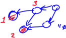
Dynamic Programming



"Функцию от вершины"

1) F_n.

$f_0 = 0$

$f_1 = 1$

$f_n = f_{n-1} + f_{n-2}$

①
```
def f(n):
    if n<=1:
        ret n
    ret f(n-1)+f(n-2)
```
✗ $\Theta(F_n)$

②
```
f = [0.... ]
f[0]=0
f[1]=1
for k=2..n
    f[k] = f[k-1]+f[k-2]
ret f[n]
```
✓ $\Theta(n)$

2) $\binom{n}{k} = C(n,k)$

$C(n,k) = C(n-1, k-1) + C(n-1, k)$

$C(0,0) = 1$

$C(n, >n) = 0$

```
C[0][0] = 1
for n = 1..N
    for k = 0..n:
        C[n][k] = C[n-1][k-1] +
                + C[n-1][k].
```

$O(n^2)$

3)

$x \to \begin{cases} x+3 \\ x+4 \\ x+5 \end{cases}$

$1 \rightsquigarrow N$

dp_i = min число прыжков $1 \to i$

$dp_i = \begin{cases} +\infty, & i\text{-дырка} \\ \min(dp[i-3],\ dp[i-4],\ dp[i-5]) + 1 \end{cases}$

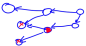
dp[i-2]

d

```
dp = [ - - - - - ]
dp[1] = 0
for k = 2 .. N:
    dp[k] = +∞
    if not hole[k]
        for t = 3 .. 5
            if k >= t:
                dp[k] = min(dp[k], dp[k-t]+1)
print dp[N]
```

O(N)

dp[k-4] → k (4, 3)



Динамика назад. f[N]

$dp = [- - - - -]$

$dp[1] = 0$

dp_i = число прыжков $1 \to i$

«Вперёд»

```
for k = 1 .. N:
    if not hole[k]:
        for t = 3 .. 5
            if k+t ≤ N and
               not hole[k+t]:
                dp[k+t] = min(dp[k+t],
                              dp[k] + 1)
```

$O(n)$

Динамика ленивая (Lazy)

$dp_n : 1 \to n$

dp = [-1, -1, ..., -1]

```
def f(n):   <- if n==1: return 0.
    if dp[n] ≠ -1:
        return dp[n]        Релаксация
    dp[n] = +∞
    if not hole[n]:
        for t = 3..5
            dp[n] = min(dp[n], f(n-t)+1)
    return dp[n]
```

$O(n)$

Назад
Вперёд
Lazy

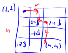
1) Не пытаемся угадать порядок вычисления

2) Чуть больше

3) Иногда удобнее $O(1)$

4) Можно посетить меньше состояний

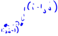
Пример



dp[i][j] = max $

(1,1) → (i,j)

Print (dp[n][m])



dp[1][1] = a[1][1]

dp[i][j] ← a[i][j] + dp[i][j-1]

← a[i][j] + dp[i-1][j]

for i = 1..n

for j = 1..m

dp[i][j] = MAX(...)

par[i][j] = [Left / Up

$cnt[i][j] = cnt[i-1][j] + cnt[i][j-1]$,

если (i,j) не дыры.

```
for i = n..1
  for j = m..1
    (i,j) → (i+1,j)
          → (i,j+1)
```

Рюкзак

$w_1, \dots, w_N \in \mathbb{N}$

dp[N][S].

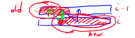
∃ ли подмн-во веса S

dp[i][s] = 1/0, можно ли набрать вес s если доступны предм. $w_1 \dots w_i$

```
dp[0][0]=1
for i=1..N:
  for p=0..S:
    dp[i][p] = dp[i-1][p]
    if p >= w_i
      dp[i][p] |= dp[i-1][p-w_i]
```

$O(nS)$

$O(S)$



d 1 = 6
a = a or 6

100100
S

dp = [- - - -]
dp[0] = 1

for i = 1 .. n:
    for p = S .. w:
        dp[p] |= dp[p - w]
    dp |= (dp << w)

100...0

17 = 10001

17 << 2 = 1000100 = 68

w = 2
w = 3
1 0 1 0 0 0 0 0 0
1 1 1

1 0 0 0 0 0 0 0 0
1 1

```
dp = [1 0 0 - - - - 0 ]
dp[0] = 1
for i = 1 .. n:
    for p = S .. w:
        dp[p] |= dp[p-w_i]
        if dp[p-w_i] and not dp[p]:
            dp[p] = 1
            par[p] = w_i
```

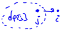
par[i][w]

НВП LIS

4 3 2 5 7

dp_i = длина НВП заканчив в i

```
dp = [0 .. ]
for i=1..n
  dp[i]=1   {i}
  for j=1..i-1:
    if a[j] < a[i]:
      dp[i] max= dp[j]+1
```



$O(n^2)$

$O(n \log n)$

HDN (LCS)

$S = ABACD$

$t = ACADE$

$lcs(S, t)$

$O(nm)$

$O(n^2)$

$O\left(\frac{n^2}{\log^2 n}\right)$

$O(n^{2-\epsilon})$

$lcs[i][j] = lcs(S_1 .. S_i,\ t_1 .. t_j)$

$lcs[i][j] \leftarrow lcs[i-1][j]$

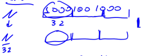
$\quad \leftarrow lcs[i][j-1]$

$S_i = t_j$: $lcs[i][j] \leftarrow 1 + lcs[i-1][j-1]$.

Résumé :

uint32_t   x|y x&y, x<<10
O(1) O(1)   O(1)

bitset<N>   vector<uint32_t>



<< ⌈N/32⌉

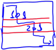
⌈N/w⌉

for i=1..n:
dp1 = (dp << wi)

$O\left(\frac{nS}{w}\right)$

w=32

$O(ns)$ → $O(s)$



$dp[i][j] \leftarrow$

$\leftarrow a[i][j] +$
$+ dp[i-1][j]$

$dp[i][j] \leftarrow dp[i][j-1]$
$+ a[i][j].$

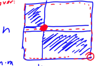
$O(n^2)$
↓ memory
$O(n)$

10^9 ops/sec

$n = 30K$ $O(n^2)$ time

10^9 int

recover:



Хиршберг

$(1,1) \to (n,m)$

$(1,1) \to (\frac{n}{2}, i)$

$(\frac{n}{2}, i) \to (n,m)$

$\frac{n \cdot m}{2}$

$\frac{n \cdot m}{4}$

recover:

dp[i][j]: <int, int>

+a[i] <value, mark>

dp[i+1][j]: <value', mark'>

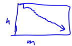
$i = n/2$

dp[i][j].mark = j

dp[n][m].mark

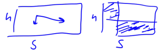
$$T_{recover}(n,m) = n \cdot m + T_{rec}(\frac{n}{2}, i) + T_{rec}(\frac{n}{2}, m-i) = \Theta(nm)$$

1)



$O(nm)$ time
$O(m)$ space

2)



i-1
i

```
for i=1..h
  for j=1...m
     dp[i][j] = ....
  if i=h/2:
    for j=1..m:  dp[i][j].MARK=j
```