BLG336E - Analysis of Algorithms II

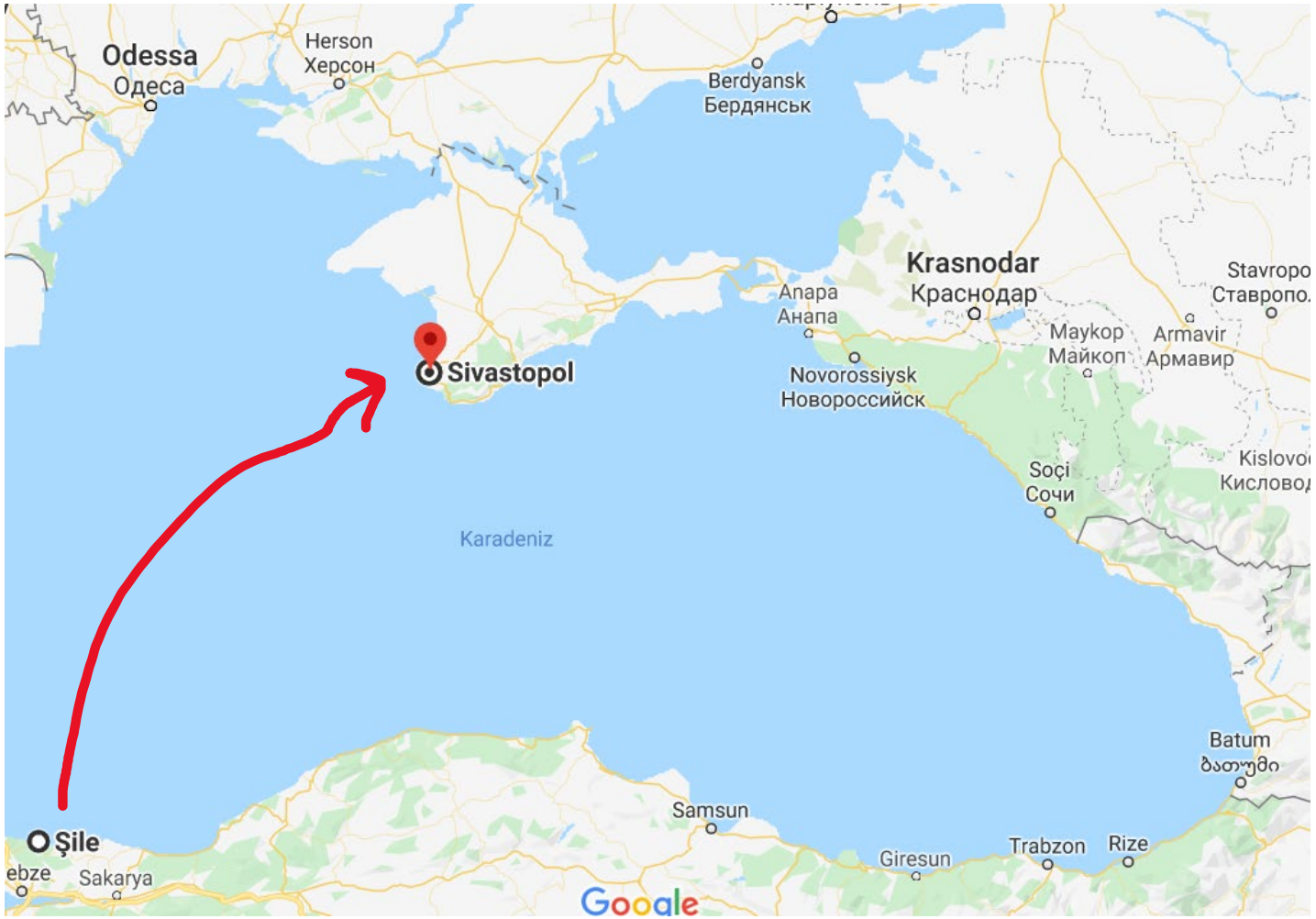
Recitation 9 – Midterm Corrections & Dynamic Programming

10.05.2021

Alperen Kantarcı

A **knapsack** practice: problem definition

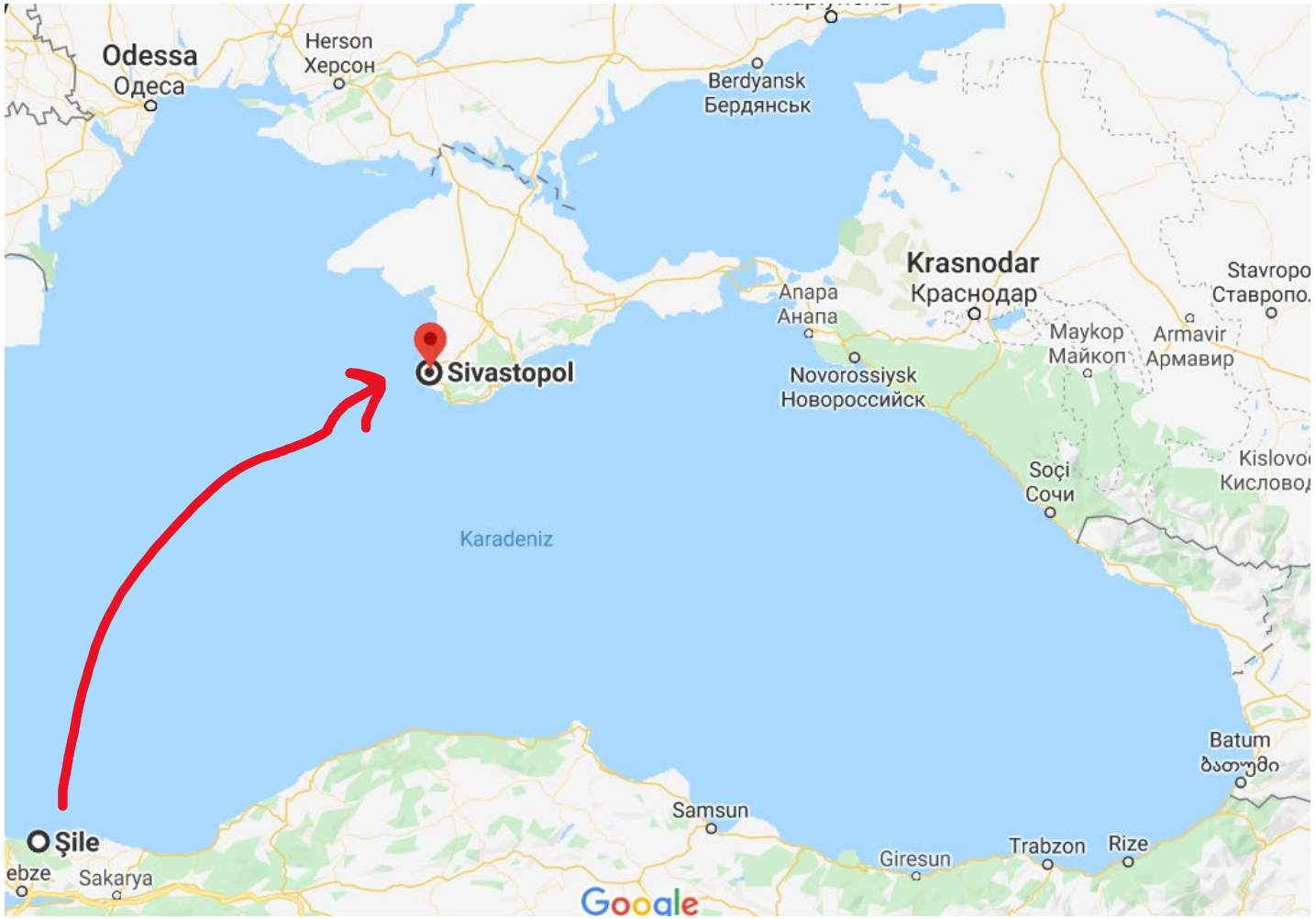
- Since you haven't exercised much during pandemic, after the COVID-19 disappears, you decided to cross the Black Sea from *Şile* to *Sevastopol/Ukraine* on a boat with a friend.
- On this trip, you need to get some canned food with you. You need to **maximize the amount of energy** that you get from the canned food.



A **knapsack** practice: problem definition

- You can carry **at most 10 kgs** of food. The kinds of canned food, their weight and energy are given below.
- **Which foods should you take with you?**

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	5	10



Let's remember the bottom-up dynamic programming (DP) approach
1. step: iteratively fill the DP table

```
Input: n, w_1, . . . , w_N, v_1, . . . , v_n
for w = 0 to W
  M[0, w] = 0
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
return M[n, W]
```

We have n items, weights (w) & values (v) of them

Let's remember the bottom-up dynamic programming (DP) approach
1. step: iteratively fill the DP table

```
Input: n, w_1, . . . , w_N, v_1, . . . , v_n
for w = 0 to W
  M[0, w] = 0
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
return M[n, W]
```

We have n items, weights (w) & values (v) of them

We are filling first row of the table with 0.

Let's remember the bottom-up dynamic programming (DP) approach
1. step: iteratively fill the DP table

```
Input: n, w_1, . . . , w_N, v_1, . . . , v_n
for w = 0 to W
  M[0, w] = 0
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
return M[n, W]
```

We have n items, weights (w) & values (v) of them

We are filling first row of the table with 0.

We are filling rest of the rows by computing the optimal values.

Bellman equation (optimal value calculation)

$\boldsymbol{OPT}(\boldsymbol{i}, \boldsymbol{w})$: optimal value of knapsack problem with items 1, 2, … , i subject to a weight limit w

$\boldsymbol{OPT}(\boldsymbol{n}, \boldsymbol{W})$: optimal value of the knapsack problem with all given items and the weight limit

$$OPT(i,w) = \begin{cases} 0 & \text{if } i = 0 \\ OPT(i-1,w) & \text{if } w_i > w \\ \max\{\, OPT(i-1,w),\ \ v_i + OPT(i-1, w-w_i)\,\} & \text{otherwise} \end{cases}$$

Bellman equation (optimal value calculation)

$\boldsymbol{OPT(i, w)}$: optimal value of knapsack problem with items 1, 2, … , i subject to a weight limit w

$\boldsymbol{OPT(n, W)}$: optimal value of the knapsack problem with all given items and the weight limit

Our goal is to calculate $\boldsymbol{OPT(n, W)}$

$$OPT(i,w) = \begin{cases} 0 & \text{if } i = 0 \\ OPT(i-1,w) & \text{if } w_i > w \\ \max\{\, OPT(i-1,w),\ \ v_i + OPT(i-1, w-w_i)\,\} & \text{otherwise} \end{cases}$$

Bellman equation (optimal value calculation)

$\boldsymbol{OPT(i, w)}$: optimal value of knapsack problem with items 1, 2, … , i subject to weight limit

$\boldsymbol{OPT(n, W)}$: optimal value of the knapsack problem with all given items and weight limit

Our goal is to calculate $\boldsymbol{OPT(n, W)}$

$$OPT(i,w) = \begin{cases} 0 & \text{if } i = 0 \\ OPT(i-1,w) & \text{if } w_i > w \\ \max\{\, OPT(i-1,w),\ \ v_i + OPT(i-1, w-w_i)\,\} & \text{otherwise} \end{cases}$$

Case 1: $\boldsymbol{OPT(i-1, w)}$

That means we **do not select** the item i

Bellman equation (optimal value calculation)

$\boldsymbol{OPT(i, w)}$: optimal value of knapsack problem with items 1, 2, … , i subject to weight limit

$\boldsymbol{OPT(n, W)}$: optimal value of the knapsack problem with all given items and weight limit

→ Our goal is to calculate $\boldsymbol{OPT(n, W)}$

$$OPT(i,w) = \begin{cases} 0 & \text{if } i = 0 \\ OPT(i-1,w) & \text{if } w_i > w \\ \max\{\, OPT(i-1,w),\ v_i + OPT(i-1, w-w_i)\,\} & \text{otherwise} \end{cases}$$

Case 1: $\boldsymbol{OPT(i-1, w)}$
That means we **do not select** the item i

Case 2: $\boldsymbol{v_i + OPT(i-1, w-w_i)}$
That means we **select** the item i:
Collect the value v_i
New weight limit $= w - w_i$

Let's remember the bottom-up dynamic programming (DP) approach
1. step: iteratively fill the DP table

```
Input: n, w_1, . . . , w_N, v_1, . . . , v_n
for w = 0 to W
  M[0, w] = 0
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
return M[n, W]
```

We have n items, weights (w) & values (v) of them

We are filling first row of the table with 0.

We are filling rest of the rows by computing the optimal values.

Finally we are returning the filled table

Let's remember the bottom-up dynamic programming (DP) approach
2. step: construct the optimal solution using the DP table

```
Input: M, n, w_1,...,w_N, v_1,...,v_n
i = n, k = W
while i, k > 0
   if (M[i, k] != M[i - 1, k])
      print(i)
      k = k - w_i
   i = i - 1
```

We have M (the DP table) n items, weights (w) & values (v) of them

Let's remember the bottom-up dynamic programming (DP) approach
2. step: construct the optimal solution using the DP table

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
   if (M[i, k] != M[i - 1, k])
      print(i)
      k = k - w_i
   i = i - 1
```

We have M (the DP table) n items, weights (w) & values (v) of them

We set the i to number of items (n), k to weight limit (W)

Let's remember the bottom-up dynamic programming (DP) approach
2. step: construct the optimal solution using the DP table

```
Input: M, n, w_1,...,w_N, v_1,...,v_n
i = n, k = W
while i, k > 0
   if (M[i, k] != M[i - 1, k])
      print(i)
      k = k - w_i
   i = i - 1
```

We have M (the DP table) n items, weights (w) & values (v) of them

We set the i to number of items (n), k to weight limit (W)

Loop through the table starting from most right-bottom cell while there are items & weights in the table that are not checked yet

Let's remember the bottom-up dynamic programming (DP) approach
2. step: construct the optimal solution using the DP table

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
   if (M[i, k] != M[i - 1, k])
      print(i)
      k = k - w_i
   i = i - 1
```

We have M (the DP table) n items, weights (w) & values (v) of them

We set the i to number of items (n), k to weight limit (W)

Loop through the table starting from most right-bottom cell while there are items & weights in the table that are not checked yet

For a weight limit k, If the optimal value gained by selecting items 1, 2, …, i **is not the same** as the optimal value gained by selecting items 1, 2, …, i – 1 include the item i in the knapsack and reduce the weight limit by the item's weight, then continue to loop without the recently selected item

Let's remember the bottom-up dynamic programming (DP) approach
2. step: construct the optimal solution using the DP table

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
   if (M[i, k] != M[i - 1, k])
      print(i)
      k = k - w_i
   i = i - 1
```

We have M (the DP table) n items, weights (w) & values (v) of them

We set the i to number of items (n), k to weight limit (W)

Loop through the table starting from most right-bottom cell while there are items & weights in the table that are not checked yet

For a weight limit k, If the optimal value gained by selecting items 1, 2, …, i **is not the same** as the optimal value gained by selecting items 1, 2, …, i – 1 include the item i in the knapsack and reduce the weight limit by the item's weight, then continue to loop without the recently selected item

Let's remember the bottom-up dynamic programming (DP) approach
2. step: construct the optimal solution using the DP table

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
    if (M[i, k] != M[i - 1, k])
        print(i)
        k = k - w_i
    i = i - 1
```

We have M (the DP table) n items, weights (w) & values (v) of them

We set the i to number of items (n), k to weight limit (W)

Loop through the table starting from most right-bottom cell while there are items & weights in the table that are not checked yet

For a weight limit k, If the optimal value gained by selecting items 1, 2, …, i **is not the same** as the optimal value gained by selecting items 1, 2, …, i – 1 include the item i in the knapsack and reduce the weight limit by the item's weight, then continue to loop without the recently selected item

Let's remember the bottom-up dynamic programming (DP) approach
2. step: construct the optimal solution using the DP table

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
    if (M[i, k] != M[i - 1, k])
        print(i)
        k = k - w_i
    i = i - 1
```

We have M (the DP table) n items, weights (w) & values (v) of them

We set the i to number of items (n), k to weight limit (W)

Loop through the table starting from most right-bottom cell while there are items & weights in the table that are not checked yet

For a weight limit k, If the optimal value gained by selecting items 1, 2, …, i **is not the same** as the optimal value gained by selecting items 1, 2, …, i – 1 include the item i in the knapsack and reduce the weight limit by the item's weight, then continue to loop without the recently selected item

Let's remember the bottom-up dynamic programming (DP) approach
2. step: construct the optimal solution using the DP table

```
Input: M, n, w_1,...,w_N, v_1,...,v_n
i = n, k = W
while i, k > 0
    if (M[i, k] != M[i - 1, k])
        print(i)
        k = k - w_i
    i = i - 1
```

We have M (the DP table) n items, weights (w) & values (v) of them

We set the i to number of items (n), k to weight limit (W)

Loop through the table starting from most right-bottom cell while there are items & weights in the table that are not checked yet

For a weight limit k, If the optimal value gained by selecting items 1, 2, …, i **is not the same** as the optimal value gained by selecting items 1, 2, …, i – 1 include the item i in the knapsack and reduce the weight limit by the item's weight, then continue to loop without the recently selected item

A knapsack practice: fill DP the table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	5	10

Input: n, $w_1, \ldots, w_N$, $v_1, \ldots, v_n$

```
for w = 0 to W
```
← Loop through weights: zero weight to max. allowed limit

```
  M[0, w] = 0
```
← Assign zero (0) to the first row which represents of selecting **no** item

`w = 0 . . .` `. . . w = 10`

	Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
M[0,]	{ }	0	0	0	0	0	0	0	0	0	0	0
	{A}											
	{A, B}											
	{A, B, C}											
	{A, B, C, D}											
	{A, B, C, D, E}											

A knapsack practice: fill DP the table

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

Loop through all foods (A, B, C, D, E)

Loop through weights: zero weight (0) to max. allowed limit (10)

Compare the weight of food (w_i) with the weight limit (w)

If the food's weight exceeds the limit, do not select that food

Else select the maximum of two cases:
1) **do not** including the current item to knapsac
2) including the current item to knapsack

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}											
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_1 > w ?
2 > 0 ?
YES

w = 0

i = 1

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	?										
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[1, 0] = M[0, 0]

w = 0

i = 1

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0										
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

$w_1 > w$?
2 > 1 ?
YES

w = 1

i = 1

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	?									
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

`M[1, 1] = M[0, 1]`

`w = 1`

`i = 1`

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0									
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_1 > w ?
2 > 2 ?
NO

w = 2

i = 1

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	?								
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (wi > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - wi]}
```

```
M[1, 2] = max {M[0, 2], 6 + M[0, 2 - 2]}
M[1, 2] = max {M[0, 2], 6 + M[0, 0]}
M[1, 2] = max {0, 6 + 0]} = 6
```

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

w = 2

	Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
	{ }	0	0	0	0	0	0	0	0	0	0	0
i = 1	{A}	0	0	6								
	{A, B}											
	{A, B, C}											
	{A, B, C, D}											
	{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_1 > w ?
2 > 3 ?
NO

w = 3

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	?							
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 1

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[1, 3] = max {M[0, 3], 6 + M[0, 3 - 2]}
M[1, 3] = max {M[0, 3], 6 + M[0, 1]}
M[1, 3] = max {0, 6 + 0]} = 6

Food	Weight	Energy
A	w_i 2	v_i 6
B	3	5
C	5	10
D	4	4
E	5	10

w = 3

	Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
	{ }	0	0	0	0	0	0	0	0	0	0	0
i = 1	{A}	0	0	6	6							
	{A, B}											
	{A, B, C}											
	{A, B, C, D}											
	{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_1 > w ?
2 > 4 ?
NO

w = 4

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	?						
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 1

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[1, 4] = max {M[0, 4], 6 + M[0, 4 - 2]}
M[1, 4] = max {M[0, 4], 6 + M[0, 2]}
M[1, 4] = max {0, 6 + 0]} = 6

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

w = 4

	Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
	{ }	0	0	0	0	0	0	0	0	0	0	0
i = 1	{A}	0	0	6	6	6						
	{A, B}											
	{A, B, C}											
	{A, B, C, D}											
	{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

$w_1 > w$?
2 > 5 ?
NO

w = 5

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	?					
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 1

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (wi > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - wi]}
```

M[1, 5] = max {M[0, 5], 6 + M[0, 5 - 2]}
M[1, 5] = max {M[0, 5], 6 + M[0, 3]}
M[1, 5] = max {0, 6 + 0]} = 6

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

w = 5

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
i = 1 {A}	0	0	6	6	6	6					
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_1 > w ?
2 > 6 ?
NO

w = 6

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	?				
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 1

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (wi > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - wi]}
```

M[1, 6] = max {M[0, 6], 6 + M[0, 6 - 2]}
M[1, 6] = max {M[0, 6], 6 + M[0, 4]}
M[1, 6] = max {0, 6 + 0]} = 6

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

w = 6

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
i = 1 {A}	0	0	6	6	6	6	6				
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

$w_1 > w$?
2 > 7 ?
NO

w = 7

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
i = 1 {A}	0	0	6	6	6	6	6	?			
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (wi > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - wi]}
```

M[1, 7] = max {M[0, 7], 6 + M[0, 7 - 2]}
M[1, 7] = max {M[0, 7], 6 + M[0, 5]}
M[1, 7] = max {0, 6 + 0]} = 6

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

w = 7

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
i = 1 {A}	0	0	6	6	6	6	6	6	?		
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

$w_1 > w$?
2 > 8 ?
NO

w = 8

i = 1

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	?		
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (wi > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - wi]}
```

```
M[1, 8] = max {M[0, 8], 6 + M[0, 8 - 2]}
M[1, 8] = max {M[0, 8], 6 + M[0, 6]}
M[1, 8] = max {0, 6 + 0]} = 6
```

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

w = 8

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
i = 1 {A}	0	0	6	6	6	6	6	6	6		
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_1 > w ?
2 > 9 ?
NO

w = 9

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
i = 1 {A}	0	0	6	6	6	6	6	6	6	?	
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (wi > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - wi]}
```

M[1, 9] = max {M[0, 9], 6 + M[0, 9 - 2]}
M[1, 9] = max {M[0, 9], 6 + M[0, 7]}
M[1, 9] = max {0, 6 + 0]} = 6

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

w = 9

	Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
	{ }	0	0	0	0	0	0	0	0	0	0	0
i = 1	{A}	0	0	6	6	6	6	6	6	6	6	
	{A, B}											
	{A, B, C}											
	{A, B, C, D}											
	{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (wi > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - wi]}
```

w_1 > w ?
2 > 10 ?
NO

w = 10

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
i = 1 {A}	0	0	6	6	6	6	6	6	6	6	?
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[1, 10] = max {M[0, 10], 6 + M[0, 10 - 2]}
M[1, 10] = max {M[0, 10], 6 + M[0, 8]}
M[1, 10] = max {0, 6 + 0]} = 6
```

Food	Weight	Energy
A	w_1 2	v_1 6
B	3	5
C	5	10
D	4	4
E	5	10

w = 9

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
i = 1 {A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}											
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_1 > w ?
3 > 0 ?
YES

w = 0

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	?										
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 2

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

`M[2, 0] = M[1, 0]`

`w = 0`

`i = 2`

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0										
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_2 > w ?
3 > 1 ?
YES

w = 1

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	?									
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 2

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[2, 1] = M[1, 1]

w = 1

i = 2

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0									
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_2 > w ?
3 > 2 ?
YES

w = 2

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	?								
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 2

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

`M[2, 2] = M[1, 2]`

`w = 2`

`i = 2`

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6								
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_2 > w ?
3 > 3 ?
NO

w = 3

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
i = 2 {A, B}	0	0	6	?							
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[2, 3] = max {M[1, 3], 5 + M[1, 3 - 3]}
M[2, 3] = max {M[1, 3], 5 + M[1, 0]}
M[2, 3] = max {6, 5 + 0]} = 6

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

w = 3

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
i = 2 {A, B}	0	0	6	6							
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_2 > w ?
3 > 4 ?
NO

w = 4

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	?						
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 2

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[2, 4] = max {M[1, 4], 5 + M[1, 4 - 3]}
M[2, 4] = max {M[1, 4], 5 + M[1, 1]}
M[2, 4] = max {6, 5 + 0]} = 6
```

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

w = 4

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
i = 2 {A, B}	0	0	6	6	6						
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_2 > w ?
3 > 5 ?
NO

w = 5

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	?					
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 2

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[2, 5] = max {M[1, 5], 5 + M[1, 5 - 3]}
M[2, 5] = max {M[1, 5], 5 + M[1, 2]}
M[2, 5] = max {6, 5 + 6]} = 11
```

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

w = 5

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
i = 2 {A, B}	0	0	6	6	6	11					
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_2 > w ?
3 > 6 ?
NO

w = 6

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
i = 2 {A, B}	0	0	6	6	6	11	?				
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[2, 6] = max {M[1, 6], 5 + M[1, 6 - 3]}
M[2, 6] = max {M[1, 6], 5 + M[1, 3]}
M[2, 6] = max {6, 5 + 6]} = 11
```

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

w = 6

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
i = 2 {A, B}	0	0	6	6	6	11	11				
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_2 > w ?
3 > 7 ?
NO

w = 7

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	?			
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 2

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[2, 7] = max {M[1, 7], 5 + M[1, 7 - 3]}
M[2, 7] = max {M[1, 7], 5 + M[1, 4]}
M[2, 7] = max {6, 5 + 6]} = 11

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

w = 7

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11			
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 2

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_2 > w ?
3 > 8 ?
NO

w = 8

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	?		
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 2

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[2, 8] = max {M[1, 8], 5 + M[1, 8 - 3]}
M[2, 8] = max {M[1, 8], 5 + M[1, 5]}
M[2, 8] = max {6, 5 + 6]} = 11

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

w = 8

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11		
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 2

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_2 > w ?
3 > 9 ?
NO

w = 9

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	?	
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 2

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[2, 9] = max {M[1, 9], 5 + M[1, 9 - 3]}
M[2, 9] = max {M[1, 9], 5 + M[1, 6]}
M[2, 9] = max {6, 5 + 6]} = 11

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

w = 9

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
i = 2 {A, B}	0	0	6	6	6	11	11	11	11	11	
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_2 > w ?
3 > 10 ?
NO

w = 10

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
i = 2 {A, B}	0	0	6	6	6	11	11	11	11	11	?
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[2, 10] = max {M[1, 10], 5 + M[1, 10 - 3]}
M[2, 10] = max {M[1, 10], 5 + M[1, 7]}
M[2, 10] = max {6, 5 + 6]} = 11
```

Food	Weight	Energy
A	2	6
B	w_2 **3**	v_2 **5**
C	5	10
D	4	4
E	5	10

w = 10

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}											
{A, B, C, D}											
{A, B, C, D, E}											

i = 2

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_3 > w ?
5 > 0 ?
YES

w = 0

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	?										
{A, B, C, D}											
{A, B, C, D, E}											

i = 3

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[3, 0] = M[2, 0]

w = 0

i = 3

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0										
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_3 > w ?
5 > 1 ?
YES

w = 1

i = 3

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	?									
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[3, 1] = M[2, 1]

w = 1

i = 3

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0									
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_3 > w ?
5 > 2 ?
YES

w = 2

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	?								
{A, B, C, D}											
{A, B, C, D, E}											

i = 3

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[3, 2] = M[2, 2]

w = 2

i = 3

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6								
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_3 > w ?
5 > 3 ?
YES

w = 3

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	?							
{A, B, C, D}											
{A, B, C, D, E}											

i = 3

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[3, 3] = M[2, 3]

w = 3

i = 3

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6							
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_3 > w ?
5 > 4 ?
YES

w = 4

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
i = 3 {A, B, C}	0	0	6	6	?						
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[3, 4] = M[2, 4]

w = 4

i = 3

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6						
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_3 > w ?
5 > 5 ?
NO

w = 5

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	?					
{A, B, C, D}											
{A, B, C, D, E}											

i = 3

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[3, 5] = max {M[2, 5], 10 + M[2, 5 - 5]}
M[3, 5] = max {M[2, 5], 10 + M[2, 0]}
M[3, 5] = max {11, 10 + 0]} = 11

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

w = 5

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11					
{A, B, C, D}											
{A, B, C, D, E}											

i = 3

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_3 > w ?
5 > 6 ?
NO

w = 6

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	?				
{A, B, C, D}											
{A, B, C, D, E}											

i = 3

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[3, 6] = max {M[2, 6], 10 + M[2, 6 - 5]}
M[3, 6] = max {M[2, 6], 10 + M[2, 1]}
M[3, 6] = max {11, 10 + 0]} = 11
```

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

w = 6

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
i = 3 {A, B, C}	0	0	6	6	6	11	11				
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_3 > w ?
5 > 7 ?
NO

w = 7

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	?			
{A, B, C, D}											
{A, B, C, D, E}											

i = 3

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[3, 7] = max {M[2, 7], 10 + M[2, 7 - 5]}
M[3, 7] = max {M[2, 7], 10 + M[2, 2]}
M[3, 7] = max {11, 10 + 6]} = 16
```

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

`w = 7`

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16			
{A, B, C, D}											
{A, B, C, D, E}											

`i = 3`

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_3 > w ?
5 > 8 ?
NO

w = 8

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	?		
{A, B, C, D}											
{A, B, C, D, E}											

i = 3

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[3, 8] = max {M[2, 8], 10 + M[2, 8 - 5]}
M[3, 8] = max {M[2, 8], 10 + M[2, 3]}
M[3, 8] = max {11, 10 + 6]} = 16
```

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

w = 8

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
i = 3 {A, B, C}	0	0	6	6	6	11	11	16	16		
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_3 > w ?
5 > 9 ?
NO

w = 9

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	?	
{A, B, C, D}											
{A, B, C, D, E}											

i = 3

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[3, 9] = max {M[2, 9], 10 + M[2, 9 - 5]}
M[3, 9] = max {M[2, 9], 10 + M[2, 4]}
M[3, 9] = max {11, 10 + 6]} = 16
```

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

w = 9

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
i = 3 {A, B, C}	0	0	6	6	6	11	11	16	16	16	
{A, B, C, D}											
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_3 > w ?
5 > 10 ?
NO

w = 10

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	?
{A, B, C, D}											
{A, B, C, D, E}											

i = 3

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[3, 10] = max {M[2, 10], 10 + M[2, 10 - 5]}
M[3, 10] = max {M[2, 10], 10 + M[2, 5]}
M[3, 10] = max {11, 10 + 11]} = 21

Food	Weight	Energy
A	2	6
B	3	5
C	w_3 **5**	v_3 **10**
D	4	4
E	5	10

w = 10

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}											
{A, B, C, D, E}											

i = 3

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_4 > w ?
4 > 0 ?
YES

w = 0

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	?										
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[4, 0] = M[3, 0]

w = 0

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0										
{A, B, C, D, E}											

i = 4

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_4 > w ?
4 > 1 ?
YES

w = 1

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	0	?									
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[4, 1] = M[3, 1]

w = 1

i = 4

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0									
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_4 > w ?
4 > 2 ?
YES

w = 2

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	?								
{A, B, C, D, E}											

i = 4

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[4, 2] = M[3, 2]

w = 2

i = 4

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6								
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_4 > w ?
4 > 3 ?
YES

w = 3

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	0	0	6	?							
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[4, 3] = M[3, 3]

w = 3

i = 4

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6							
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

$w_4 > w$?
4 > 4 ?
NO

w = 4

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	0	0	6	6	?						
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[4, 4] = max {M[3, 4], 4 + M[3, 4 - 4]}
M[4, 4] = max {M[3, 4], 4 + M[3, 0]}
M[4, 4] = max {6, 4 + 0]} = 6

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

w = 4

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	0	0	6	6	6						
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_4 > w ?
4 > 5 ?
NO

w = 5

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	0	0	6	6	6	?					
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[4, 5] = max {M[3, 5], 4 + M[3, 5 - 4]}
M[4, 5] = max {M[3, 5], 4 + M[3, 1]}
M[4, 5] = max {11, 4 + 0]} = 11
```

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

w = 5

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	0	0	6	6	6	11					
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_4 > w ?
4 > 6 ?
NO

w = 6

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	0	0	6	6	6	11	?				
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[4, 6] = max {M[3, 6], 4 + M[3, 6 - 4]}
M[4, 6] = max {M[3, 6], 4 + M[3, 2]}
M[4, 6] = max {11, 4 + 6]} = 11

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

w = 6

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	0	0	6	6	6	11	11				
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_4 > w ?
4 > 7 ?
NO

w = 7

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	?			
{A, B, C, D, E}											

i = 4

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[4, 7] = max {M[3, 7], 4 + M[3, 7 - 4]}
M[4, 7] = max {M[3, 7], 4 + M[3, 3]}
M[4, 7] = max {16, 4 + 6]} = 16
```

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

w = 7

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	0	0	6	6	6	11	11	16			
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_4 > w ?
4 > 8 ?
NO

w = 8

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	0	0	6	6	6	11	11	16	?		
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[4, 8] = max {M[3, 8], 4 + M[3, 8 - 4]}
M[4, 8] = max {M[3, 8], 4 + M[3, 4]}
M[4, 8] = max {16, 4 + 6]} = 16

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

w = 8

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	0	0	6	6	6	11	11	16	16		
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_4 > w ?
4 > 9 ?
NO

w = 9

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	0	0	6	6	6	11	11	16	16	?	
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[4, 9] = max {M[3, 9], 4 + M[3, 9 - 4]}
M[4, 9] = max {M[3, 9], 4 + M[3, 5]}
M[4, 9] = max {16, 4 + 11]} = 16
```

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

w = 9

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	
{A, B, C, D, E}											

i = 4

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_4 > w ?
4 > 10 ?
NO

w = 10

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	?
{A, B, C, D, E}											

i = 4

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[4, 10] = max {M[3, 10], 4 + M[3, 10 - 4]}
M[4, 10] = max {M[3, 10], 4 + M[3, 6]}
M[4, 10] = max {21, 4 + 11]} = 21
```

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	w_4 **4**	v_4 **4**
E	5	10

w = 10

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
i = 4 {A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}											

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_5 > w ?
5 > 0 ?
YES

w = 0

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	?										

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[5, 0] = M[4, 0]

w = 0

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0										

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_5 > w ?
5 > 1 ?
YES

w = 1

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	?									

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

`M[5, 1] = M[4, 1]`

`w = 1`

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0									

`i = 5`

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (wi > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - wi]}
```

w_5 > w ?
5 > 2 ?
YES

w = 2

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	0	?								

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[5, 2] = M[4, 2]

w = 2

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	0	6								

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_5 > w ?
5 > 3 ?
YES

w = 3

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	0	6	?							

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

M[5, 3] = M[4, 3]

w = 3

i = 5

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0	6	6							

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_5 > w ?
5 > 4 ?
YES

w = 4

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	0	6	6	?						

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

`M[5, 4] = M[4, 4]`

`w = 4`

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
`i = 5` {A, B, C, D, E}	0	0	6	6	6						

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_5 > w ?
5 > 5 ?
NO

w = 5

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	0	6	6	6	?					

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[5, 5] = max {M[4, 5], 5 + M[4, 5 - 5]}
M[5, 5] = max {M[4, 5], 5 + M[4, 0]}
M[5, 5] = max {11, 10 + 0]} = 11
```

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

w = 5

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	0	6	6	6	11					

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_5 > w ?
5 > 6 ?
NO

w = 6

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	0	6	6	6	11	?				

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[5, 6] = max {M[4, 6], 5 + M[4, 6 - 5]}
M[5, 6] = max {M[4, 6], 5 + M[4, 1]}
M[5, 6] = max {11, 10 + 0]} = 11
```

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

w = 6

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	0	6	6	6	11	11				

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_5 > w ?
5 > 7 ?
NO

w = 7

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	0	6	6	6	11	11	?			

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[5, 7] = max {M[4, 7], 5 + M[4, 7 - 5]}
M[5, 7] = max {M[4, 7], 5 + M[4, 2]}
M[5, 7] = max {16, 10 + 6]} = 16
```

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

w = 7

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	0	6	6	6	11	11	16			

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_5 > w ?
5 > 8 ?
NO

w = 8

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	0	6	6	6	11	11	16	?		

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[5, 8] = max {M[4, 8], 5 + M[4, 8 - 5]}
M[5, 8] = max {M[4, 8], 5 + M[4, 3]}
M[5, 8] = max {16, 10 + 6]} = 16
```

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

`w = 8`

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
`i = 5` {A, B, C, D, E}	0	0	6	6	6	11	11	16	16		

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

w_5 > w ?
5 > 9 ?
NO

w = 9

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0	6	6	6	11	11	16	16	?	

i = 5

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[5, 9] = max {M[4, 9], 5 + M[4, 9 - 5]}
M[5, 9] = max {M[4, 9], 5 + M[4, 4]}
M[5, 9] = max {16, 10 + 6]} = 16
```

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

w = 9

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	0	6	6	6	11	11	16	16	16	

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

```
for i = 1 to n
  for w = 0 to W
    if (wi > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - wi]}
```

w_5 > w ?
5 > 10 ?
NO

w = 10

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0	6	6	6	11	11	16	16	16	?

i = 5

A knapsack practice: fill the DP table

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
for i = 1 to n
  for w = 0 to W
    if (w_i > w)
      M[i, w] = M[i - 1, w]
    else
      M[i, w] = max {M[i - 1, w], vi + M[i - 1, w - w_i]}
```

```
M[5, 10] = max {M[4, 10], 5 + M[4, 10 - 5]}
M[5, 10] = max {M[4, 10], 5 + M[4, 5]}
M[5, 10] = max {21, 10 + 11]} = 16
```

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	w_5 **5**	v_5 **10**

w = 10

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
i = 5 {A, B, C, D, E}	0	0	6	6	6	11	11	16	16	16	21

A knapsack practice: optimal solution

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	5	10

The amount of energy we could gain is **21**.

Which foods did we select?

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0	6	6	6	11	11	16	16	16	21

A knapsack practice: optimal selection of foods

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	**5**	**10**

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
  if (M[i, k] != M[i - 1, k])
     print(i)
     k = k - w_i
  i = i - 1
```

`M[5, 10] != M[4, 10] ? NO`

`w = 10`

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0	6	6	6	11	11	16	16	16	21

`i = 5`

A knapsack practice: optimal selection of foods

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	**4**	**4**
E	5	10

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
  if (M[i, k] != M[i - 1, k])
     print(i)
     k = k - w_i
  i = i - 1
```

`M[4, 10] != M[3, 10] ? NO`

`w = 10`

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0	6	6	6	11	11	16	16	16	21

`i = 4`

A knapsack practice: optimal selection of foods

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
  if (M[i, k] != M[i - 1, k])
    print(i)
    k = k - w_i
  i = i - 1
```

M[3, 10] != M[2, 10] ? YES

Food	Weight	Energy
A	2	6
B	3	5
C	**5**	**10**
D	4	4
E	5	10

w = 10

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0	6	6	6	11	11	16	16	16	21

i = 3

A knapsack practice: optimal selection of foods

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	w_i **5**	**10**
D	4	4
E	5	10

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
  if (M[i, k] != M[i - 1, k])
    print(i)
    k = k - w_i
  i = i - 1
```

M[3, 10] != M[2, 10] ? YES

print(3)

k = k - w_i = 10 − 5 = 5

Foods in the knapsack: "C"

w = 10

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0	6	6	6	11	11	16	16	16	21

i = 3

A knapsack practice: optimal selection of foods

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	**3**	**5**
C	5	10
D	4	4
E	5	10

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
  if (M[i, k] != M[i - 1, k])
     print(i)
     k = k - w_i
   i = i - 1
```

M[2, 5] != M[1, 5] ? YES

Foods in the knapsack: "C"

w = 5

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
i = 2 {A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0	6	6	6	11	11	16	16	16	21

A knapsack practice: optimal selection of foods

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	w_i 3	5
C	5	10
D	4	4
E	5	10

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
  if (M[i, k] != M[i - 1, k])
    print(i)
    k = k - w_i
  i = i - 1
```

M[2, 5] != M[1, 5] ? YES

print(2)

$k = k - w_i = 5 - 3 = 2$

Foods in the knapsack: "C" "B"

w = 5

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
i = 2 {A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0	6	6	6	11	11	16	16	16	21

A knapsack practice: optimal selection of foods

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	5	10

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
  if (M[i, k] != M[i - 1, k])
     print(i)
     k = k - w_i
   i = i - 1
```

M[1, 2] != M[0, 2] ? YES

Foods in the knapsack: "C" "B"

w = 2

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0	6	6	6	11	11	16	16	16	21

i = 1

A knapsack practice: optimal selection of foods

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	w_i 2	6
B	3	5
C	5	10
D	4	4
E	5	10

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
  if (M[i, k] != M[i - 1, k])
    print(i)
    k = k - w_i
  i = i - 1
```

M[1, 2] != M[0, 2] ? YES
print(1)
k = k - w_i = 2 − 2 = 0

Foods in the knapsack: "C" "B" "A"

w = 2

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
i = 1 {A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0	6	6	6	11	11	16	16	16	21

A knapsack practice: optimal selection of foods

Aim: Maximize the amount of energy
Criterion: We can take max. 10 kgs

Food	Weight	Energy
A	2	6
B	3	5
C	5	10
D	4	4
E	5	10

```
Input: M, n, w_1, ..., w_N, v_1, ..., v_n
i = n, k = W
while i, k > 0
  if (M[i, k] != M[i - 1, k])
    print(i)
    k = k - w_i                    k & i = 0
  i = i - 1
```

Foods in the knapsack: "C" "B" "A"

w = 0

i = 0

Foods\Weights	0	1	2	3	4	5	6	7	8	9	10
{ }	0	0	0	0	0	0	0	0	0	0	0
{A}	0	0	6	6	6	6	6	6	6	6	6
{A, B}	0	0	6	6	6	11	11	11	11	11	11
{A, B, C}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D}	0	0	6	6	6	11	11	16	16	16	21
{A, B, C, D, E}	0	0	6	6	6	11	11	16	16	16	21