Dinamičko programiranje

Problem ranca

Problem ranca

- Provalnik ima ranac zapremine **N**. Pljačka prostoriju u kojoj se nalazi **M** različitih vrsta vrednosnih predmeta.
- Za svaki predmet je data njegova vrednost **vred[i]** i njegova zapremina **zapr[i]**, i=1,...,M. Broj primeraka svake vrste predmeta je neograničen.
- Potrebno je popuniti ranac predmetima tako da vrednost predmeta u rancu bude najveća moguća.

Korak 1 – struktura rešenja

- Potproblemi – rančevi različitih zapremina
- $pred_1, pred_2, \dots, pred_i, \dots, pred_m$
- $ranac_1, ranac_2, \dots, ranac_j, \dots, ranac_n$
- Ako stavimo predmet i u ranac zapremine j, dodajemo njegovu vrednost na optimalno popunjen ranac zapremine $j - zapr_i$
- Biramo predmet koji daje najveću vrednost popunjenog ranca j.

Korak 2 - rekurzija

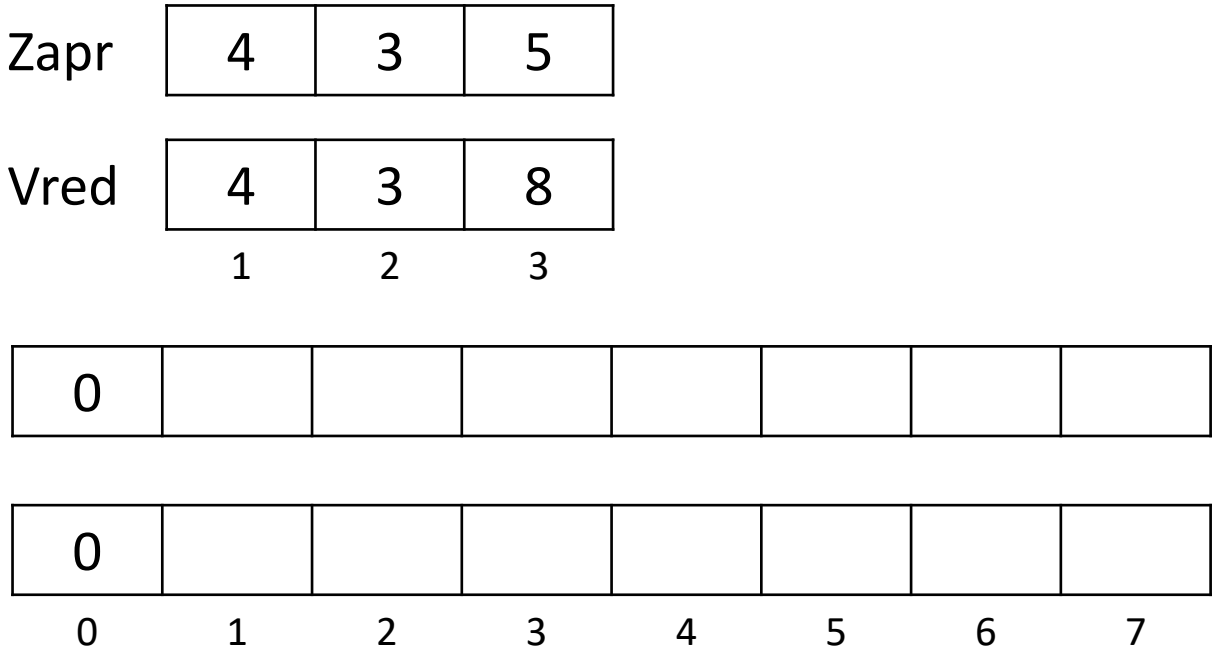
- $opt(j) = \max(vred_i + opt(j - zapr_i))$
  - $1 \leq i \leq m$
  - $zapr_i \leq j$

- $opt(0) = 0$

Korak 3 – bottom-up

- `N = 7; M = 3;`

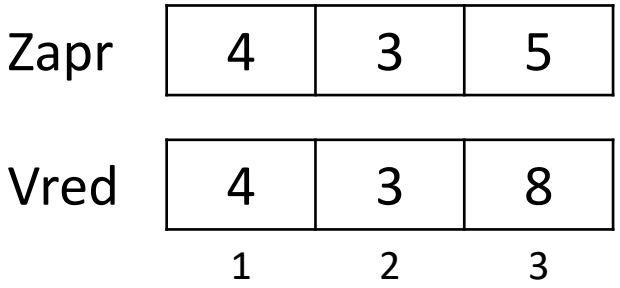
```
opt[0] = 0;
```

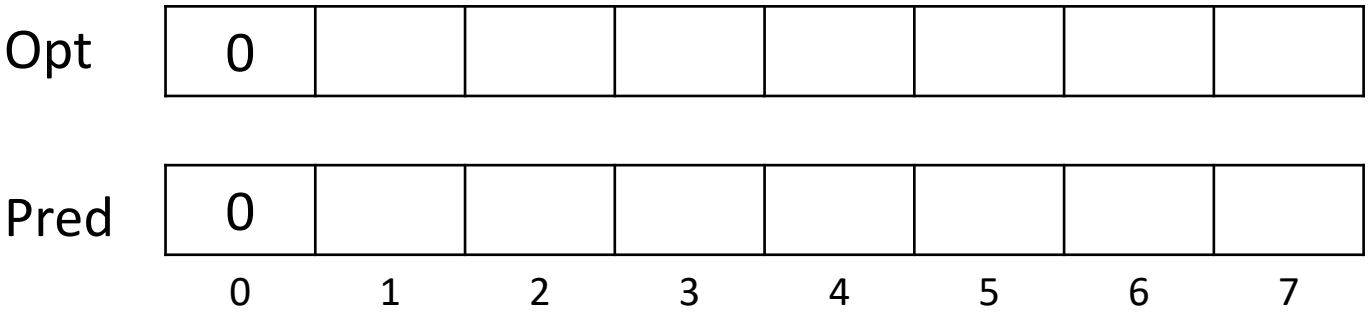


Korak 3 – bottom-up

- `N = 7; M = 3;`

```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         …
```

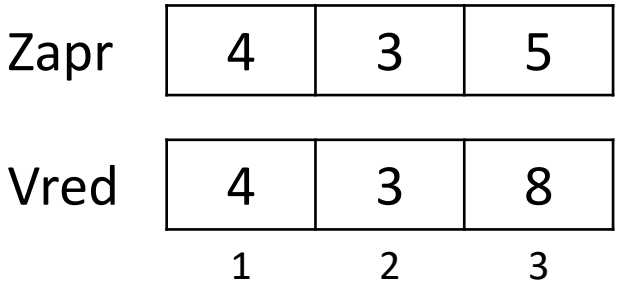


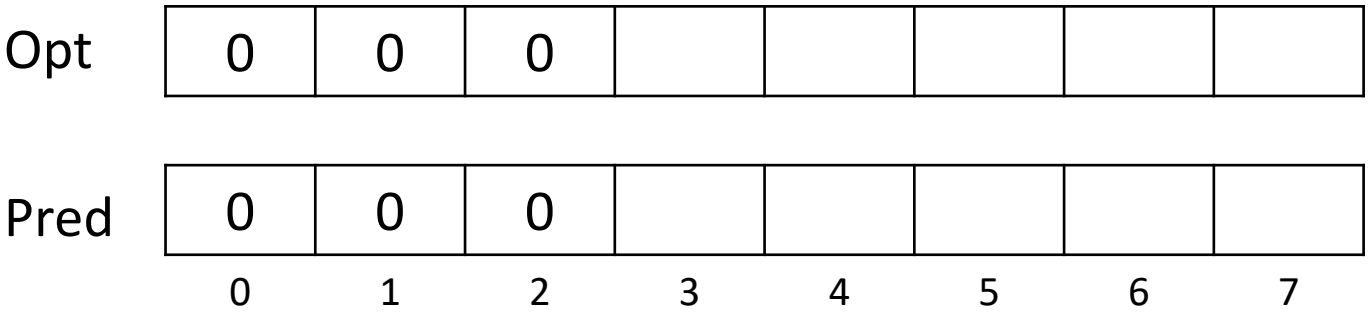


Korak 3 – bottom-up

- `N = 7; M = 3;`

```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         …
```





Korak 3 – bottom-up

- `N = 7; M = 3;`

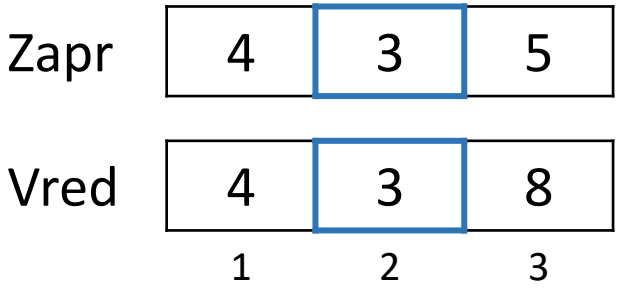
Zapr	4	3	5
Vred	4	3	8
	1	2	3

```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         opt[j] = max(opt[j],
                      vred[i]+opt[j-zapr[i]])
```

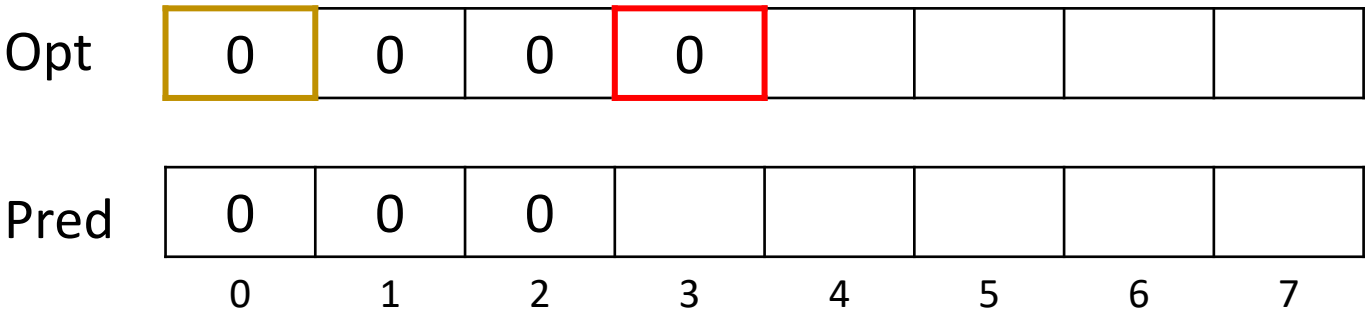
Opt	0	0	0	0				
Pred	0	0	0					
	0	1	2	3	4	5	6	7

Korak 3 – bottom-up

- `N = 7; M = 3;`

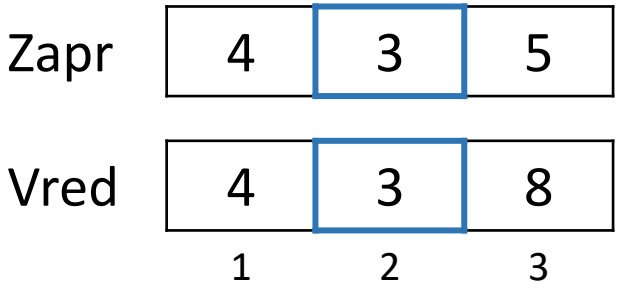



```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```

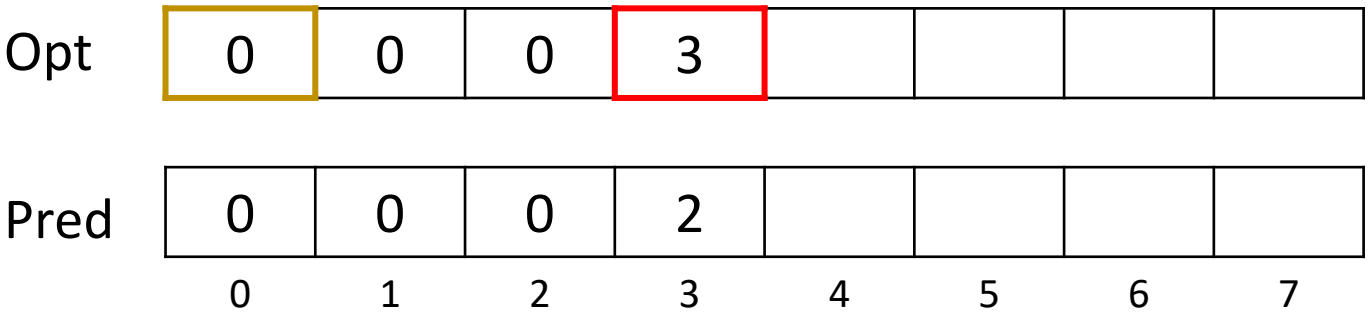


Korak 3 – bottom-up

- `N = 7; M = 3;`

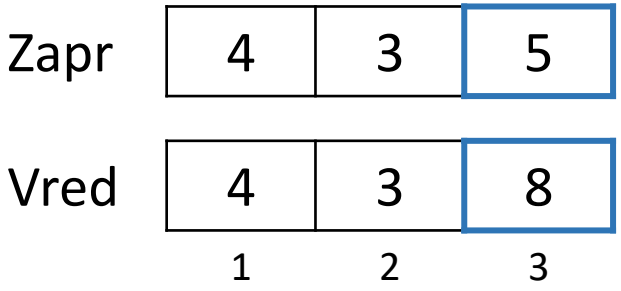



```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```

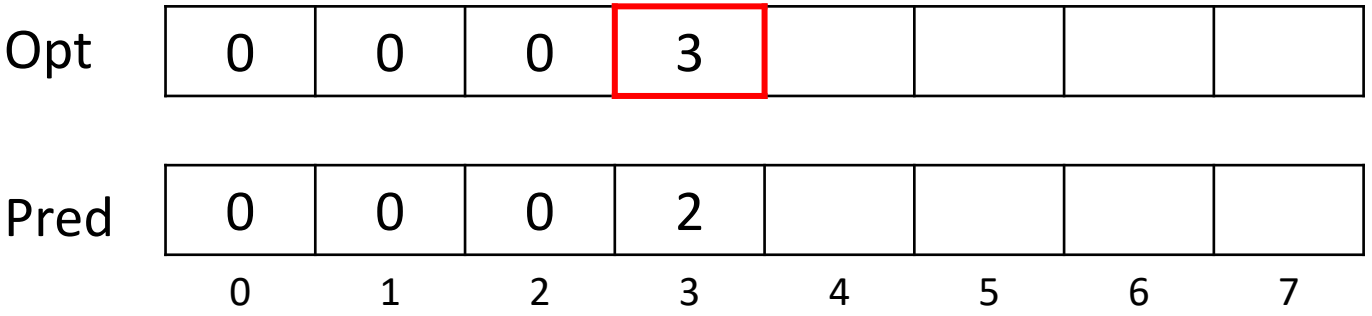


Korak 3 – bottom-up

- `N = 7; M = 3;`

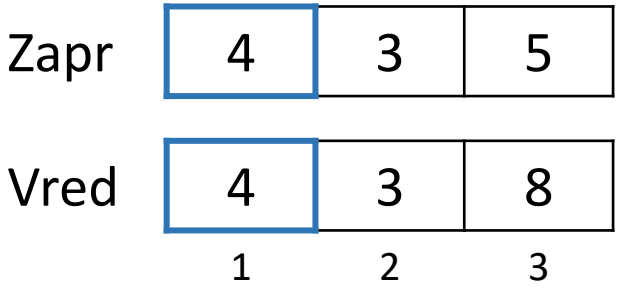



```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         opt[j] = max(opt[j],
                      vred[i]+opt[j-zapr[i]])
```

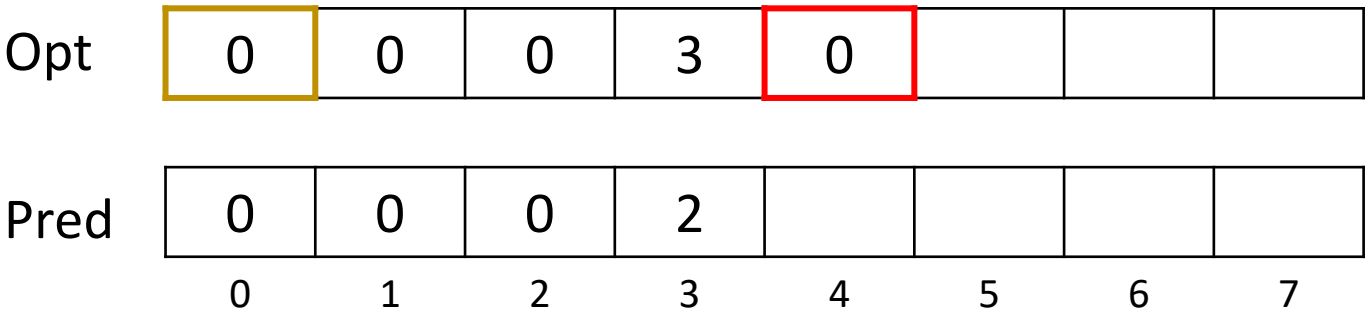


Korak 3 – bottom-up

- `N = 7; M = 3;`

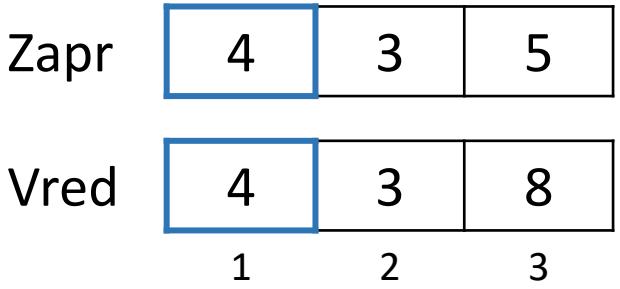



```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```

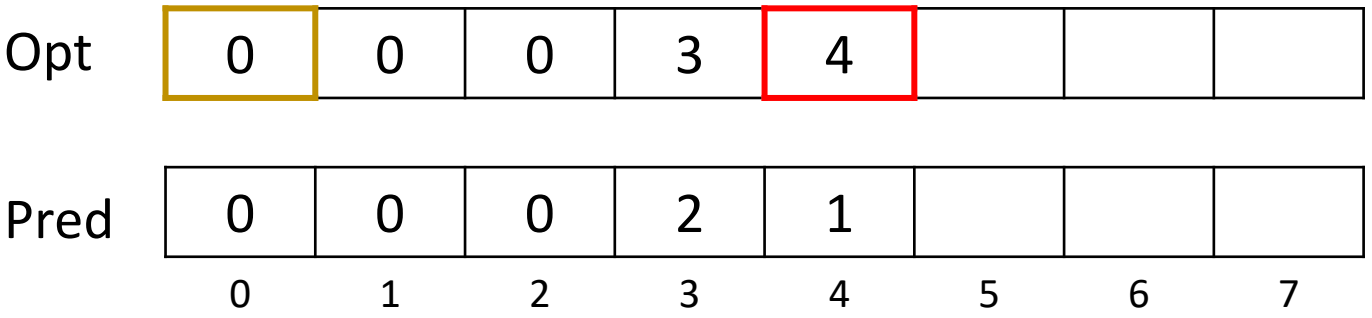


Korak 3 – bottom-up

- `N = 7; M = 3;`

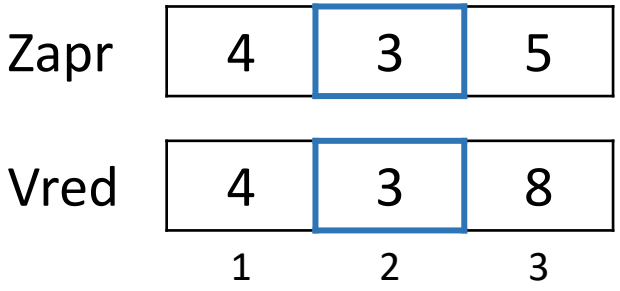



```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```

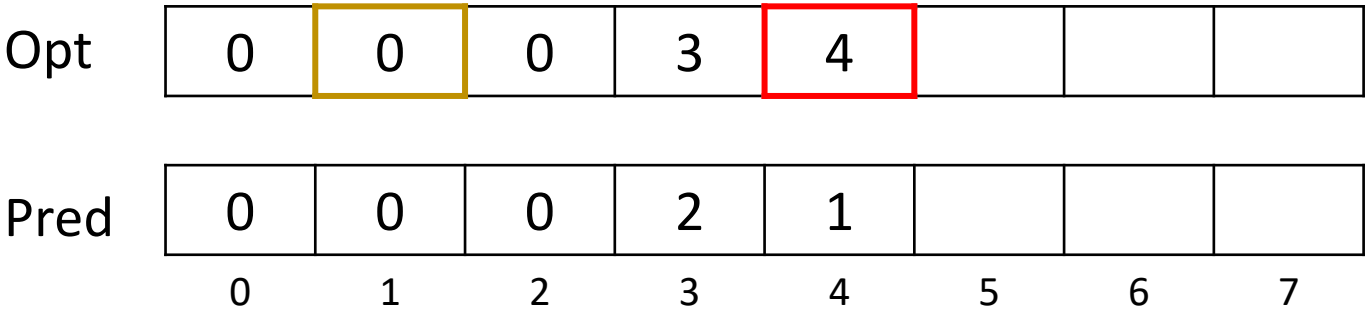


Korak 3 – bottom-up

- `N = 7; M = 3;`

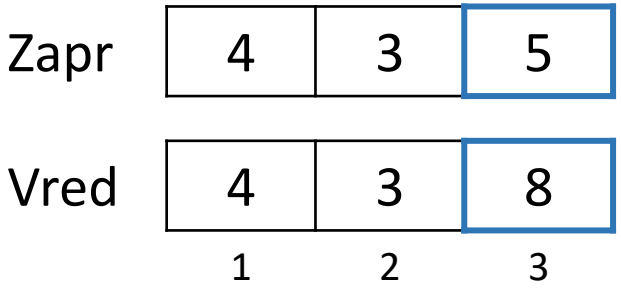



```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```

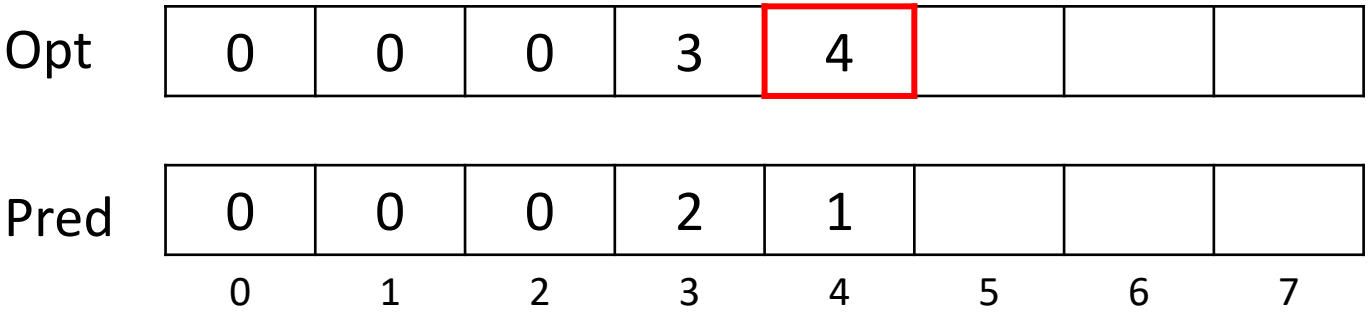


Korak 3 – bottom-up

- `N = 7; M = 3;`

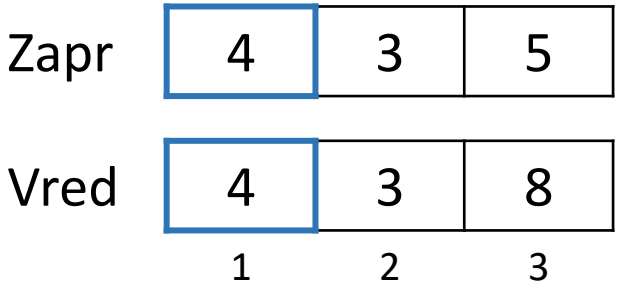



```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         opt[j] = max(opt[j],
                      vred[i]+opt[j-zapr[i]])
```

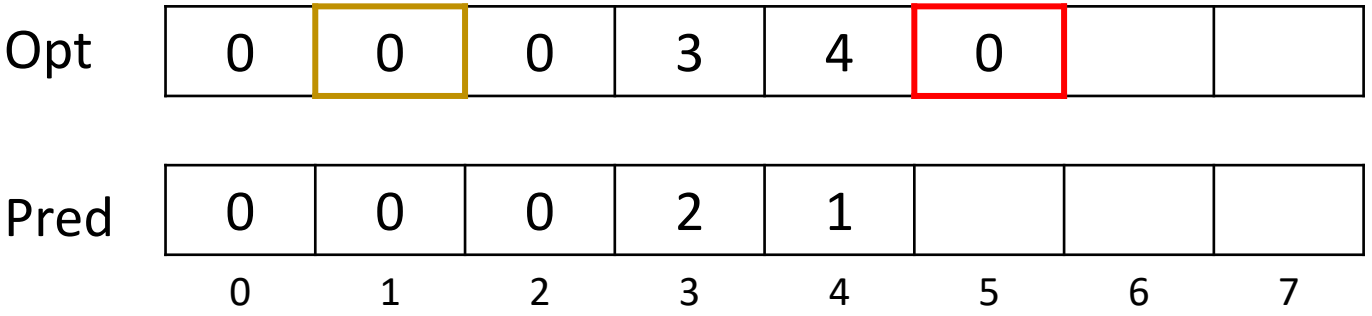


Korak 3 – bottom-up

- `N = 7; M = 3;`




```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```



Korak 3 – bottom-up

- `N = 7; M = 3;`

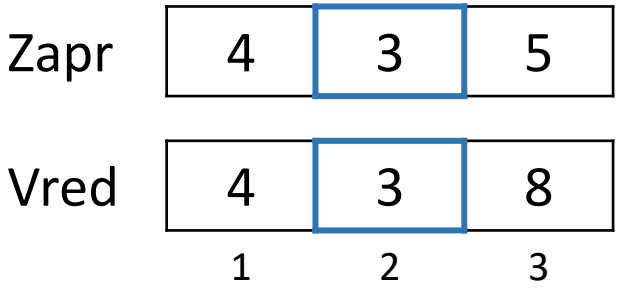
Zapr	4	3	5
Vred	4	3	8
	1	2	3

```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```

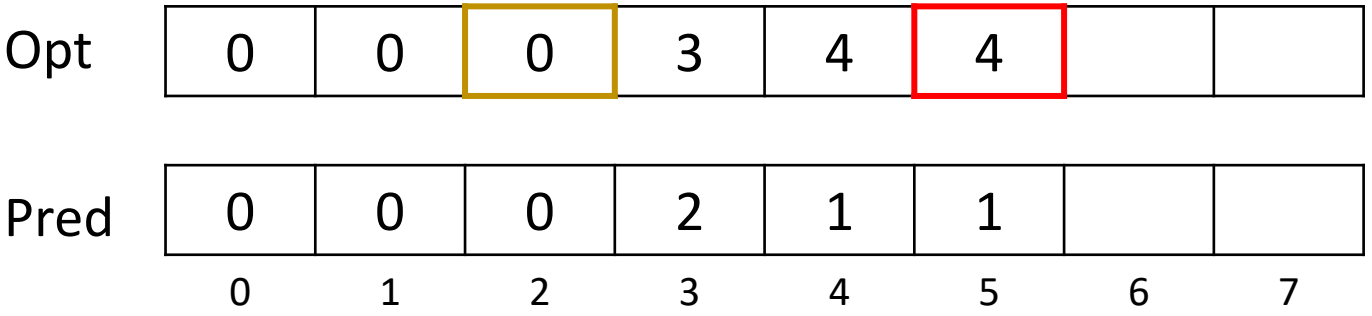
Opt	0	0	0	3	4	4		
Pred	0	0	0	2	1	1		
	0	1	2	3	4	5	6	7

Korak 3 – bottom-up

- `N = 7; M = 3;`

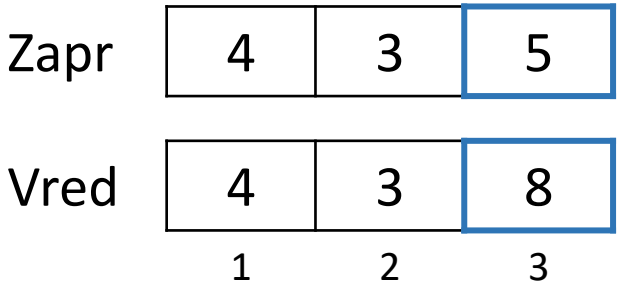



```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```

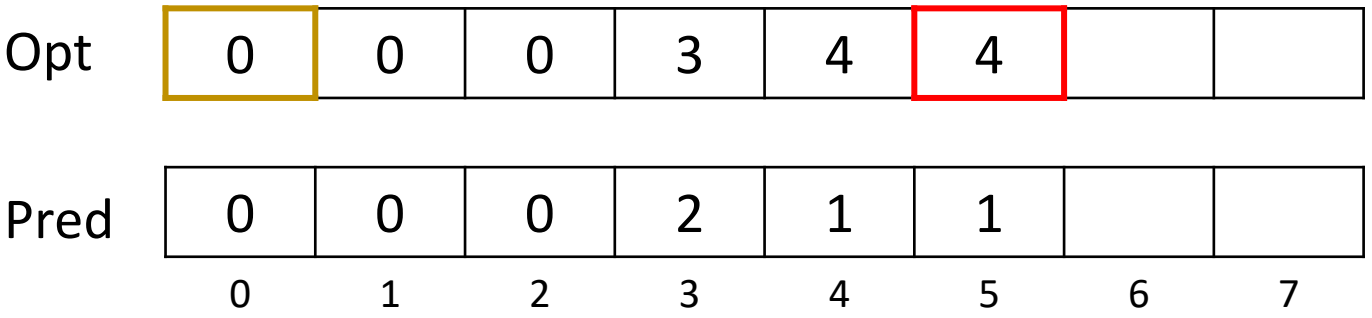


Korak 3 – bottom-up

- `N = 7; M = 3;`

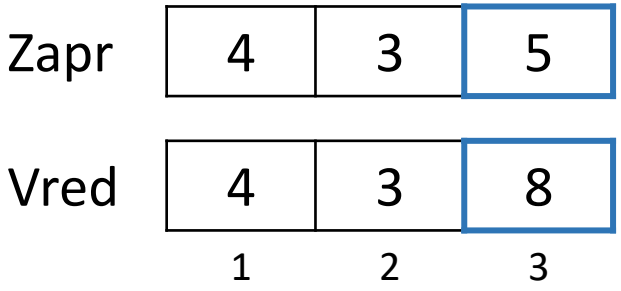



```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         opt[j] = max(opt[j],
                      vred[i]+opt[j-zapr[i]])
```

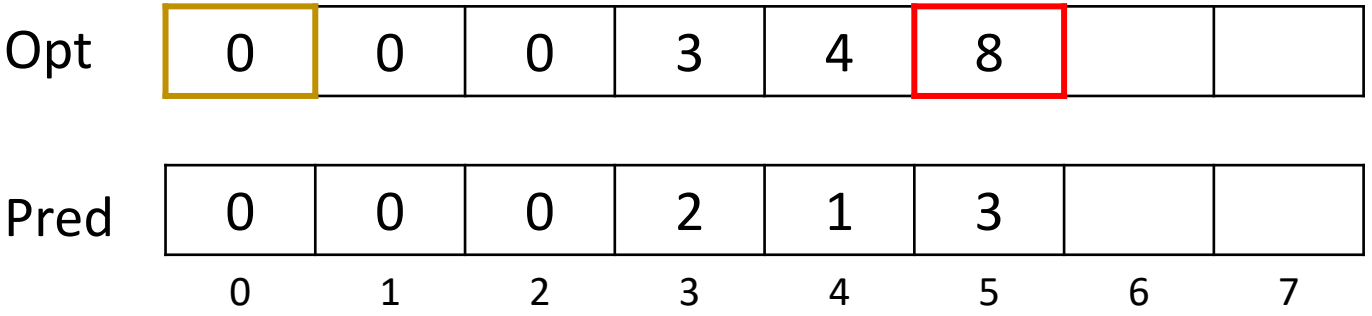


Korak 3 – bottom-up

- `N = 7; M = 3;`




```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```



Korak 3 – bottom-up

- `N = 7; M = 3;`

	1	2	3
Zapr	4	3	5
Vred	4	3	8

```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         opt[j] = max(opt[j],
                      vred[i]+opt[j-zapr[i]])
```

	0	1	2	3	4	5	6	7
Opt	0	0	0	3	4	8	0	
Pred	0	0	0	2	1	3		

Korak 3 – bottom-up

- `N = 7; M = 3;`

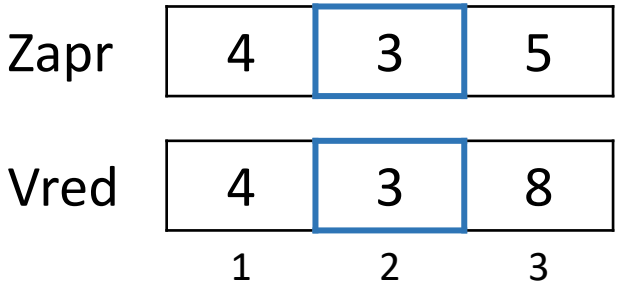
	1	2	3
Zapr	4	3	5
Vred	4	3	8

```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         opt[j] = max(opt[j],
                      vred[i]+opt[j-zapr[i]])
```

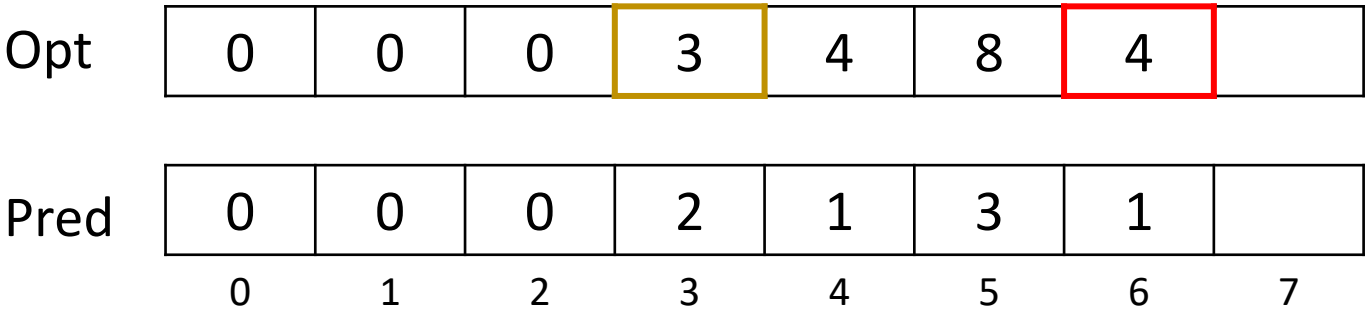
	0	1	2	3	4	5	6	7
Opt	0	0	0	3	4	8	4	
Pred	0	0	0	2	1	3	1	

Korak 3 – bottom-up

- `N = 7; M = 3;`

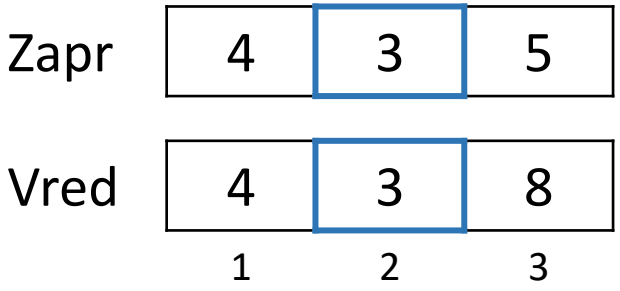



```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```

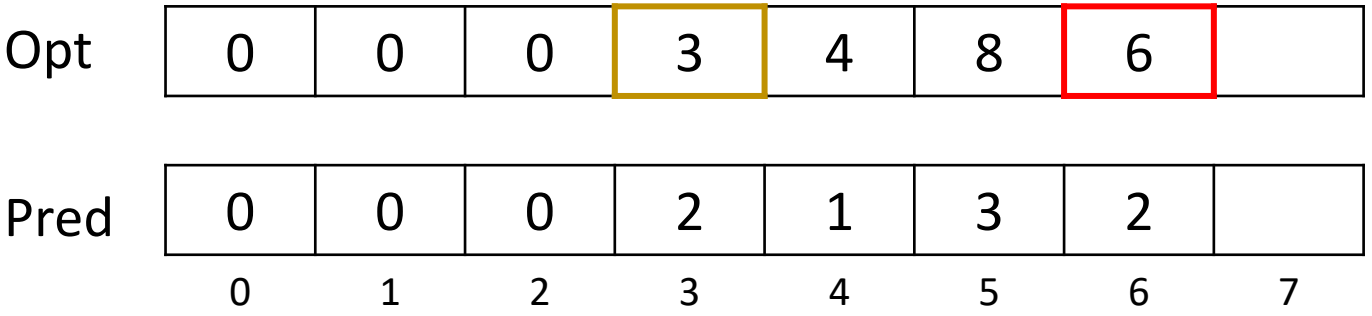


Korak 3 – bottom-up

- `N = 7; M = 3;`




```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```



Korak 3 – bottom-up

- `N = 7; M = 3;`

Zapr	4	3	5
Vred	4	3	8
	1	2	3

```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         opt[j] = max(opt[j],
                      vred[i]+opt[j-zapr[i]])
```

Opt	0	0	0	3	4	8	6	
Pred	0	0	0	2	1	3	2	
	0	1	2	3	4	5	6	7

Korak 3 – bottom-up

- `N = 7; M = 3;`

Zapr	4	3	5
Vred	4	3	8
	1	2	3

```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         opt[j] = max(opt[j],
                      vred[i]+opt[j-zapr[i]])
```

Opt	0	0	0	3	4	8	8	
Pred	0	0	0	2	1	3	3	
	0	1	2	3	4	5	6	7

Korak 3 – bottom-up

- `N = 7; M = 3;`

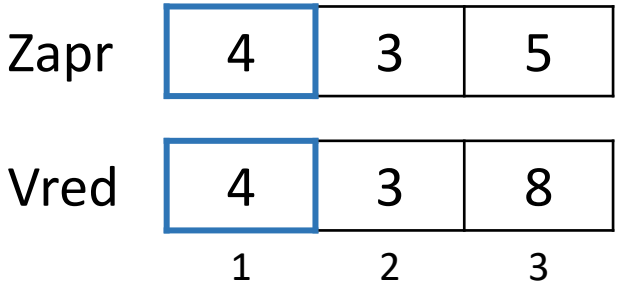
Zapr	4	3	5
Vred	4	3	8
	1	2	3

```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         opt[j] = max(opt[j],
                      vred[i]+opt[j-zapr[i]])
```

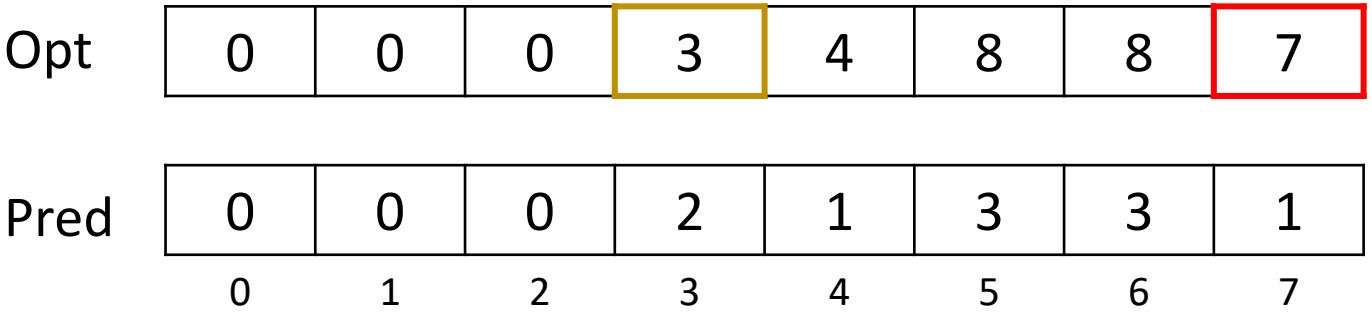
Opt	0	0	0	3	4	8	8	0
Pred	0	0	0	2	1	3	3	
	0	1	2	3	4	5	6	7

Korak 3 – bottom-up

- `N = 7; M = 3;`

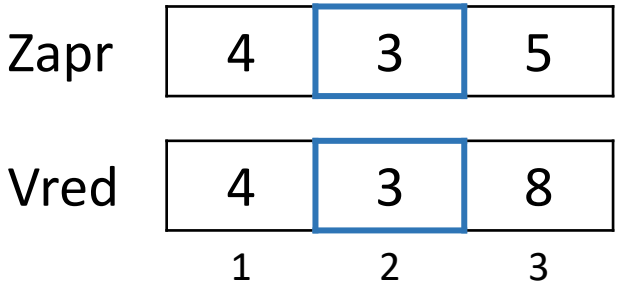



```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```

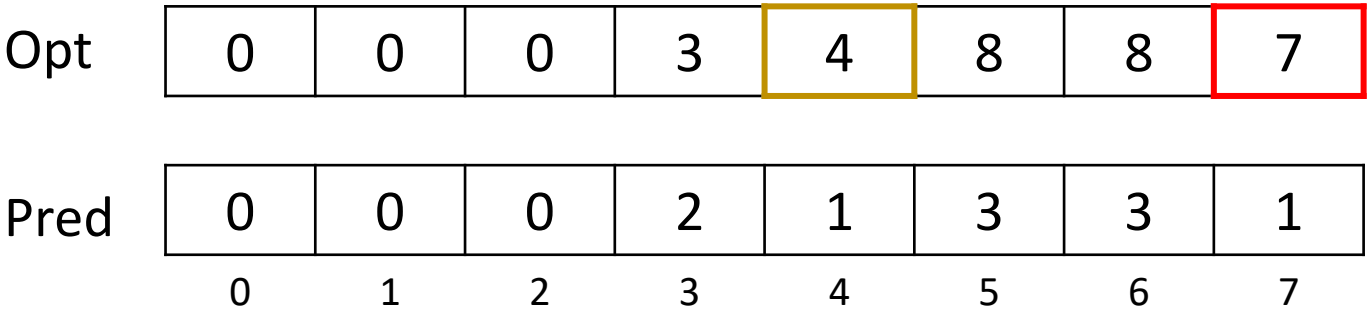


Korak 3 – bottom-up

- `N = 7; M = 3;`




```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```



Korak 3 – bottom-up

- `N = 7; M = 3;`

	1	2	3
Zapr	4	3	5
Vred	4	3	8

```
opt[0] = 0;
for j = 1,n
    opt[j] = 0;
    for i = 1,m
        if (zapr[i] <= j)
            opt[j] = max(opt[j],
                         vred[i]+opt[j-zapr[i]])
```

	0	1	2	3	4	5	6	7
Opt	0	0	0	3	4	8	8	7
Pred	0	0	0	2	1	3	3	1

Korak 3 – bottom-up

- `N = 7; M = 3;`

Zapr	4	3	5
Vred	4	3	8
	1	2	3

```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         opt[j] = max(opt[j],
                      vred[i]+opt[j-zapr[i]])
```

Opt	0	0	0	3	4	8	8	8
Pred	0	0	0	2	1	3	3	3
	0	1	2	3	4	5	6	7

Korak 3 – bottom-up

- `N = 7; M = 3;`

Zapr	4	3	5
Vred	4	3	8
	1	2	3

```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         opt[j] = max(opt[j],
                      vred[i]+opt[j-zapr[i]])
```

Opt	0	0	0	3	4	8	8	8
Pred	0	0	0	2	1	3	3	3
	0	1	2	3	4	5	6	7

optimalno rešenje, maksimalna vrednost u rancu

Korak 4 – rekonstrukcija

- `N = 7; M = 3;`

```
i = n;
while (pred[i]>0)
    print(pred[i]);
    i -= zapr[pred[i]];
```

	1	2	3
Zapr	4	3	5
Vred	4	3	8

	0	1	2	3	4	5	6	7
Opt	0	0	0	3	4	8	8	8
Pred	0	0	0	2	1	3	3	3

Korak 4 – rekonstrukcija

- `N = 7; M = 3;`

```
i = n;
while (pred[i]>0)
    print(pred[i]);
    i -= zapr[pred[i]];
```

	1	2	3
Zapr	4	3	5
Vred	4	3	8

	0	1	2	3	4	5	6	7
Opt	0	0	0	3	4	8	8	8
Pred	0	0	0	2	1	3	3	3

Korak 4 – rekonstrukcija

- `N = 7; M = 3;`

```
i = n;
while (pred[i]>0)
    print(pred[i]);
    i -= zapr[pred[i]];
```

Zapr	4	3	5
Vred	4	3	8
	1	2	3

Opt	0	0	0	3	4	8	8	8
Pred	0	0	0	2	1	3	3	3
	0	1	2	3	4	5	6	7

Predmeti u rancu:

3

Test primer 2

- N = 7; M = 3;

Zapr	4	3	5
Vred	4	3	6
	1	2	3

```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         opt[j] = max(opt[j],
                      vred[i]+opt[j-zapr[i]])
```

Opt	0							
Pred	0							
	0	1	2	3	4	5	6	7

Test primer 2

- `N = 7; M = 3;`

Zapr	4	3	5
Vred	4	3	6
	1	2	3

```
opt[0] = 0;
for j = 1,n
   opt[j] = 0;
   for i = 1,m
      if (zapr[i] <= j)
         opt[j] = max(opt[j],
                      vred[i]+opt[j-zapr[i]])
```

Opt	0	0	0	3	4	6	6	7
Pred	0	0	0	2	1	3	2	1
	0	1	2	3	4	5	6	7

Test primer 2

- `N = 7; M = 3;`

```
while (pred[i]>0)
    print(pred[i]);
    i -= zapr[pred[i]];
```

Zapr	4	3	5
Vred	4	3	6
	1	2	3

Opt	0	0	0	3	4	6	6	7
Pred	0	0	0	2	1	3	2	1
	0	1	2	3	4	5	6	7

Predmeti u rancu:

1, 2

Problem ranca – varijanta II

- Od svake vrste predmeta postoji po jedan primerak.
- Dimenzije problema – zapremina i predmeti
- Potproblem $(i,\ j)$
  - Iskorišćeni predmeti do i
  - Popunjena zapremina do j

1. Dodajemo predmet i
   - na zapreminu $j - zapr_i$
   - popunjenu predmetima od 1 do $i - 1$
2. Ne dodajemo predmet i
   - Zadržavamo optimalno $(i - 1, j)$

Problem ranca – varijanta II

- $opt(i,\ j) = \max\begin{pmatrix} vred_i + opt(i-1,\ j - zapr_i), \\ opt(i-1,\ j) \end{pmatrix}$
  - $zapr_i \leq j$

- $opt(0, j) = 0$
- $opt(i, 0) = 0$

Problem ranca – varijanta II

- `N = 7; M = 3;`

Zapr	3	4	5
Vred	3	4	6
	1	2	3

```
for i = 1,m
   for j = 1,n
      if (zapr[i]<=j)
         opt[i][j] = max(opt[i-1][j-zapr[i]]+vred[i],
                         opt[i-1][j])
      else
         opt[i][j] = opt[i-1][j];
```

Opt

0	0	0	0	0	0	0	0	0
1	0							
2	0							
3	0							
	0	1	2	3	4	5	6	7

Problem ranca – varijanta II

- `N = 7; M = 3;`

Zapr	3	4	5
Vred	3	4	6
	1	2	3

```
for i = 1,m
   for j = 1,n
      if (zapr[i]<=j)
         opt[i][j] = max(opt[i-1][j-zapr[i]]+vred[i],
                         opt[i-1][j])
      else
         opt[i][j] = opt[i-1][j];
```

Opt

0	0	0	0	0	0	0	0	0
1	0	0	0	3	3	3	3	3
2	0	0	0	3	4	4	4	7
3	0	0	0	3	4	6	6	7
	0	1	2	3	4	5	6	7

optimalno rešenje, maksimalna vrednost u rancu

Problem ranca – varijanta II

- N = 7; M = 3;

Zapr	3	4	5
Vred	3	4	6
	1	2	3

```
i=m; j=n;
   while (opt[i][j]!=0)
      if (opt[i][j] == opt[i-1][j])
         i--;
      else
         print(i);
         j -= zapr[i];
         i--;
```

Opt

0	0	0	0	0	0	0	0	0
1	0	0	0	3	3	3	3	3
2	0	0	0	3	4	4	4	7
3	0	0	0	3	4	6	6	7
	0	1	2	3	4	5	6	7

Problem ranca – varijanta II

- `N = 7; M = 3;`

Zapr	3	4	5
Vred	3	4	6
	1	2	3

```
i=m; j=n;
   while (opt[i][j]!=0)
      if (opt[i][j] == opt[i-1][j])
         i--;
      else
         print(i);
         j -= zapr[i];
         i--;
```

Opt

0	0	0	0	0	0	0	0	0
1	0	0	0	3	3	3	3	3
2	0	0	0	3	4	4	4	7
3	0	0	0	3	4	6	6	7
	0	1	2	3	4	5	6	7

Predmeti u rancu:

1, 2