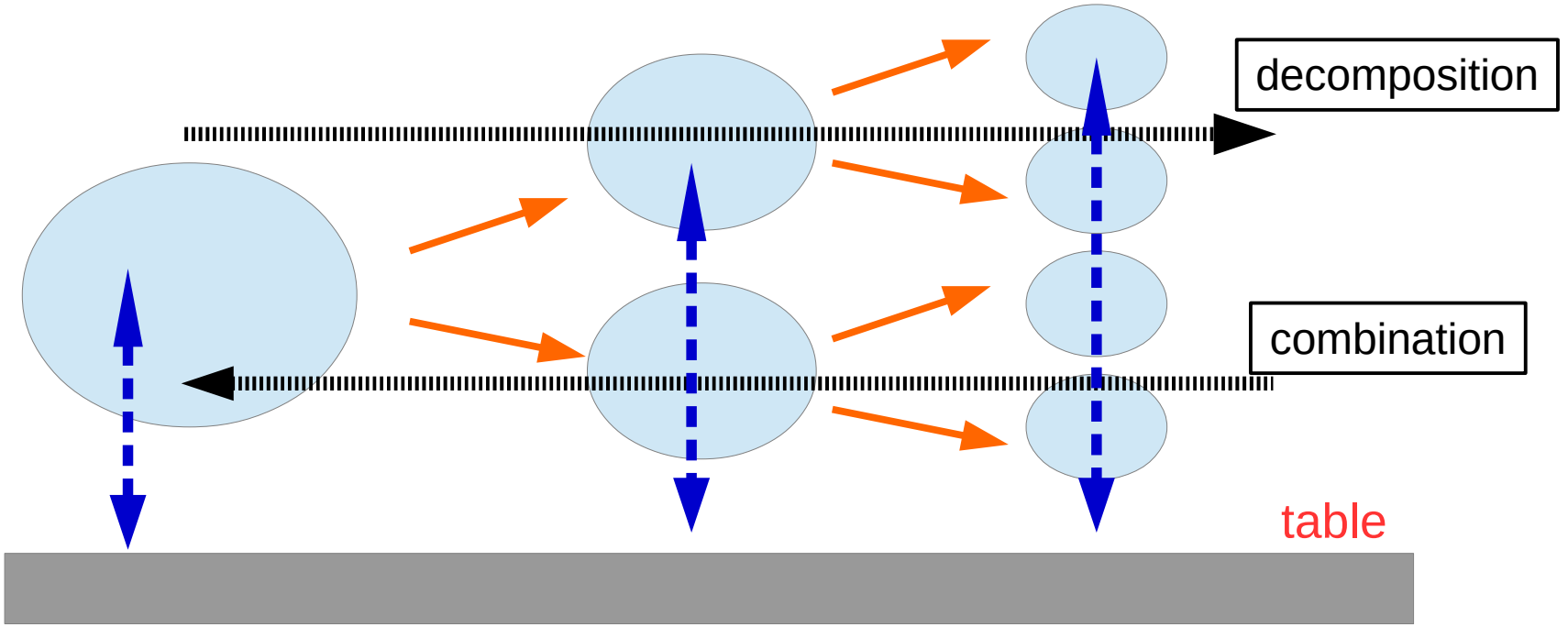
Dynamic Programming

a method for solving a recursive problem in iterative way with a table

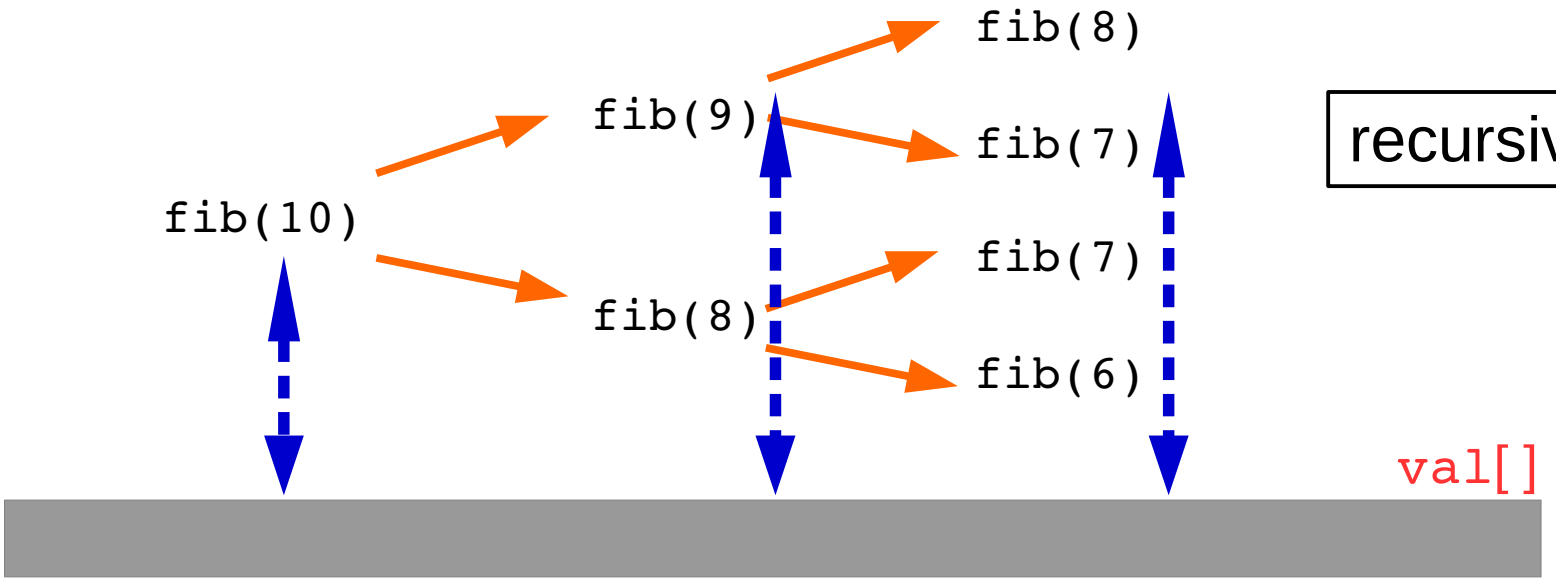
a recursive procedure with a table (**memoization**)
a iterative procedure with a table



```
int got[N_FIB];
int val[N_FIB];

int fib(int n)
{
    if(got[n]) return val[n];
    if      (n==0) r = 0;
    else if(n==1) r = 1;
    else          r = fib(n-1)+fib(n-2)
    val[n] = r;
    got[n] = 1;
    return r;
}
```

```
fib(0) = 0
fib(1) = 1
fib(n) = fib(n-1) + fib(n-2),   n>=2
```

```
int fib(int n)
{
    if(n==0) return 0;
    if(n==1) return 1;
    return fib(n-1)+fib(n-2)
}
```



recursive

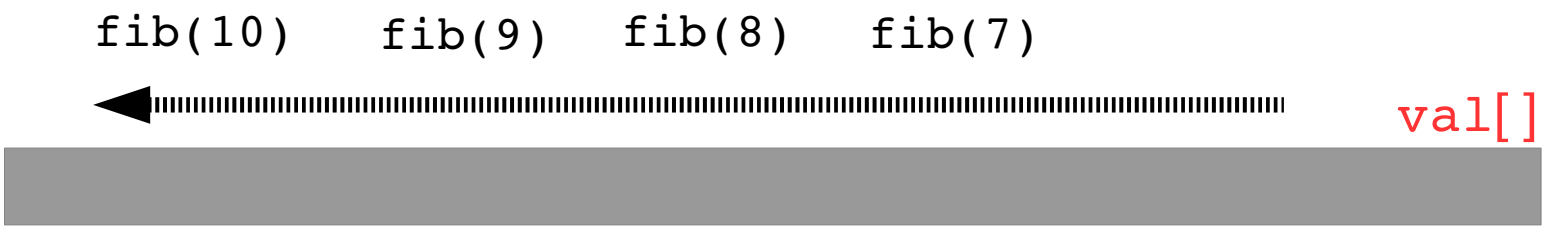
```
int val[N_FIB];
int fib(int n)
{
  val[0]=0;  val[1]=1;
  for(i=2; i<=n; i++)
    val[i] = val[i-1] + val[i-2];
  return val[n];
}
```

```
fib(0) = 0
fib(1) = 1
fib(n) = fib(n-1) + fib(n-2),  n>=2
```

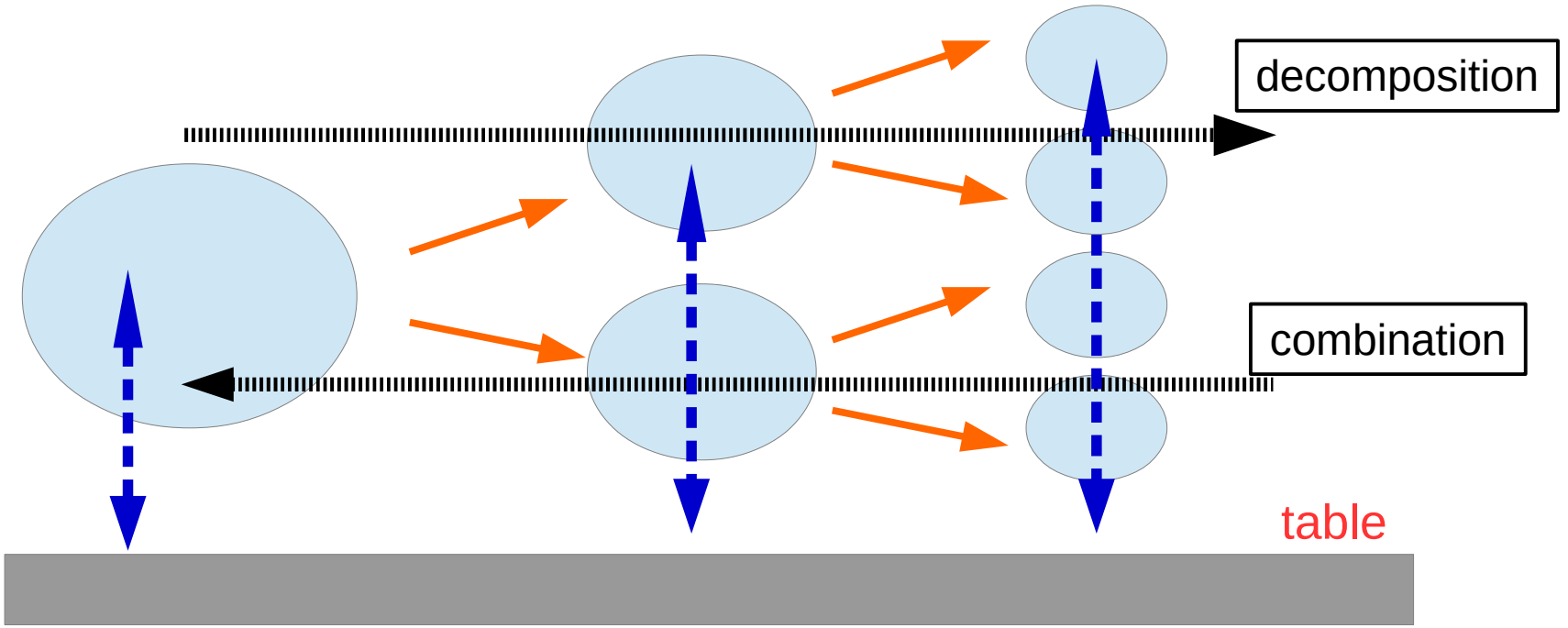
```
int fib(int n)
{
  if(n==0) return 0;
  a=0; b=1;
  for(i=2; i<=n; i++){
    t = a + b; a = b; b = t;
  }
  return b;
}
```

```
int val[N_FIB]= {0, 1};
int lastn = 1;
int fib(int n)
{
  for(i=lastn+1; i<=n; i++)
    val[i] = val[i-1] + val[i-2];
  lastn = max(lastn, n);
  return val[n];
}
```

iterative



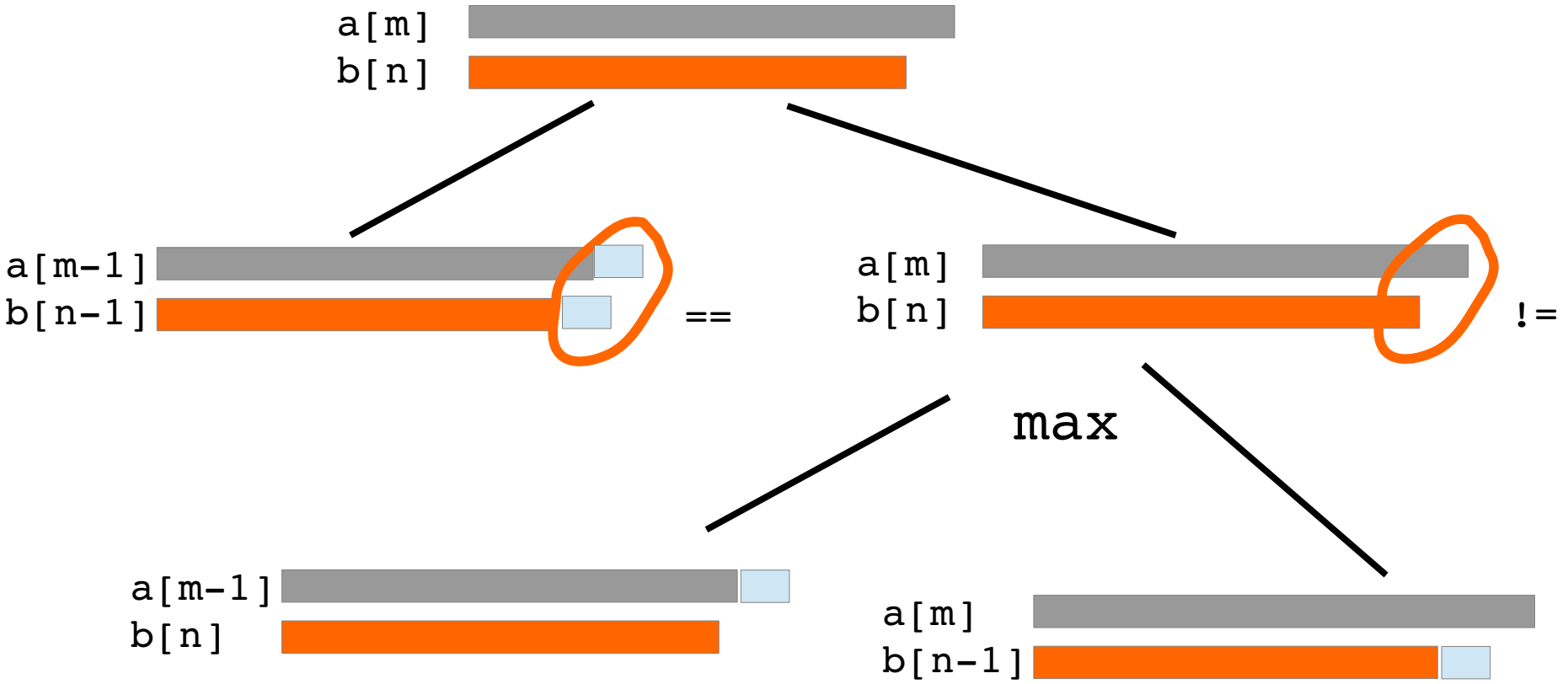
recursive: decomposition, combination
How?



LCS, uva-111
Longest Common subsequence

```
a[] = 1 4 3 2 1 8 3
b[] = 5 4 9 1 3

common subsequence:
a[] = 1 4 3 2 1 8 3  ==> 4
b[] = 5 4 9 1 3

a[] = 1 4 3 2 1 8 3  ==> 1 3
b[] = 5 4 9 1 3

a[] = 1 4 3 2 1 8 3  ==> 4 1 3
b[] = 5 4 9 1 3
```

LCS
Longest Common subsequence



LCS
Longest Common subsequence

```
int a[N];
int b[N};
lcs(int m, int n)
{ /* m==0 or n==0 ? */
  if(a[m] == b[n])
    return lcs(m-1, n-1)+1;
  else
    return max(lcs(m, n-1), lcs(m-1, n);
}
```

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

```
int ans[N][N];
lcs(int m, int n)
{ /* m==0 or n==0 ? */
  for(i=1; i<=m; i++)
    for(j=1; j<=n; j++)
      if(a[i] == a[j])
         ans[i][j] = a[i-1][j-1] + 1;
       else
         ans[i][j] = max(a[i-1][j], a[i][j-1]);
}
```

LIS, uva-10131
Longest Increasing subsequence

```
a[] = 1 4 3 2 1 8 3

increasing subsequence:

a[] = 1 4 3 2 1 8 3   ==> 4

a[] = 1 4 3 2 1 8 3   ==> 1 3

a[] = 1 4 3 2 1 8 3   ==> 1 4 8

a[] = 1 4 3 2 1 8 3   ==> 1 3 8
```

```
a[] = 1 4 3 2 1 8 3

LIS ended in 3:  1 2 3
LIS ended in 8:  1 2 8
LIS ended in 1:  1
LIS ended in 2:  1 2
LIS ended in 3:  1 3
LIS ended in 4:  1 4
LIS ended in 1:  1
```

LIS of a sequence
--->
LIS ended in k-th position of a sequence, k=1,2,3...

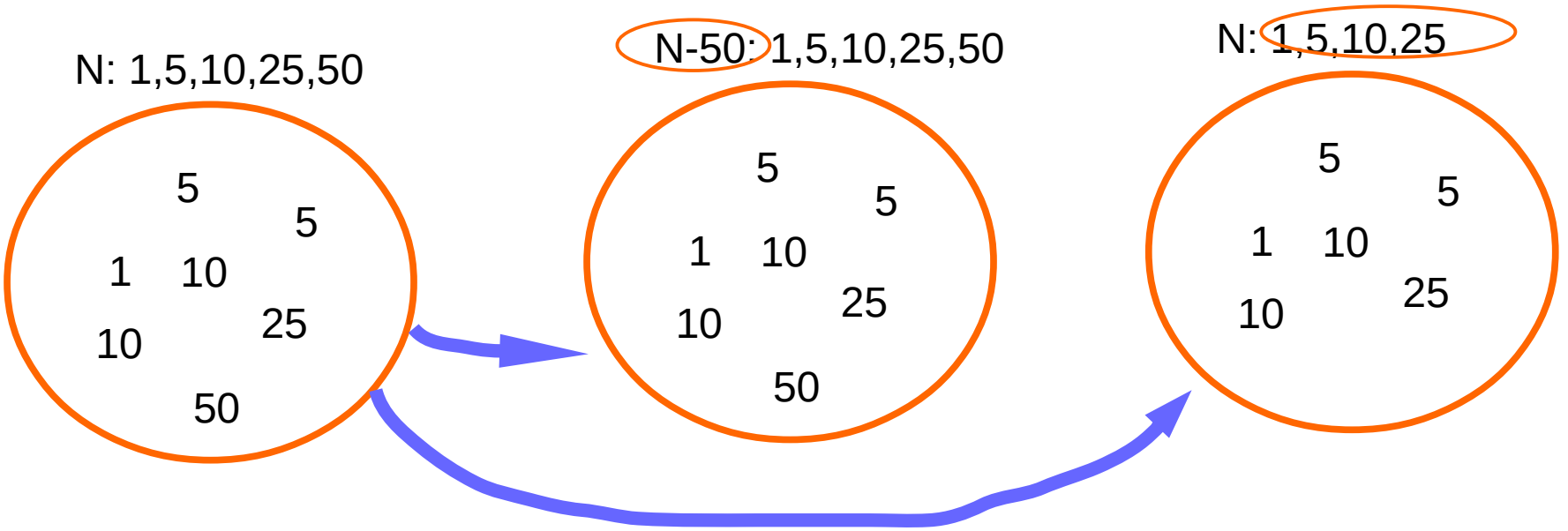
LIS
Longest Increasing subsequence

lis(k) = LIS ended in k-th position of a sequence
lis(k) = max(lis(j)+1), j=1..k-1, a[j]<a[k]

```
int a[N];
int lis[N};

ans = 0;
for(k=1; k<=N; i++){
  lis[k] = 1;
  for(j=1; j<k; j++)
    if(a[j] < a[k] && lis[j]+1 > lis[i])
      lis[i] = lis[j] + 1;
  ans = max(ans, lis[i]);
}
```

Coin Change, uva-674

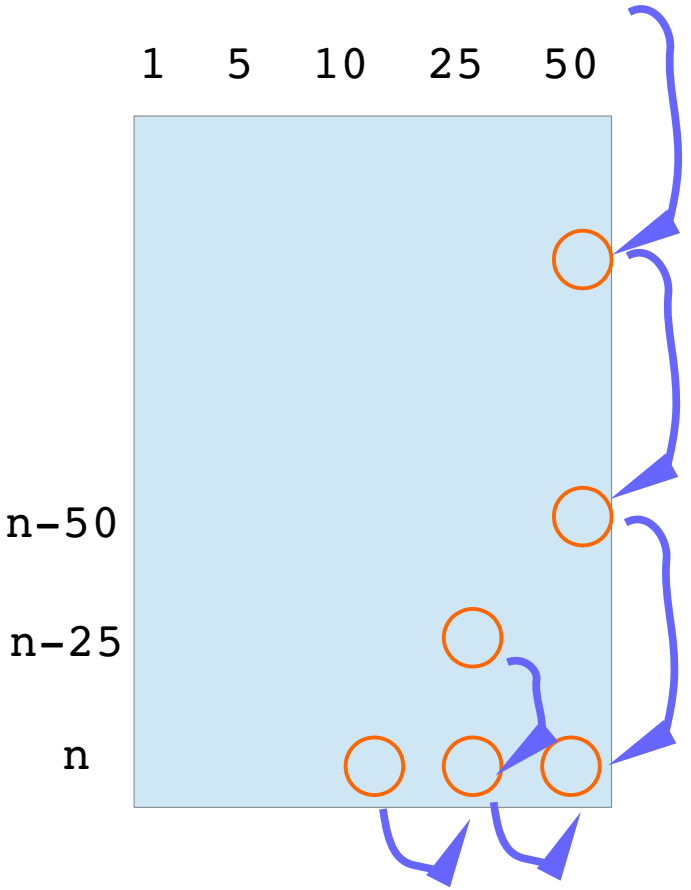
```
five kinds of coins: 1, 5, 10, 25, 50

How many ways can N be composed of
such coins?
--->
How many solutions are there in
a+ 5*b + 10*c + 25*d + 50*e = N,
 a,b,c,d,e >=0, integer
```



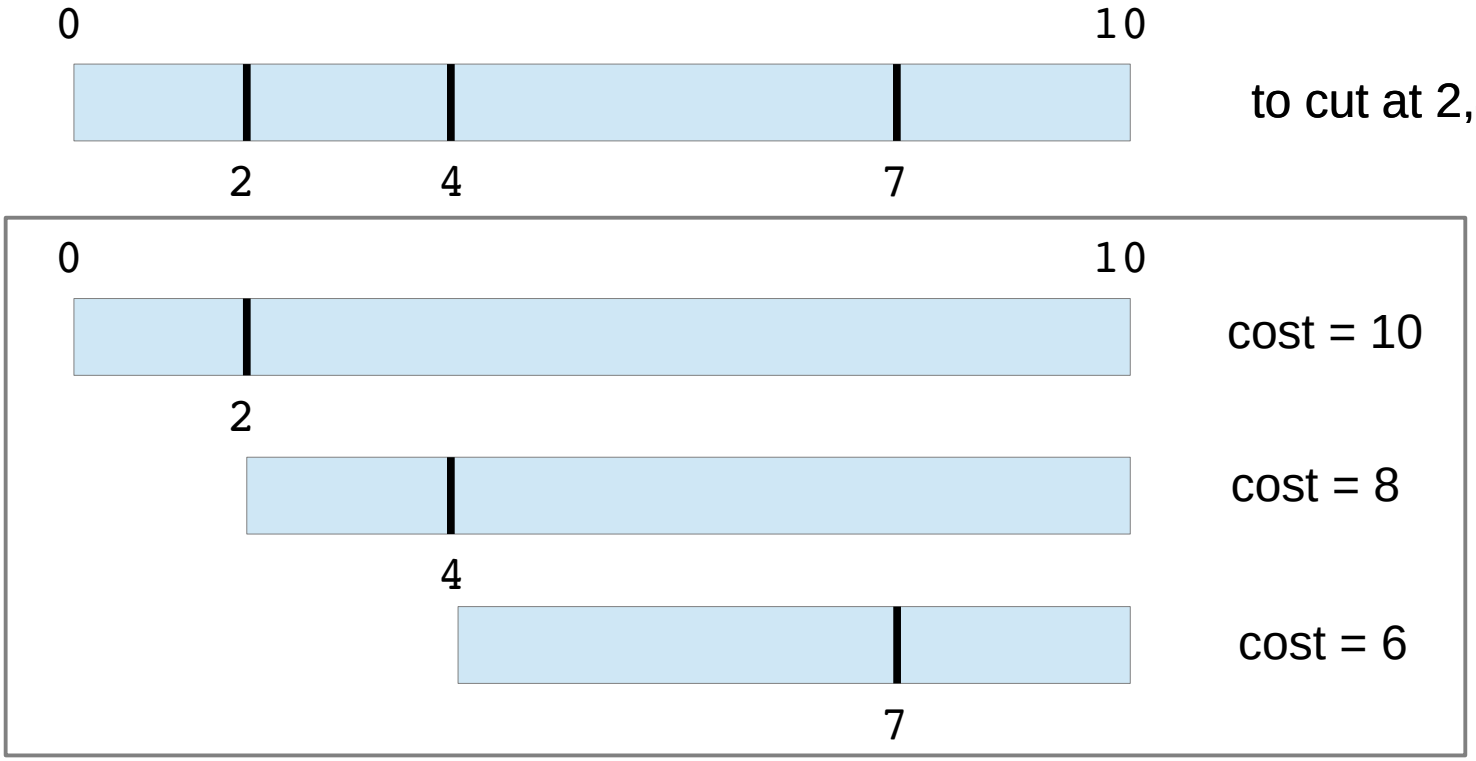
Coin Change, uva-674

```
int val[6] = {0,1,5,10,25,50};

int cc(int n, int k)
{
  if(n==0) return 1;
  if(n<0 || k==0) return 0
  return cc(n-val[k], k) + cc(n, k-1);
}
```

```
int val[6] = {0,1,5,10,25,50};
int ans[N][K];

for(i=0; i<=n; i++)  /* i<0 ?  */
  for(j=1; j<=K; j++)/* j<=0 ? */
    ans[i][j] = a[i        ][j-1]
              + a[i-val[j]][j];
```



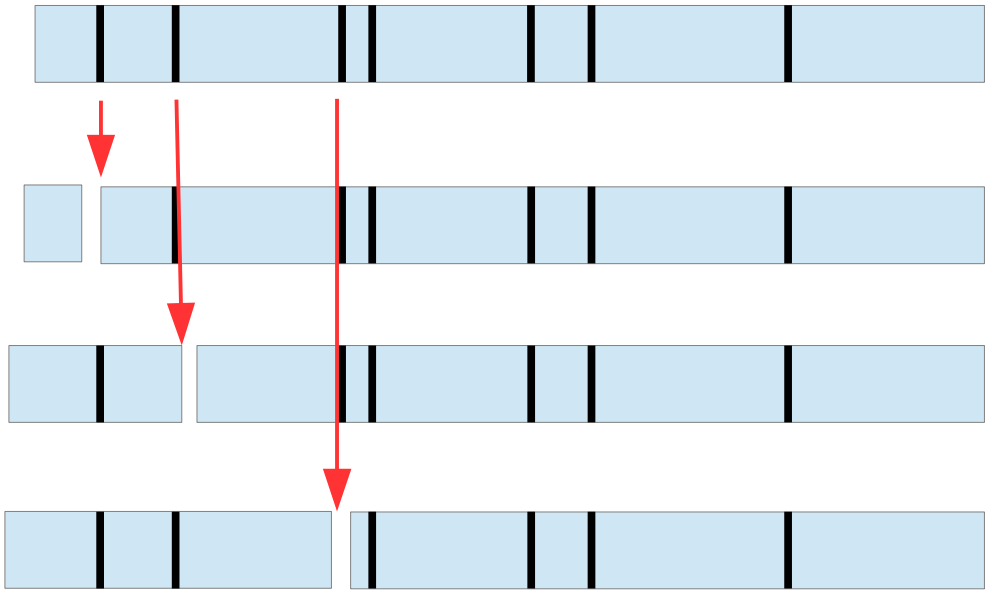
Cutting Sticks, uva-10003



cost of cut in the order 2,4,7 ==> 24
cost of cut in the order 4,2,7 ==> 10 + 4 + 6 = 20
cost of cut in the order 7,2,4 ==> 10 + 7 + 5 = 22

Cutting Sticks, uva-10003

```
int saved[MAX_N_CUT][MAX_N_CUT];
int value[MAX_N_CUT][MAX_N_CUT];
int cuts [MAX_N_CUT];

int try(int s, int t)
{ int k,cost,ans;

  if(s > t) return 0;
  if(saved[s][t]) return value[s][t];
  if(s == t)
    ans =  0;
  else {
    ans = 8000000;
    for(k=s; k<=t; k++) {
      cost = try(s,k-1) + try(k+1,t);
      if(ans > cost) ans = cost;
    }
  }
  ans += cuts[t+1] - cuts[s-1];
  saved[s][t] = 1;  value[s][t] = ans;
  return ans;
}

int main(int argc, char *argv[])
{ try(1,ncuts));}
```



Cutting Sticks, uva-10003

```
int value[MAX_N_CUT][MAX_N_CUT];
int cuts [MAX_N_CUT];

int try(int ncuts)
{ int i,k,len,cost,min;

  for(i=0; i<=ncuts;   i++) value[i][1] = 0;
  for(i=0; i<=ncuts-1; i++) value[i][2] = cuts[i+2]-cuts[i];
  for(len=3; len<=ncuts+1; len++) {
    for(i=0; i<=ncuts+1-len; i++) {
      min = 8000000;
      for(k=1; k<=len-1; k++) {
        cost = value[i][k] + value[i+k][len-k];
        if(min > cost) min = cost;
      }
      value[i][len] = min + cuts[i+len] - cuts[i];
    }
  }
  return value[0][ncuts+1];
}
```