,. . . ,n: E(0,j) =j Delete j characters.

for i=0,. . . ,m:
  for j=0,. . . ,n:

$$E(i,j) = \min \{ E(i-1,j)+1, \\ E(i-1,j-1)+1, \\ E(i-1,j-1)+\text{diff}(i,j) \}$$

Time:

(A) $\Theta(n+m)$

(B) $\Theta(nm)$

(C) $\Theta(n^m)$

Dynamic Programming Program.

Make a table to store subproblem solutions: E(i,j)

for i=0,...,m: E(i,0) =i Add i characters.
for j=0,...,n: E(0,j) =j Delete j characters.

for i=0,...,m:
  for j=0,...,n:

$$\text{E(i,j)} = \min \{ E(i-1,j)+1, \\ E(i-1,j-1)+1, \\ E(i-1,j-1)+\text{diff}(i,j) \}$$

Time:

(A) $\Theta(n+m)$

(B) $\Theta(nm)$

(C) $\Theta(n^m)$

(B)

Dynamic Programming Program.

Make a table to store subproblem solutions: E(i,j)

for i=0,...,m: E(i,0) =i Add i characters.
for j=0,...,n: E(0,j) =j Delete j characters.

for i=0,...,m:
  for j=0,...,n:
    $\text{E(i,j)} = \min\{\, E(i-1,j)+1,$
    $E(i-1,j-1)+1,$
    $E(i-1,j-1)+\text{diff}(i,j)\,\}$

Time:

(A) $\Theta(n+m)$

(B) $\Theta(nm)$

(C) $\Theta(n^m)$

(B) Nested loops: m outer iterations times n inner.

Example

	T	H	E	A	R	F

Example

		T	H	E	A	R	F
	0	1	2	3	4	5	6

Example

(T–,TH) cost 1
(T – –, THE) cost 2.

		T	H	E	A	R	F
	0	1	2	3	4	5	6
T	1	0	1	2	3	4	5

Example

(T–,TH) cost 1
(T – –, THE) cost 2.
(TH,TH) cost 0
(T H –, THE) cost 1.

		T	H	E	A	R	F
	0	1	2	3	4	5	6
T	1	0	1	2	3	4	5
H	2	1	0	1	2	3	4

Example

(T–,TH) cost 1
(T – –, THE) cost 2.

(TH,TH) cost 0
(T H –, THE) cost 1.

(THE,THE) cost 0
(THE–, THEA) cost 1.

		T	H	E	A	R	F
	0	1	2	3	4	5	6
T	1	0	1	2	3	4	5
H	2	1	0	1	2	3	4
E	3	2	1	0	1	2	3

Example

(T–,TH) cost 1
(T – –, THE) cost 2.

(TH,TH) cost 0
(T H –, THE) cost 1.

(THE,THE) cost 0
(THE–, THEA) cost 1.

(THE–,THER) cost 1
(THER, THEA) cost 1.

		T	H	E	A	R	F
	0	1	2	3	4	5	6
T	1	0	1	2	3	4	5
H	2	1	0	1	2	3	4
E	3	2	1	0	1	2	3
R	4	3	2	1	1	2	3

Example

(T–,TH) cost 1
(T – –, THE) cost 2.

(TH,TH) cost 0
(T H –, THE) cost 1.

(THE,THE) cost 0
(THE–, THEA) cost 1.

(THE–,THER) cost 1
(THER, THEA) cost 1.

		T	H	E	A	R	F
	0	1	2	3	4	5	6
T	1	0	1	2	3	4	5
H	2	1	0	1	2	3	4
E	3	2	1	0	1	2	3
R	4	3	2	1	1	2	3
E	5	4	3	2	2	2	3

Example

(T–,TH) cost 1
(T – –, THE) cost 2.

(TH,TH) cost 0
(T H –, THE) cost 1.

(THE,THE) cost 0
(THE–, THEA) cost 1.

(THE–,THER) cost 1
(THER, THEA) cost 1.

		T	H	E	A	R	F
	0	1	2	3	4	5	6
T	1	0	1	2	3	4	5
H	2	1	0	1	2	3	4
E	3	2	1	0	1	2	3
R	4	3	2	1	1	2	3
E	5	4	3	2	2	2	3
A	6	5	4	3	3	3	3

Dag View

What is the “dag of subproblems”?

Dag View

What is the “dag of subproblems”?

Node for each subproblem $E(i,j)$.

Dag View

What is the “dag of subproblems”?

Node for each subproblem $E(i,j)$.

Edges

Dag View

What is the “dag of subproblems”?

Node for each subproblem $E(i,j)$.

Edges
across $(E(i-1,j),E(i,j))$

Dag View

What is the "dag of subproblems"?

Node for each subproblem $E(i,j)$.

Edges
across $(E(i-1,j), E(i,j))$
down $(E(i,j-1), E(i,j))$

Dag View

What is the "dag of subproblems"?

Node for each subproblem $E(i, j)$.

Edges
across $(E(i-1, j), E(i, j))$
down $(E(i, j-1), E(i, j))$
diagonal $(E(i-1, j-1), E(i, j))$

Dag View

What is the "dag of subproblems"?

Node for each subproblem $E(i,j)$.

Edges
across $(E(i-1,j),E(i,j))$
down $(E(i,j-1),E(i,j))$
diagonal $(E(i-1,j-1),E(i,j))$

$O(nm)$ nodes.
$O(1)$ edges/node

Dag View

What is the "dag of subproblems"?

Node for each subproblem $E(i,j)$.

Edges
across $(E(i-1,j),E(i,j))$
down $(E(i,j-1),E(i,j))$
diagonal $(E(i-1,j-1),E(i,j))$

$O(nm)$ nodes.
$O(1)$ edges/node .. $O(nm)$ edges.

The Knapsack Problem

Want Knapsack of weight at most 29,

The Knapsack Problem

Want Knapsack of weight at most 29,

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

Max ratio heuristic?

The Knapsack Problem

Want Knapsack of weight at most 29,

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

Max ratio heuristic? Item 2

The Knapsack Problem

Want Knapsack of weight at most 29,

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

Max ratio heuristic? Item 2 , 1

The Knapsack Problem

Want Knapsack of weight at most 29,

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

Max ratio heuristic? Item 2 , 1 , 3.
Weight: 15+6+7 = 28.
Value: 43+18+21 = 82.

The Knapsack Problem

Want Knapsack of weight at most 29,

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

Max ratio heuristic? Item 2 , 1 , 3.
Weight: 15+6+7 = 28.
Value: 43+18+21 = 82.

Better Solution??

The Knapsack Problem

Want Knapsack of weight at most 29,

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

Max ratio heuristic? Item 2 , 1 , 3.
Weight: 15+6+7 = 28.
Value: 43+18+21 = 82.

Better Solution??

Switch item 4 for item 3.
Value: 43+18+23 = 84.

Can you make greedy lose by (almost) a factor of two?

Can you make greedy lose by (almost) a factor of two?

Size: 2,000,000

item	weight	value
1	2,000,000	1,999,999
2	1,000,001	1,000,001
3	1,000,001	1,000,001

Can you make greedy lose by (almost) a factor of two?

Size: 2,000,000

item	weight	value
1	2,000,000	1,999,999
2	1,000,001	1,000,001
3	1,000,001	1,000,001

Off by almost a factor of two!

Can you make greedy lose by (almost) a factor of two?

Size: 2,000,000

item	weight	value
1	2,000,000	1,999,999
2	1,000,001	1,000,001
3	1,000,001	1,000,001

Off by almost a factor of two!

Later:

Can you make greedy lose by (almost) a factor of two?

Size: 2,000,000

item	weight	value
1	2,000,000	1,999,999
2	1,000,001	1,000,001
3	1,000,001	1,000,001

Off by almost a factor of two!

Later: NP-complete...

Can you make greedy lose by (almost) a factor of two?

Size: 2,000,000

item	weight	value
1	2,000,000	1,999,999
2	1,000,001	1,000,001
3	1,000,001	1,000,001

Off by almost a factor of two!

Later: NP-complete...

But we will give a weakly polynomial time dynamic program!

...with Repetition.

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

Item 1 and

...with Repetition.

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

Item 1 and Item 1 again!

...with Repetition.

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

Item 1 and Item 1 again!

86 versus 85!

...with Repetition.

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

Item 1 and Item 1 again!

86 versus 85!

3 Items 3 and one item 4.

...with Repetition.

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

Item 1 and Item 1 again!

86 versus 85!

3 Items 3 and one item 4.

Also 86.

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack of weight w"

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack of weight w"

Solve subproblem $K(w)$

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack of weight w"

Solve subproblem $K(w)$ ...using smaller subproblems?

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack of weight w"

Solve subproblem $K(w)$ ...using smaller subproblems?

Consider solution. $\{i, j, k, \ldots\}$.

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack of weight w"

Solve subproblem $K(w)$ ...using smaller subproblems?

Consider solution. $\{i, j, k, \ldots\}$.

Take out one item,

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack of weight w"

Solve subproblem $K(w)$ ...using smaller subproblems?

Consider solution. $\{i, j, k, \ldots\}$.

Take out one item, say i,

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack of weight w"

Solve subproblem $K(w)$ ...using smaller subproblems?

Consider solution. $\{i, j, k, \ldots\}$.

Take out one item, say i, weight is $w - w_i$.

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack of weight w"

Solve subproblem $K(w)$ ...using smaller subproblems?

Consider solution. $\{i, j, k, \ldots\}$.

Take out one item, say i, weight is $w - w_i$.

Rest should be best knapsack of weight $w - w_i$.

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack of weight w"

Solve subproblem $K(w)$ ...using smaller subproblems?

Consider solution. $\{i, j, k, \ldots\}$.

Take out one item, say i, weight is $w - w_i$.

Rest should be best knapsack of weight $w - w_i$.

$K(w - w_i)$

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack of weight w"

Solve subproblem $K(w)$ ...using smaller subproblems?

Consider solution. $\{i, j, k, \ldots\}$.

Take out one item, say i, weight is $w - w_i$.

Rest should be best knapsack of weight $w - w_i$.
    $K(w - w_i)$ and add value of v_i.

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack of weight w"

Solve subproblem $K(w)$ ...using smaller subproblems?

Consider solution. $\{i, j, k, \ldots\}$.

Take out one item, say i, weight is $w - w_i$.

Rest should be best knapsack of weight $w - w_i$.
  $K(w - w_i)$ and add value of v_i.

$K(w) = \max_i (K(w - w_i) + v_i)$

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack of weight w"

Solve subproblem $K(w)$ ...using smaller subproblems?

Consider solution. $\{i, j, k, \ldots\}$.

Take out one item, say i, weight is $w - w_i$.

Rest should be best knapsack of weight $w - w_i$.
$K(w - w_i)$ and add value of v_i.

$K(w) = \max_i (K(w - w_i) + v_i)$ for $w - w_i \geq 0$.

Knapsack with Repetition

Given: W, $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value multiset of items of weight $\leq W$.

Dynamic Program.

Subproblems?
$K(w) =$ "Best value of knapsack of weight w"

Solve subproblem $K(w)$ ...using smaller subproblems?

Consider solution. $\{i, j, k, \ldots\}$.

Take out one item, say i, weight is $w - w_i$.

Rest should be best knapsack of weight $w - w_i$.
$K(w - w_i)$ and add value of v_i.

$K(w) = \max_i(K(w - w_i) + v_i)$ for $w - w_i \geq 0$.

$K(0) = 0$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$,

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0.$
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i)$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0.$
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18.$
$K(7) = 21,$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$,

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12)$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0.$
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18.$
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36.$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13)$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w-w_i)+v_i)$, $K(w-w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6-w_i)+v_i) = K(0)+v_2 = 0+18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12-6)+18 = K(6)+18 = 36$.
$K(13) = K(6)+21$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0.$
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18.$
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36.$
$K(13) = K(6) + 21 = K(7) + 18$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13) = K(6) + 21 = K(7) + 18 = 39$,

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13) = K(6) + 21 = K(7) + 18 = 39$,
$K(14)$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0.$
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18.$
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36.$
$K(13) = K(6) + 21 = K(7) + 18 = 39,$
$K(14) = K(7) + 21$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13) = K(6) + 21 = K(7) + 18 = 39$,
$K(14) = K(7) + 21 = 42$,

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0.$
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18.$
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36.$
$K(13) = K(6) + 21 = K(7) + 18 = 39,$
$K(14) = K(7) + 21 = 42,$
$K(15)$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13) = K(6) + 21 = K(7) + 18 = 39$,
$K(14) = K(7) + 21 = 42$,
$K(15) = K(8) + 21$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13) = K(6) + 21 = K(7) + 18 = 39$,
$K(14) = K(7) + 21 = 42$,
$K(15) = K(8) + 21 = 44$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13) = K(6) + 21 = K(7) + 18 = 39$,
$K(14) = K(7) + 21 = 42$,
$K(15) = K(8) + 21 = 44$
$K(16)$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18.$
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36.$
$K(13) = K(6) + 21 = K(7) + 18 = 39,$
$K(14) = K(7) + 21 = 42,$
$K(15) = K(8) + 21 = 44$
$K(16) = K(8) + 23$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13) = K(6) + 21 = K(7) + 18 = 39$,
$K(14) = K(7) + 21 = 42$,
$K(15) = K(8) + 21 = 44$
$K(16) = K(8) + 23 = 46$.

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13) = K(6) + 21 = K(7) + 18 = 39$,
$K(14) = K(7) + 21 = 42$,
$K(15) = K(8) + 21 = 44$
$K(16) = K(8) + 23 = 46$. $K(17)$ undefined,

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13) = K(6) + 21 = K(7) + 18 = 39$,
$K(14) = K(7) + 21 = 42$,
$K(15) = K(8) + 21 = 44$
$K(16) = K(8) + 23 = 46$. $K(17)$ undefined, $K(18)$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13) = K(6) + 21 = K(7) + 18 = 39$,
$K(14) = K(7) + 21 = 42$,
$K(15) = K(8) + 21 = 44$
$K(16) = K(8) + 23 = 46$. $K(17)$ undefined, $K(18) = K(12) + 18$

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i (K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i (K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13) = K(6) + 21 = K(7) + 18 = 39$,
$K(14) = K(7) + 21 = 42$,
$K(15) = K(8) + 21 = 44$
$K(16) = K(8) + 23 = 46$. $K(17)$ undefined, $K(18) = K(12) + 18 = 54$,

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13) = K(6) + 21 = K(7) + 18 = 39$,
$K(14) = K(7) + 21 = 42$,
$K(15) = K(8) + 21 = 44$
$K(16) = K(8) + 23 = 46$. $K(17)$ undefined, $K(18) = K(12) + 18 = 54$, ...

Example

Weight 29

item	weight	value
1	15	43
2	6	18
3	7	21
4	8	23

$K(0) = 0$.
Recurrence: $K(w) = \max_i(K(w - w_i) + v_i)$, $K(w - w_i)$ is defined.

$K(1), \ldots, K(5)$ undefined

$K(6) = \max_i(K(6 - w_i) + v_i) = K(0) + v_2 = 0 + 18 = 18$.
$K(7) = 21$, $K(8) = 23$, $K(9), K(10), K(11)$ undefined.
$K(12) = K(12 - 6) + 18 = K(6) + 18 = 36$.
$K(13) = K(6) + 21 = K(7) + 18 = 39$,
$K(14) = K(7) + 21 = 42$,
$K(15) = K(8) + 21 = 44$
$K(16) = K(8) + 23 = 46$. $K(17)$ undefined, $K(18) = K(12) + 18 = 54$, ...

Read off highest valued $K(w)$ for value of solution.

Complexity: Knapsack with Repetition.

$K(w) = \max_i(K(w - w_i) + v_i)$

Complexity: Knapsack with Repetition.

$K(w) = \max_i(K(w - w_i) + v_i)$ for $w - w_i \geq 0$.

Complexity: Knapsack with Repetition.

$K(w) = \max_i(K(w - w_i) + v_i)$ for $w - w_i \geq 0$.

$K(0) = 0$

Complexity: Knapsack with Repetition.

$K(w) = \max_i(K(w - w_i) + v_i)$ for $w - w_i \geq 0$.

$K(0) = 0$

W entries,

Complexity: Knapsack with Repetition.

$K(w) = \max_i(K(w - w_i) + v_i)$ for $w - w_i \geq 0$.

$K(0) = 0$

W entries, $O(n)$ time per entry.

Complexity: Knapsack with Repetition.

$K(w) = \max_i(K(w - w_i) + v_i)$ for $w - w_i \geq 0$.

$K(0) = 0$

W entries, $O(n)$ time per entry.
(Scan over all n items in $\max_i$.)

Complexity: Knapsack with Repetition.

$K(w) = \max_i(K(w - w_i) + v_i)$ for $w - w_i \geq 0$.

$K(0) = 0$

W entries, $O(n)$ time per entry.
(Scan over all n items in $\max_i$.)

Total: $O(nW)$ time.

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value *set* of items of weight $\leq W$.

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value *set* of items of weight $\leq W$.

Previous Dynamic Program.

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value *set* of items of weight $\leq W$.

Previous Dynamic Program.

$K(w) =$

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value *set* of items of weight $\leq W$.

Previous Dynamic Program.

$K(w) =$ "Best value of knapsack

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value *set* of items of weight $\leq W$.

Previous Dynamic Program.

$K(w) =$ "Best value of knapsack of weight w"

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value *set* of items of weight $\leq W$.

Previous Dynamic Program.

$K(w) =$ "Best value of knapsack of weight w"

$K(w) = \max_i K(w - w_i) + v_i$

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value *set* of items of weight $\leq W$.

Previous Dynamic Program.

$K(w) =$ "Best value of knapsack of weight w"

$K(w) = \max_i K(w - w_i) + v_i$ for $w - w_i \geq 0$.

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value *set* of items of weight $\leq W$.

Previous Dynamic Program.

$K(w) =$ "Best value of knapsack of weight w"

$K(w) = \max_i K(w - w_i) + v_i$ for $w - w_i \geq 0$.

$K(0) = 0$

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value *set* of items of weight $\leq W$.

Previous Dynamic Program.

$K(w) =$ "Best value of knapsack of weight w"

$K(w) = \max_i K(w - w_i) + v_i$ for $w - w_i \geq 0$.

$K(0) = 0$

No way to control for using items over and over again!

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Subproblem?

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Subproblem?

Idea?

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Subproblem?

Idea? Subset!

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Subproblem?

Idea? Subset! In subset or not!

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Subproblem?

Idea? Subset! In subset or not! Sequential.

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Subproblem?

Idea? Subset! In subset or not! Sequential.

Best knapsack using first i items?

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Subproblem?

Idea? Subset! In subset or not! Sequential.

Best knapsack using first i items?

Best depends on how much space is left.

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Subproblem?

Idea? Subset! In subset or not! Sequential.

Best knapsack using first i items?

Best depends on how much space is left.

Best knapsack of weight w using first i items.

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Subproblem?

Idea? Subset! In subset or not! Sequential.

Best knapsack using first i items?

Best depends on how much space is left.

Best knapsack of weight w using first i items.

$K(w, i) =$

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Subproblem?

Idea? Subset! In subset or not! Sequential.

Best knapsack using first i items?

Best depends on how much space is left.

Best knapsack of weight w using first i items.

$K(w, i) =$ "Best weight w Knapsack with subset of first i items."

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Subproblem?

Idea? Subset! In subset or not! Sequential.

Best knapsack using first i items?

Best depends on how much space is left.

Best knapsack of weight w using first i items.

$K(w, i) =$ "Best weight w Knapsack with subset of first i items."

Either add item or not!

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Subproblem?

Idea? Subset! In subset or not! Sequential.

Best knapsack using first i items?

Best depends on how much space is left.

Best knapsack of weight w using first i items.

$K(w, i) =$ "Best weight w Knapsack with subset of first i items."

Either add item or not!

$K(w, i) = \max\{K(w - w_i, i - 1) + v_i, K(w, i - 1)\}$

Knapsack without Repetition

Given: Weight: W, Items: $(v_1, w_1), (v_2, w_2), \ldots, (v_n, w_n)$.

Find: highest value subset of items of weight $\leq W$.

Subproblem?

Idea? Subset! In subset or not! Sequential.

Best knapsack using first i items?

Best depends on how much space is left.

Best knapsack of weight w using first i items.

$K(w,i) =$ "Best weight w Knapsack with subset of first i items."

Either add item or not!

$K(w,i) = \max\{K(w-w_i, i-1) + v_i, K(w, i-1)\}$

$K(0,0) = 0$

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$,

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1)$

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.

Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i,i-1)+v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$,

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.

Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i,i-1)+v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$,

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2)$

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$,

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2)$

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i,i-1)+v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$,

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2)$

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i,i-1)+v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i,i-1)+v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$,

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$, $K(6,3)$

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$, $K(6,3) = 18$,

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$, $K(6,3) = 18$, $K(7,3)$

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$, $K(6,3) = 18$, $K(7,3) = 21$,

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$, $K(6,3) = 18$, $K(7,3) = 21$, $K(13,3)$

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$, $K(6,3) = 18$, $K(7,3) = 21$, $K(13,3) = 39$,

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$, $K(6,3) = 18$, $K(7,3) = 21$, $K(13,3) = 39$, $K(15,3)$

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$, $K(6,3) = 18$, $K(7,3) = 21$, $K(13,3) = 39$, $K(15,3) = 43$,

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$, $K(6,3) = 18$, $K(7,3) = 21$, $K(13,3) = 39$, $K(15,3) = 43$,
$K(22,3)$

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$, $K(6,3) = 18$, $K(7,3) = 21$, $K(13,3) = 39$, $K(15,3) = 43$,
$K(22,3) = 64$

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$, $K(6,3) = 18$, $K(7,3) = 21$, $K(13,3) = 39$, $K(15,3) = 43$,
$K(22,3) = 64$ ...

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$, $K(6,3) = 18$, $K(7,3) = 21$, $K(13,3) = 39$, $K(15,3) = 43$,
$K(22,3) = 64$ ...

....

Example no Repetition

	item	weight	value
	1	15	43
Weight 30	2	6	18
	3	7	21
	4	8	23

$K(0,0) = 0$.
Recurrence: $K(w,i) = \max(K(w,i-1), K(w-w_i, i-1) + v_i)$.

$K(0,1) = 0$, $K(15,1) = 43$, All other $K(w,1)$ undefined.

$K(0,2) = 0$, $K(6,2) = 18$, $K(15,2) = 43$, $K(21,2) = 61$,
All other $K(w,i)$ undefined.

$K(0,3) = 0$, $K(6,3) = 18$, $K(7,3) = 21$, $K(13,3) = 39$, $K(15,3) = 43$,
$K(22,3) = 64$ ...

....

Read off highest value of $K(w,n)$ for answer.