

Quick	Sum	$1 + 4 + 9 = 14$
1	2	3
3	1	4

Q_s 3 4 8 17 27 29

~~$O(n)$~~ Per Query
 \downarrow
 $O(1)$ Per Query

$$Q_s(i) = Q_s(i-1) + arr(i)$$

$$Q_s(0) = 0$$

$$\text{sum}(a, b) = Q_s(b) - Q_s(a-1)$$

$$\begin{aligned} \text{sum}(2, 4) &= Q_s(4) - Q_s(1) \\ &= 17 - 3 = 14 \end{aligned}$$

```
void preprocess() {
    for (i = 1 to N) {
         $Q_s(i) = Q_s(i-1) + arr(i)$ 
    }
}
```

```
int sum(int a, int b) {
    return Q_s(b) - Q_s(a-1)
}
```

$O(n + Q)$ \downarrow \rightarrow จัดการ (รวมทั้ง $O(1)$)
 \downarrow preprocess

DP

แบบ

จะ optimize : expo \rightarrow poly

1. Overlapping Subproblem

2. Optimal Substructure

Classical \rightarrow Pattern

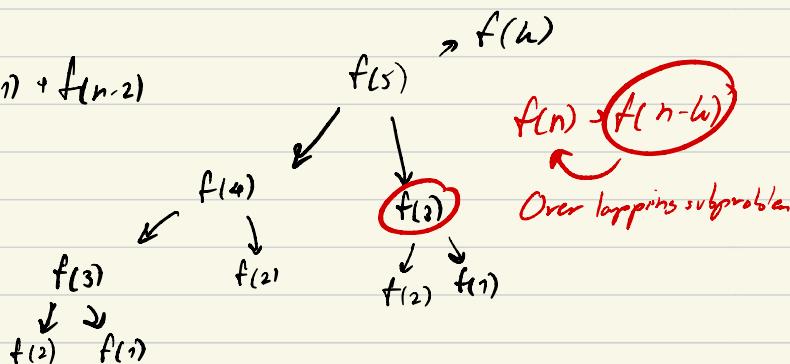
Non-Classical

Fibonacci

$$f(n) = f(n-1) + f(n-2)$$

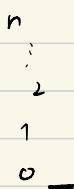
$$f(1) = 1$$

$$f(2) = 1$$



$f(n) \rightarrow f(n-w)$
Overlapping subproblem

Stair



ก้าวที่ 1 ชั้นที่ 1
ก้าวที่ 2 ชั้นที่ 2
ก้าวที่ 3 ชั้นที่ 3
ก้าวที่ 4 ชั้นที่ 4
ก้าวที่ 5 ชั้นที่ 5
ก้าวที่ 6 ชั้นที่ 6
ก้าวที่ 7 ชั้นที่ 7
ก้าวที่ 8 ชั้นที่ 8
ก้าวที่ 9 ชั้นที่ 9
ก้าวที่ 10 ชั้นที่ 10
ก้าวที่ 11 ชั้นที่ 11
ก้าวที่ 12 ชั้นที่ 12
ก้าวที่ 13 ชั้นที่ 13
ก้าวที่ 14 ชั้นที่ 14
ก้าวที่ 15 ชั้นที่ 15
ก้าวที่ 16 ชั้นที่ 16
ก้าวที่ 17 ชั้นที่ 17
ก้าวที่ 18 ชั้นที่ 18
ก้าวที่ 19 ชั้นที่ 19
ก้าวที่ 20 ชั้นที่ 20
ก้าวที่ 21 ชั้นที่ 21
ก้าวที่ 22 ชั้นที่ 22
ก้าวที่ 23 ชั้นที่ 23
ก้าวที่ 24 ชั้นที่ 24
ก้าวที่ 25 ชั้นที่ 25
ก้าวที่ 26 ชั้นที่ 26
ก้าวที่ 27 ชั้นที่ 27
ก้าวที่ 28 ชั้นที่ 28
ก้าวที่ 29 ชั้นที่ 29
ก้าวที่ 30 ชั้นที่ 30
ก้าวที่ 31 ชั้นที่ 31
ก้าวที่ 32 ชั้นที่ 32
ก้าวที่ 33 ชั้นที่ 33
ก้าวที่ 34 ชั้นที่ 34
ก้าวที่ 35 ชั้นที่ 35
ก้าวที่ 36 ชั้นที่ 36
ก้าวที่ 37 ชั้นที่ 37
ก้าวที่ 38 ชั้นที่ 38
ก้าวที่ 39 ชั้นที่ 39
ก้าวที่ 40 ชั้นที่ 40
ก้าวที่ 41 ชั้นที่ 41
ก้าวที่ 42 ชั้นที่ 42
ก้าวที่ 43 ชั้นที่ 43
ก้าวที่ 44 ชั้นที่ 44
ก้าวที่ 45 ชั้นที่ 45
ก้าวที่ 46 ชั้นที่ 46
ก้าวที่ 47 ชั้นที่ 47
ก้าวที่ 48 ชั้นที่ 48
ก้าวที่ 49 ชั้นที่ 49
ก้าวที่ 50 ชั้นที่ 50
ก้าวที่ 51 ชั้นที่ 51
ก้าวที่ 52 ชั้นที่ 52
ก้าวที่ 53 ชั้นที่ 53
ก้าวที่ 54 ชั้นที่ 54
ก้าวที่ 55 ชั้นที่ 55
ก้าวที่ 56 ชั้นที่ 56
ก้าวที่ 57 ชั้นที่ 57
ก้าวที่ 58 ชั้นที่ 58
ก้าวที่ 59 ชั้นที่ 59
ก้าวที่ 60 ชั้นที่ 60
ก้าวที่ 61 ชั้นที่ 61
ก้าวที่ 62 ชั้นที่ 62
ก้าวที่ 63 ชั้นที่ 63
ก้าวที่ 64 ชั้นที่ 64
ก้าวที่ 65 ชั้นที่ 65
ก้าวที่ 66 ชั้นที่ 66
ก้าวที่ 67 ชั้นที่ 67
ก้าวที่ 68 ชั้นที่ 68
ก้าวที่ 69 ชั้นที่ 69
ก้าวที่ 70 ชั้นที่ 70
ก้าวที่ 71 ชั้นที่ 71
ก้าวที่ 72 ชั้นที่ 72
ก้าวที่ 73 ชั้นที่ 73
ก้าวที่ 74 ชั้นที่ 74
ก้าวที่ 75 ชั้นที่ 75
ก้าวที่ 76 ชั้นที่ 76
ก้าวที่ 77 ชั้นที่ 77
ก้าวที่ 78 ชั้นที่ 78
ก้าวที่ 79 ชั้นที่ 79
ก้าวที่ 80 ชั้นที่ 80
ก้าวที่ 81 ชั้นที่ 81
ก้าวที่ 82 ชั้นที่ 82
ก้าวที่ 83 ชั้นที่ 83
ก้าวที่ 84 ชั้นที่ 84
ก้าวที่ 85 ชั้นที่ 85
ก้าวที่ 86 ชั้นที่ 86
ก้าวที่ 87 ชั้นที่ 87
ก้าวที่ 88 ชั้นที่ 88
ก้าวที่ 89 ชั้นที่ 89
ก้าวที่ 90 ชั้นที่ 90
ก้าวที่ 91 ชั้นที่ 91
ก้าวที่ 92 ชั้นที่ 92
ก้าวที่ 93 ชั้นที่ 93
ก้าวที่ 94 ชั้นที่ 94
ก้าวที่ 95 ชั้นที่ 95
ก้าวที่ 96 ชั้นที่ 96
ก้าวที่ 97 ชั้นที่ 97
ก้าวที่ 98 ชั้นที่ 98
ก้าวที่ 99 ชั้นที่ 99
ก้าวที่ 100 ชั้นที่ 100

$n=1 ; 1$
 $n=2 ; 2 / 1,1 2$
 $n=3 ; 3 1,1,1 2,1 1,2$

1. จัดเรียง ลำดับ

2. คิด recurrence

3. คิด base case

$dp(i) :=$ คำนวณวิธีการวางบันไดที่ i ;

$dp(i) = dp(i-1) + dp(i-2)$

$dp(i) = 0 ; i < 0$

$dp(i) = 1 ; i = 0$

$$1 + 0 = 1$$

$$dp(1) = dp(0) + dp(1-1)$$

$$dp(2) = dp(1) + dp(0) = 2$$

$$dp(3) = dp(2) + dp(1) = 3$$

$$\downarrow$$

$$dp(2) \xrightarrow{3} \xrightarrow{2} dp(1)$$

Coin Combination I

জুড়ে নীচে কুমুদী মুদ্রার সমূহ, 1, 2, 4 টাঙ্কা টাঙ্কা

$dp(3) \rightarrow 1, 1, 1$
 $1, 2$
 $2, 1$

$$dp(i) = 1; i \geq 0$$

$$dp(i) = 0; i \leq 0$$

$dp(4) \rightarrow 1, 1, 1, 1$
 $1, 1, 2$
 $1, 2, 1$
 $2, 1, 1$
 $2, 2$
 4

$$dp(i) = dp(i-1) + dp(i-2) + dp(i-4) \rightarrow O(1) + O(1) + O(1) = O(1)$$

$dp(i) :=$ কুমুদী মুদ্রার সমূহ কুমুদী মুদ্রার সমূহ, 1, 2, 4

$$O(n) \times n = O(n^2)$$

Push / Pull

```
dp(0)=1
for (i=0 to N) {
    dp(i+1) += dp(i)
    dp(i+2) += dp(i)
    dp(i+4) += dp(i)
}
```

i	0	1	2	3	4	5
dp(i)	1	1	2	3	6	1

?

Iterative / Recursive

Recur (Top down) \rightarrow Pros: Easy to use
 Cons: 1. Slow
 2. Use a lot of mem

```
int solve ( int n ) {
    if (n < 0) return 0
    if (n == 0) return 1
    if (vi[n]) return dp[n]
    if (dp[n] != 0) return dp[n]
    vi[n] = true
}
```

return $dp[n] = solve(n-1) + solve(n-2) + solve(n-4)$

```
int N=10
solve(N)
```

$O(3^n) \rightarrow O(n)$
 expo poly

Iterative (Bottom-up)

Pros: Faster

Cons: Hard

Easy to manage

```

dp(0)=1
for(j=1 to N){
    dp(j) = dp(j-1)
    if(i-2 ≥ 0) dp(j) += dp(j-2)
    if(i-4 ≥ 0) dp(j) += dp(j-4)
}
cout dp(N)
  
```

time: $O(n)$

mem: $O(n) \rightarrow O(1)$

\rightarrow \rightarrow $dp[i] \rightarrow dp[0]$
vector $dp[4] = \{0, 0, 0, 1\}$

for(i=1 to N){

{0, 0, 0, 1}

vector <int> ndp(4)

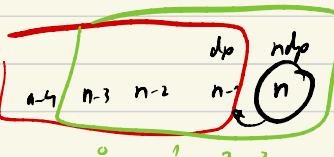
ndp(3) = dp(3) + dp(2) + dp(0)

ndp(2) = dp(3)

ndp(1) = dp(2)

ndp(0) = dp(1)

dp = ndp



}

cout dp[3]

mem: $O(1)$

Line of Wines (Errichto)



wine n වෙත තුන් නොමැති - මෙයි මුදල් නොමැති නොමැති ; $n \leq 100$

Greedy Approach:

$1^{\text{st}}: 2 \times 1 = 2$	}
$2^{\text{nd}}: 4 \times 2 = 8$	
$3^{\text{rd}}: 5 \times 8 = 15$	
$4^{\text{th}}: 2 \times 4 = 8$	
$5^{\text{th}}: 6 \times 5 = 30$	

63

Optimal:

$1^{\text{st}}: 2 \times 1 = 2$	}
$2^{\text{nd}}: 5 \times 2 = 10$	
$3^{\text{rd}}: 2 \times 3 = 6$	
$4^{\text{th}}: 4 \times 2 = 16$	
$5^{\text{th}}: 6 \times 5 = 30$	

64

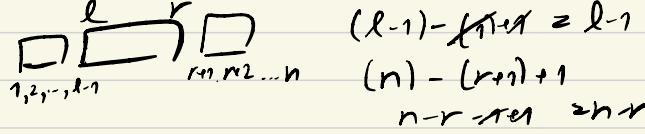
ឧបករណ៍

$dp(l, r) :=$ ឈើនតិចនៃរាយការទាំងអស់ចំនួន (l, r)

$$dp(l, r) = \max \begin{cases} dp(l+1, r) + \text{year} \times \text{deli}(l) \\ dp(l, r-1) + \text{year} \times \text{deli}(r) \end{cases}$$

$$dp(l, r) = \text{year} \times \text{deli}(l), l > r$$

$$\text{year} = n - r + l - 1$$



Recur

int solve (int l, int r) {

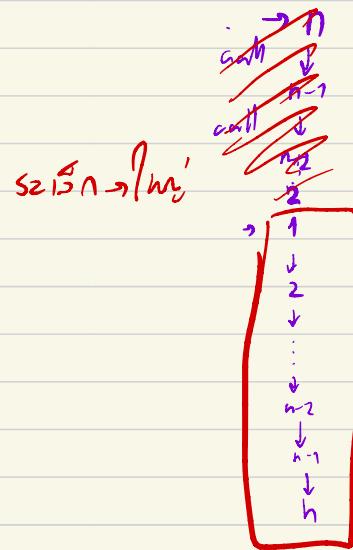
$$\text{year} = n - r + l - 1$$

```
if (l > r) return year * deli(l);  
if (dp[l][r] != 0) return dp[l][r];
```

```
dp[l][r] = max (solve(l+1, r) + year * deli(l), solve(l, r-1) + year * deli(r));  
return dp[l][r];
```

}

cont solve (1, n)



Iterative

```
for (sz: 1 > N) {
    for (l: 1 > N) {
        r = l + sz - 1
        if (r > n) break
        year = n - r + l - 1
        if (l + n <= n)
            dp(l)[r] = max(dp(l+1, r) + year * deli(l), dp(l)[r])
        if (r - 1 >= 1)
            dp(l)[r] = max(dp(l, r-1) + year * deli(r), dp(l)[r])
        if (l == r)
            dp(l)[r] = year * deli(l)
    }
}
```

$O(n^2)$

Coin Combination II

សរុបដើម្បី n នៃការបូលអ៊ីញ្ញ 1, 2, 4 តើតិចតិវត្ថុ និងតាមលក្ខណៈបីដែលបានបង្ហាញ ដូចខាងក្រោម $\{1, 1, 2\} = \{1, 2, 1\}$; $1 \leq n \leq 10^5$

$dp(i, k)$:= សរុបដើម្បីតួនាទី i នៃការបូលអ៊ីញ្ញតាមការបង្ហាញ k

$O(n) O(k)$

$dp(i, k) = \sum_{i=0}^{k-1} dp(i - \text{coin}[k], k) - O(k)$

$dp(i, k) = \begin{cases} 1 & ; i = 0 \\ 0 & ; i < 0 \end{cases}$

use quick sum to opt

$i = 1, 1, 1, 1$	1	2	3	4	5	6	7
$1, 1, 2$	1	2	2	4	4	7	6
$2, 2$	2	1	2	1, 2	1, 2, 2		
4				2, 2	1, 2, 2		
				4	1, 4		

$O(n \cdot k \cdot k) = O(n \cdot k^2)$

$dp(4, 2) \quad 1, 1, 2 \quad 2, 2$

$n \backslash k$	①	②	④
0	1	1	1
1	1	0	0
2	1	2	0
3	1	1	0

$3 \rightarrow ②$

$\begin{matrix} ① \\ ② \end{matrix} \rightarrow 0$

$\begin{matrix} ④ \\ ③ + ④ \\ ① + ② \end{matrix}$

$dp(0) = \{0, 0, 0\}$

for ($k: 0 \rightarrow L-1$) $\rightarrow ①, ②, ④$

for ($i: \text{coin}[k] \rightarrow N$)

$dp(i) += dp(i - \text{coin}[k])$

}

Time/Space: $O(n \cdot w)$

Frog

និង នៅពេល ឈរការណ៍ ឃុំ ស្ថិតិយវិធី ការប្រើប្រាស់ នូវការគ្រប់គ្រង ទីតាំងនឹងវិសាង $|h(i) - h(i+k)|$ ទាំងអស់នៅក្នុងការពិនិត្យចុច្ចាប់ និងការរៀបចំ

$$1 \leq h(i) \leq 10^9, \quad 1 \leq n \leq 10^3, \quad 1 \leq k \leq n \leq 10^3$$

Classical DP

Grid DP

	1	2	3	4
1	5 → -2	-1	3	
2	↓ 10	5	7	
3	-4 → 2	3	2	
4	10	4	9 → 8	
5	1	2	5	7

(R-1) x (C-1)

minimum (R, C)

O(R,C)

$r \geq 1$ $dp(r, c) :=$ ការណើរដ្ឋសាន្តនីរវាង (1,1) នៃចនាសម្រាប់ (r,c)

$$dp(r, c) = arr(r, c) + \max \begin{cases} dp(r-1, c) & ; r \geq 2 \\ dp(r, c-1) & ; c \geq 2 \end{cases}$$

$$dp(r, c) = arr(1)(1) ; r=1 \text{ and } c=1$$

$$dp(r, c) = -\infty ; r \leq 0 \text{ or } c \leq 0$$

Island

(1,1)

0	1	1	0	0
0	1	1	1	0
0	1	1 → 3	1	1
0	1	1 → 1	1	1
0	1	0	0	0

(R, C)

O(R.C)

$dp(r, c) :=$ តម្លៃរបស់ក្រុងក្រាមទិន្នន័យចនាសម្រាប់ (r,c)

$$dp(r, c) = \begin{cases} 0 ; arr(r, c) = 0 \\ 1 + \min (dp(r+1, c), dp(r+1, c+1), dp(r, c+1)) \end{cases}$$

$$dp(r, c) = 0 ; r \leq 0 \text{ or } c \leq 0 \text{ or } r > R \text{ or } c > C$$

for (r: R → 1)

for (c: C → 1)

$$dp(r)(c) = 1 + \min (\dots)$$

Event Point

0	0	0	0	0	0
---	---	---	---	---	---

+2 +2 +2

+1 +1

+3
+4 +9 +4

1	3	2	6	7	4
---	---	---	---	---	---

+1 -1

+2 -2

op → +V [a, b]

↳ ការគាំទ្ធូនៅ idx នៃការ ធនាគារការពាណិជ្ជកម្ម

for (q: 1 → Q)

$$ep[a] += V, ep[b+1] -= V$$

int sum = 0

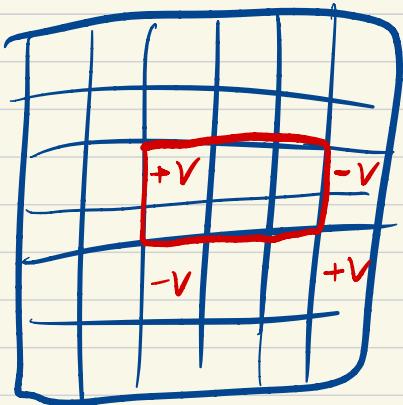
for (i: 1 → N) {

$$\text{sum} += ep[i] \rightarrow qsl[i] = qsl[i-1] + ep[i]$$

rest[i] = sum

y

Event point 2D



$\text{ep} \Rightarrow \text{qs}$
 $\text{qs}(r, c) \quad (r_1, c_1) \quad (r_2, c_2)$
 $\text{ep}(r_1)(c_1) += V$
 $\text{ep}(r_1)(c_1+1) -= V$
 $\text{ep}(r_2+1)(c_1) -= V$
 $\text{ep}(r_2+1)(c_2+1) += V$

for ($r : 1 \rightarrow R$)
 for ($c : 1 \rightarrow C$)
 $\text{qs}(r)(c) = \text{qs}(r-1)(c)$
 $+ \text{qs}(r)(c-1)$
 $- \text{qs}(r-1)(c-1)$

LIS (Longest Increasing Subsequence) $O(n^2) \rightarrow O(n \log n)$

6 3 4 7 5 6

\hookrightarrow b-search
 \hookrightarrow segment tree

$$\text{LIS} = \{3, 4, 5, 6\}$$

$\text{dp}(i)$:= គម្រោងរាប់កំណត់ចុចការខ្ពស់រាយក្នុង seq រហូតដល់ i

$$\text{dp}(i) = \max(\text{dp}(j)) + 1; j < i \text{ and } a(j) < a(i)$$

$$\text{dp}(i) = 1; \text{any } i$$

Satisfy 1 condition first

Segment tree \rightarrow query $\max(l, r) \rightarrow O(1 \log n)$ Optimization

1. Sort array $\{3, 4, 5, 6, 6, 7\}$
 $a(j) < a(i) \& j > i$

$\text{dp} \quad \frac{1}{6} \quad \frac{1}{3} \quad \frac{2}{4} \quad \frac{3}{7} \quad \frac{3}{5} \quad \frac{4}{6}$

2. find $\max(\text{dp}(1, i-1)) + 1$

B-search

$\text{mnp}(i)$ = និវាទនៃវឌ្ឍន៍រាយក្នុង i នីមួយៗ

lower_bound
រាយក្នុងនៃវឌ្ឍន៍រាយក្នុង $\geq a(i)$

6 3 4 7 5 6

1	2	3	4	5	6	
10	15	6	8	2	9	
bf_idx	-1	1	-1	3	-1	4

$\text{mnp} / \begin{matrix} 3 \\ 1 \end{matrix} \quad \begin{matrix} 4 \\ 2 \end{matrix} \quad \begin{matrix} 15 \\ 3 \end{matrix} \quad \begin{matrix} 6 \\ 4 \end{matrix}$

$\text{mnp.size()} \approx \text{LIS}$

$\text{mnp} \quad 2 \quad 8 \quad \begin{matrix} 9 \\ 6 \end{matrix} \rightarrow \underline{\underline{689}}$

vector<int> mnp strictly increasing

for (i: 1 to N) {

 int h = lower_bound(mnp.begin(), mnp.end(), a[i]) - mnp.begin()

 if (h == mnp.size())

 mnp.push_back(a[i])

 else

 mnp[h] = a[i]

}

cout << mnp.size()