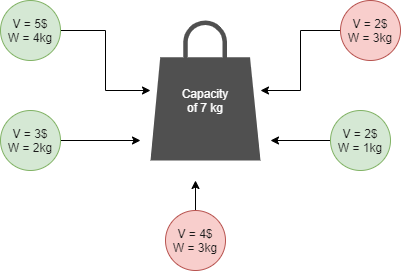
**Knapsack**

 **Example1**

*Item = 1 2 3 4 5*

V[] ={2, 2, 4, 5, 3}

W[] ={3, 1, 3, 4, 2}

The rows are representing one item.

The columns are representing the Knapsack capacity.

int[][] DP = new int[N + 1][capacity + 1];

for (int i = 1; i <= N; i++) { DP[0][i] = 0;}

for (int i = 1; i <= N; i++) {

int w = W[i - 1]

int v = V[i - 1];

for (int c = 1; c <= capacity; c++) {

DP[i][c] = DP[i - 1][c];

if (c >= w && DP[i - 1][c - w] + v > DP[i][c]) DP[i][c] = DP[i - 1][c - w] + v;

}}

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i\c | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 Empty | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1(v=2, w=3) | 0 | 0 | 0 | 2 | 2 | 2 | 2 | 2 |
| 2(v=2, w=1) | 0 | 2 | 2 | 2 | 4 | 4 | 4 | 4 |
| 3(v=4, w=3) | 0 | 2 | 2 | 4 | 6 | 6 | 6 | 8 |
| 4(v=5, w=4) | 0 | 2 | 2 | 4 | 6 | 7 | 7 | 9 |
| 5(v=3, w=2) | 0 | 2 | 3 | 5 | 6 | 7 | 9 | 10 |

First step: i=1 => w = 3, v= 2

*Capacity DP[i][c] = DP[i-1][c] c >= w && DP[i - 1][c - w] + v > DP[i][c] DP[i][c] = DP[i-1][c-w]+v*

* c=1 => DP[1][1] = DP[0][1] = 0; 1>=3 – False *(c >= w)*
* c=2 => DP[1][2] = DP[0][2] = 0; 2>=3 – False
* c=3 => DP[1][3] = DP[0][3] = 0; 3>=3 – True && DP[0][0] + v > DP[1][3] *(0+2>0)* – True => DP[1][3]=DP[0][0]+v =2
* c=4 => DP[1][4] = DP[0][4] = 0; 4>=3 – True && DP[0][1] + v > DP[1][4] *(0+2>0)* – True => DP[1][4]=DP[0][1]+v =2
* c=5 => DP[1][5] = DP[0][5] = 0; 5>=3 – True && DP[0][2] + v > DP[1][5] *(0+2>0)* – True => DP[1][5]=DP[0][2]+v =2
* c=6 => DP[1][6] = DP[0][6] = 0; 6>=3 – True && DP[0][3] + v > DP[1][6] *(0+2>0)* – True => DP[1][6]=DP[0][3]+v =2
* c=7 => DP[1][7] = DP[0][7] = 0; 7>=3 – True && DP[0][4] + v > DP[1][7] *(0+2>0)* – True => DP[1][7]=DP[0][4]+v =2

Step i=2 => w = 1, v= 2

*Capacity DP[i][c] = DP[i-1][c] c >= w && DP[i - 1][c - w] + v > DP[i][c] DP[i][c] = DP[i-1][c-w]+v*

* c=1 => DP[2][1] = DP[1][1] = 0; 1>=1 – True && DP[1][0] + v > DP[2][1] *(0+2>0)* – True => DP[2][1]=DP[1][0]+v =2
* c=2 => DP[2][2] = DP[1][2] = 0; 2>=1 – True && DP[1][1] + v > DP[2][2] *(0+2>0)* – True => DP[2][2]=DP[1][1]+v =2
* c=3 => DP[2][3] = DP[1][3] = 2; 3>=1 – True && DP[1][2] + v > DP[2][3] *(0+2>2)* – False
* c=4 => DP[2][4] = DP[1][4] = 2; 4>=1 – True && DP[1][3] + v > DP[2][4] *(2+2>2)* – True => DP[2][4]=DP[1][3]+v =4
* c=5 => DP[2][5] = DP[1][5] = 2; 5>=1 – True && DP[1][4] + v > DP[2][5] *(2+2>2)* – True => DP[2][5]=DP[1][5]+v =4

same for c = 6,7

Step i=3 => w = 3, v= 4

*Capacity DP[i][c] = DP[i-1][c] c >= w && DP[i - 1][c - w] + v > DP[i][c] DP[i][c] = DP[i-1][c-w]+v*

* c=1 => DP[3][1] = DP[2][1] = 2; 1>=3 – False
* c=2 => DP[3][2] = DP[2][2] = 2; 2>=3 – False
* c=3 => DP[3][3] = DP[2][3] = 2; 3>=3 – True && DP[2][0] + v > DP[3][3] *(0+4>2)* – True => DP[3][3]=DP[2][0]+v =4
* c=4 => DP[3][4] = DP[2][4] = 4; 4>=3 – True && DP[2][1] + v > DP[3][4] *(2+4>4)* – True => DP[3][4]=DP[2][1]+v =6
* same for c = 5,6
* c=7 => DP[3][7] = DP[2][7] = 4; 7>=3 – True && DP[2][4] + v > DP[3][7] *(4+4>4)* – True => DP[3][7]=DP[2][4]+v =8

Step i=4 => w = 4, v= 5

*Capacity DP[i][c] = DP[i-1][c] c >= w && DP[i - 1][c - w] + v > DP[i][c] DP[i][c] = DP[i-1][c-w]+v*

* c=1 => DP[4][1] = DP[3][1] = 2; 1>=4 – False
* c=2 => DP[4][2] = DP[3][2] = 2; 2>=4 – False
* c=3 => DP[4][3] = DP[3][3] = 4; 3>=4 – False
* c=4 => DP[4][4] = DP[3][4] = 6; 4>=4 – True && DP[3][0] + v > DP[4][4] *(0+5>6)* – False
* c=5 => DP[4][5] = DP[3][5] = 6; 5>=4 – True && DP[3][1] + v > DP[4][5] *(2+5>6)* – True => DP[4][5]=DP[3][1]+v =7
* c=6 => DP[4][6] = DP[3][6] = 6; 6>=4 – True && DP[3][2] + v > DP[4][6] *(2+5>6)* – True => DP[4][6]=DP[3][2]+v =7
* c=7 => DP[4][7] = DP[3][7] = 8; 7>=4 – True && DP[3][3] + v > DP[4][7] *(4+5>6)* – True => DP[4][7]=DP[3][3]+v =9

Same for step i=5; *10 is the best possible value.*

Complete solution:

c – maximum capacity; N – number of items

for (int i = N; i > 0; i--) {

if (DP[i][c] != DP[i - 1][c]) {

itemsSelected.add(i);

c -= W[i];

}

}

c = 7; N = 5

*DP[i][c] != DP[i - 1][c] add item c = c – w[i]*

* DP[5][7] != DP[4][7] (10!=9) – True add item 5 (v=3, w=2) c = 7-2 = 5
* DP[4][5] != DP[3][5] (7!=6) – True add item 4 (v=5, w=4) c = 5-4 = 1
* DP[3][1] != DP[2][1] (2!=2) – False
* DP[2][1] != DP [1][1] (2!=0) – True add item 2 (v=2, w=1) c = 1 – 1 = 0
* DP[1][0] != DP[0][0] (0!=0) – False

**Example2**

*Item = 1 2 3 4*

V[] ={10, 40, 30, 50}

W[] ={5, 4, 6, 3}

max capacity = 10

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i\c | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 Empty | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1(v=10, w=5) | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 | 10 | 10 | 10 |
| 2(v=40, w=4) | 0 | 0 | 0 | 0 | 40 | 40 | 40 | 40 | 40 | 50 | 50 |
| 3(v=30, w=6) | 0 | 0 | 0 | 0 | 40 | 40 | 40 | 40 | 40 | 50 | 70 |
| 4(v=50, w=3) | 0 | 0 | 0 | 50 | 50 | 50 | 50 | 90 | 90 | 90 | 90 |

* 90!=70 so we add item 4 (v=50, w=3); the remaining capacity is 10-3 = 7
* 40==40 so skip adding item 3
* 40!=10 so we add item 2 (v=40, w=4); the remaining capacity is 7-4 = 3
* 0 == 0 so skip adding item 1

Best result = {item2, item4} = 90

**LCS (Longest Common Subsequence)**

**Example1**

A = “GXTXAYB”

B = “AGGTAB”

int n = A.length; m = B.length;

int[][] dp = new int[n + 1][m + 1];

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= m; j++) {

if (A[i - 1] == B[j - 1]) dp[i][j] = dp[i - 1][j - 1] + 1;

else dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);

}

}

*j =* *0 1 2 3 4 5 6*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | “” | A | G | G | T | A | B |
| “” | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| X | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| T | 0 | 0 | 1 | 1 | 2 | 2 | 2 |
| X | 0 | 0 | 1 | 1 | 2 | 2 | 2 |
| A | 0 | 1 | 1 | 1 | 2 | 3 | 3 |
| Y | 0 | 1 | 1 | 1 | 2 | 3 | 3 |
| B | 0 | 1 | 1 | 1 | 2 | 3 | 4 |

*i =*

*0*

*1*

*2*

*3*

*4*

*5*

*6*

*7*

m = 6; n = 7

*0 1 2 3 4 5 6*

*A* G X T X A Y B

*B* A G G T A B

***For i=1***

*IF A[i – 1] == B[j – 1] then dp[i][j] = dp[i – 1][j – 1] + 1; else dp[i][j] = Math.max(dp[i – 1][j], dp[i][j – 1]);*

* j=1 A[0] ==B[0] False (G!=A) dp[1][1] = MAX(dp(0][1],dp[1][0]) = 0
* j=2 A[0] == B[1] True (G==G) dp[1][2] = dp[0][1] + 1 = 1
* j=3 A[0] == B[2] True (G==G) dp[1][3] = dp[0][2] + 1 = 1
* j=4 A[0] == B[3] False (G!=T) dp[1][4] = MAX(dp(0][4],dp[1][3]) = 1
* j=5 A[0] == B[4] False (G!=A) dp[1][5] = MAX(dp(0][5],dp[1][4]) = 1
* j=6 A[0] == B[5] False (G!=B) dp[1][6] = MAX(dp(0][6],dp[1][5]) = 1

***For i=2***

*IF A[i – 1] == B[j – 1] then dp[i][j] = dp[i – 1][j – 1] + 1; else dp[i][j] = Math.max(dp[i – 1][j], dp[i][j – 1]);*

* j=1 A[1] ==B[0] False (X!=A) dp[2][1] = MAX(dp(1][1],dp[2][0]) = 0
* j=2 A[1] == B[1] False (X!=G) dp[2][2] = MAX(dp(1][2],dp[2][1]) = 1
* j=3 A[1] == B[2] False (X!=G) dp[2][3] = MAX(dp(1][3],dp[2][2]) = 1
* j=4 A[1] == B[3] False (X!=T) dp[2][4] = MAX(dp(1][4],dp[2][3]) = 1
* j=5 A[1] == B[4] False (X!=A) dp[2][5] = MAX(dp(1][5],dp[2][4]) = 1
* j=6 A[1] == B[5] False (X!=B) dp[2][6] = MAX(dp(1][6],dp[2][5]) = 1

***For i=3***

*IF A[i – 1] == B[j – 1] then dp[i][j] = dp[i – 1][j – 1] + 1; else dp[i][j] = Math.max(dp[i – 1][j], dp[i][j – 1]);*

*…*

* j=4 A[2] == B[3] True (T==T) dp[3][4] = dp[2][3] + 1 = 2
* j=5 A[2] == B[4] False (T!=A) dp[3][5] = MAX(dp(2][5],dp[3][4]) = 2
* j=6 A[2] == B[6] False (T!=B) dp[3][6] = MAX(dp(2][6],dp[3][5]) = 2

***Same for i = 4,5,6,7***

***Longest common subsequence len = 4***

***Complete solution*** *(finding the characters of the longest common subset)*

int i = n, j = m;

int lcsLen = dp[n][m];

char[] lcs = new char[lcsLen];

int index = 0;

while (i >= 1 && j >= 1) {

int v = dp[i][j];

while (i > 1 && dp[i - 1][j] == v) i--;

while (j > 1 && dp[i][j - 1] == v) j--;

if (v > 0) lcs[lcsLen – index - 1] = A[i - 1]; // or B[j-1];

index++;

i--; j--;

}

// i = 7, j = 6, len = 4

*v = dp[i][j] while (i > 1 && dp[i - 1][j] == v) i--; while (j > 1 && dp[i][j - 1] == v) j--; if (v > 0) lcs[lcsLen – index - 1] = A[i - 1]; index++; i--;j—*

v = dp[7][6] = 4 dp[6][6] != v => i=7 dp[7][5] != v => j = 6 lcs[4-0-1] = A[6] = B index = 1;i=6;j=5

v= dp[6][5] = 3 dp[5][5] == v => i = 5

dp[4][5] != v => i = 5 dp[5][4] !=v => j = 5 lcs[4-1-1] = A[4] = A index = 2; i=4; j=4

v = dp[4][4] =2 dp[3][4] == v => i =3

dp[2][4] != v => i =3 dp[3][3] != v => j = 4 lcs[4-2-1] =A[2] = T index = 3; i=2; j=3

v = dp[2][3] = 1 dp[1][3] == v => i = 1 dp[1][2] == v => j = 2

dp[1][1] != v => j = 2 lcs[4-3-1] = A[0] = G index = 4; i=0;j=1

A = “GXTXAYB”

B = “AGGTAB”

**Subsequence = “GTAB”**