https://cses.fi/problemset/task/1633
5 → 15

5
4+1
3+2
2+3
1+4
3+1+1
2+2+1
2+1+2
1+3+1
1+2+2
1+1+3
2+1+1+1
1+2+1+1
1+1+2+1
1+1+1+2

```
int numberOfWays(int n) {
  return numberOfWays(n-1) + numberOfWays(n-2) + numberOfWays(n-3)
 + numberOfWays(n-4) + numberOfWays(n-5) + numberOfWays(n-6)
}
```

	0	1	2	3	4	5	6	7	8	9
ways	1									

```
ways[0] = 1;
for (int i = 1; i <= n; i ++) {
  ways[i] = 0;
  for (int j = 1; j <= 6; j ++) {
    if (i - j > 0) {
      ways[i] += ways[i - j];
    }
  }
}
```

```
return ways[n];
```

ways[X] - по колко начина може да се получи сума X
Отговорът на задачата ще е → ways[n]

https://leetcode.com/problems/palindromic-substrings/

alabala → 12
a
l
a
b
a
l
a
al
la
ab
ba
al
la
ala
lab
aba
bal
ala
alab
laba
abal
bala
alaba
labal
abala
alabal
labala
alabala

```
// имаме някакъв глобален низ S

bool isPalindromeSlow(int left, int right) {
  for (int i = 0; i < (right - left + 1); i ++) {
    if (S[left + i] != S[right - i]) {
        return false;
    }
  }
  return true;
}

bool isPalindrome(int left, int right) {
  if (left == right) {
    return true;
  }
  if (left + 1 == right) {
    return S[left] == S[right];
  }

  if (alreadyChecked[left][right]) {
    return answer[left][right]
  }

  if (S[left] == S[right] && isPalindrome(left + 1, right - 1)) {
    alreadyChecked[left][right] = true;
    answer[left][right] = true;
    return true;
  }

  alreadyChecked[left][right] = true;
  answer[left][right] = false;
  return false;
}

n = S.size();
int count = 0;
for (int i = 0; i < n; i ++) {
```

```
  for (int j = i; j < n; j ++) {
    if (isPalindrome(i, j)) {
      count ++;
    }
  }
}
return count;


// итеративен вариант
isPalindrome[l][r] -> true ако S[l...r] е палиндром и false иначе

for (int i = 1; i <= n; i ++) {
  isPalindrome[i][i] = true;
}
for (int i = 1; i < n - 1; i ++) {
  isPalindrome[i][i + 1] = S[i] == S[i+1];
}

for (int size = 3; size <= n; size ++) {
  int end = start + size - 1;
  for (int start = 0; start + size - 1 < n; start ++) {
    int end = start + size - 1;
    isPalindrome[start][end] =
        S[start] == S[end] && isPalindrome[start + 1][end - 1];
  }
}

int count = 0;
for (int i = 0; i < n; i ++) {
  for (int j = i; j < n; j ++) {
    if (isPalindrome[i, j]) {
      count ++;
    }
  }
}
return count;
```

https://cses.fi/problemset/task/1145

11
10 3 2 5 4 9 8 6 7 1 11 0

2 4 6 7 11 → 5

longest inceasing subsequence

lis[k] → колко дълга е най-дългата растяща подредица, която завършва в позиция position k
Отговорът ни би бил → max(lis[i], за i от 1 до n)

11
10 3 2 5 4 9 8 6 7 1 11 0

id	0	1	2	3	4	5	6	7	8	9	10	11
array	10	3	2	5	4	9	8	6	7	1	11	0
lis	1	1	1	2	2	3	3	3	4	1	5	1
prev	-1	-1	-1	1	1	3	3	3	7	-1	8	-1

```
for (int i = 0; i < n; i ++) {
  lis[i] = 1;
  prev[i] = -1;
  for (int j = 0; j < i; j ++) {
    if (a[j] < a[i] && lis[j] + 1 > lis[i]) {
      lis[i] = lis[j] + 1;
      prev[i] = j;
    }
  }
}
int answer = lis[0], lastPosition;
for (int i = 1; i < n; i ++) {
  if (lis[i] > answer) {
    answer = lis[i];
    lastPosition = i;
```

```
  }
}

vector<int> seq;
while(lastPosition != -1) {
  seq.insert(a[lastPosition]);
  lastPosition = prev[lastPosition];
}

return answer;
```

друга идея
lis[len] - колко е най-малкият елемент в растяща редица с дължина len

https://cses.fi/problemset/task/1639

S - стартовия
T - таргетирания

minOps[len1][len2] → минималния брой операции нужни
да превърнем S[1...len1] във T[1...len2]
Отговорът ще е → minOps[S.size()][T.size()]

LOVE
MOVIE

minOps[2][4] → колко е минимания брой операции да превърнем LO → MOVI
minOps[2][4] = LO→MO→MOV→MOVI = 3

S: ABDE
T: ABE
minOps[3][2] → minOps[2][2] + премахване на знак
ABD→AB

S: ABE
T: ABDE

minOps[2][3] → minOps[2][2] + добявне на знак
AB→ABD

S: ABC
T: ABD
minOps[3][3] → minOps[2][2] + сменяне на последния знак
ABC→ABD

minOps[i][j] = min(minOps[i-1][j] + 1, премахване на i-тия
minOps[i][j-1] + 1, добяване на знак
ако S[i] = T[j], то minOps[i-1][j-1], последните знаци съвпадат
ако S[i] != T[j], то minOps[i-1][j-1] + 1, сменяне на последния знак)