House Weight Adj set

```
maxWt[0] = 1;   maxWt[1] = W[1];
for i in range(2, len(w)):
    maxWt = max{ maxWt[i-1], maxWt[i-2]
                                + W[i] }

return MaxWeight[len(W)];
```

this returns the max weight
to get the subset →

```
final = {}
final.add(MaxWt[len(W)])   // end element
for (i = len(W)-2; i ≥ 0; i-=2) {
    final.add(MaxWt[i+2] - MaxWt[i];
}
```

KNAPSACK PROBLEM

Budget → weight constraint
maximize value of the item

Item	1	2	3	4
weight	2	3	4	5
value	3	4	5	6

Budget → / Item ↓	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

we can buy one of item one

3+4

2

COST

PROBLEM : edit distance

how similar is 'horse' and 'course' ?

h o r s e
c o u r s e

} for a program these words are quite different

if we ignore the column with u when writing horse

c o u r s e
h o r s e

now only two cols don't match out of six

Levenshtein (Edit) distance

given : 2 strings

edit distance - # of operations needed to transform one string to another

operations : add, delete, replace

minimize # of operations

eg: horse and rose

rose → horse

① oose (R) ② orse (R) ③ horse (A)

horse → rose

① rorse (R) ② rose (D)

Edist(A, B, i, j) → # of operations needed to transform A[0:i] → B[0:j]

```
if (A[i] == B[j]) {
    Edist(A,B, i,j) = Edist(A,B, i-1, j-1);
} else {
    Edist(A, B, i, j) = 1 + min {
            Edist(A,B, i, j-1),
            Edist(A,B, i-1, j),
            Edist(A,B, i-1, j-1)
    }
}
```

		1	2	3	4	5	6	7
		"	h	o	r	s	e
1	"	0	1	2	3	4	5
2	r	1	1 (1+0)	2 (1+1)	2	3	4
3	o	2	2	1	2	3	4
4	s	3	3	2	2	2	3
5	e	4	4	3	3	3	2

The first col and row of cost array should be the base case

Running time: $m \times n$

(size of cost matrix)

↑ replace

← delete

↖ nothing

optimization: for taking space less than $(m \times n)$ we can just keep track of current row and the row above

if we also want the steps taken, we can have an arr of max $(m+n)$ size to do book keeping

KT → Chp 6.6

Jeff → Chp 3.7