Quick sum

1 + 4 + 9 = 14

	1	2	3	4	5	6
	3	1	4	9	10	2
Qs	3	4	8	17	27	29

~~O(n)~~ Per Query

↓

O(1) Per Query

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

$Qs(0) = 0$

$sum(a,b) = qs(b) - qs(a-1)$

$sum(2,4) = qs(4) - qs(1)$

$= 17 - 3 = 14$

```
void preprocess(){
    for(i=1->N){
        qs[i] = qs[i-1] + arr[i]
    }
}
int sum(int a,int b){
    return qs[b]-qs[a-1]
}
```

O(n+Q)

↓ preprocess

↳ จน.คำถาม (ตอบได้ใน O(1))

DP

จำ ✱

→ เทคนิค optimize : exp → poly

→ 1. Overlapping Subproblem
  2. Opitimal Substructure

Classical → Pattern

✱ Non-Classical

Fibonacci

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

$f(1) = 1$

$f(2) = 1$

$f(5)$ → $f(4)$, $f(3)$

$f(4)$ → $f(3)$, $f(2)$

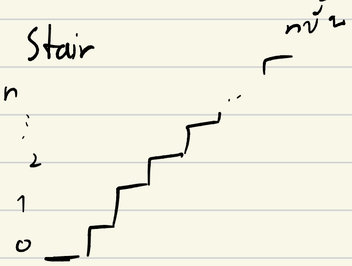
$f(3)$ → $f(2)$, $f(1)$

→ $f(k)$

$f(n) \to f(n-k)$

Over lapping subproblem

Stair



ก้าวสั้น 1 ขั้น
ก้าวยาว 2 ขั้น

จะมีกี่วิธี ในการไปถึง ชั้นที่ n

n=1 ; 1

n=2 ; 2 / 1,1 2

n=3 ; 3 1,1,1 2,1 1,2

⋮

1. นิยาม ให้ชัดเจน
2. คิด recurrence
3. คิด base case

$dp(i) :=$ จำนวนวิธี ในการมาถึงชั้นที่ i

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

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

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

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

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

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

$dp(2)$ ↙ 3 ↘ $dp(1)$

Coin Combination I

รวมเงิน n บาท โดยมีเหรียญ 1, 2, 4 จะทำได้กี่วิธี

dp(3) → 1,1,1 / 1,2 / 2,1 } 3

dp(4) → 1,1,1,1 / 1,1,2 / 1,2,1 / 2,1,1 / 2,2 / 4 } 6

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

dp(i) = dp(i-1) + dp(i-2) + dp(i-4) → O(1) + O(1) + O(1) = O(1)

dp(i) := จำนวนวิธีที่รวมเงินได้ i บาท จากเหรียญ 1,2,4

O(1) × n = O(n)

Push / Pull

```
dp(0) = 1
for(i: 0 => 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	7

Iterative / Recursive

Recur (Top down) → 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)
```

] One of them

$O(3^n)$ → $O(n)$

expo poly

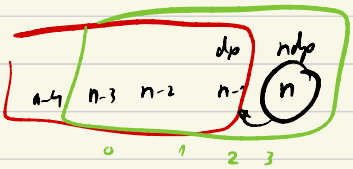
Iterative (Bottom-up)

Pros: Faster, Easy to manage Cons: Hard

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

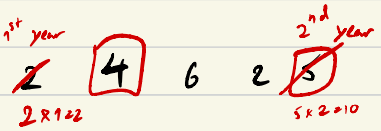
time: $O(n)$
mem: $O(n) \rightarrow O(1)$

```
vector dp(4)={0,0,0,1}   // -3 -2 dp[-1] dp[0]
for(i:1→N){              // {0,0,0,0}
    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 (Enrichto)



wine n ขวด ขายปีละขวด เงินที่ได้ = ค.อร่อย x จน.ปีที่ผ่านไป ของ ทำ ให้ ได้เงิน มากที่สุด ; $n \le 100$

Greedy Approach:
1st : 2×1 = 2
2nd : 4×2 = 8
3rd : 5×3 = 15
4th : 2×4 = 8
5th : 6×5 = 30
} 63

Optimal:
1st : 2×1 = 2
2nd : 5×2 = 10
3rd : 2×3 = 6
4th : 4×4 = 16
5th : 6×5 = 30
} 64

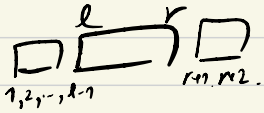
อะไรก็ได้ก่อน

$dp(l,r)$:= เงินที่เยอะที่สุดในการขายจนเหลือช่วง $[l,r]$

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

$dp(l,r) = year \times deli[l]; \quad l==r$

$year = n-r+l-1$



$(l-1) - 0 = l-1$

$(n) - (r+1) + 1$

$n-r-1+1 = n-r$

Recur

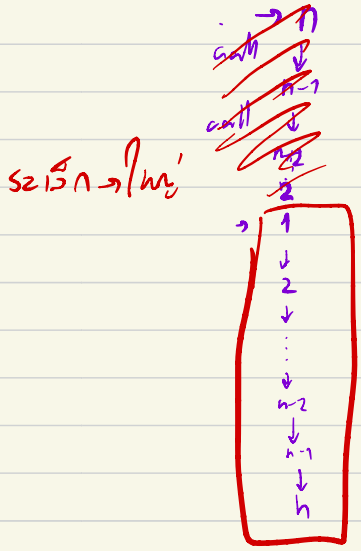
```
int solve(int l, int r){

    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]
}

cout 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+1 <= 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

รวมเป็น n บาท โดยมีเหรียญ 1, 2, 4 ได้กี่วิธี เรียงลำดับของเหรียญไม่สำคัญ {1,1,2} = {1,2,1} ; $1 \le n \le 10^5$

dp(i, k) := จำนวนวิธีที่รวมกันได้ i บาท โดยเหรียญล่าสุดคือเหรียญที่ k

O(n) O(k) k

$dp(i,k) = \sum_{l=0} dp(i - coin[l], l)$ — O(k)

↓ use quicksum to opt

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

4 = 1,1,1,1 ; 1,1,2 ; 2,2 ; 4

1	2	3	4	5	6	7
1	2	2	4	4	7	6
	1,1	1,1,1	1,1,2	1,1,1,1		
	2	1,2	1,3,2	1,1,3,2		
			2,2	1,2,2		
			4	1,4		

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

dp(4, 2) 1, 1, 2 (2,4) ; 2, 2

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

3 → ②

① ↙ [1] + ②

② → 0

④

→ ④

[3] + ④

↓ ↘

① ② ④

0 1 2 ... Less

dp(0) = 1 {①, ②, ④}

```
for (k: 0 -> L-1) { -> ①, ②, ④
  for (i: coin[k] -> N) {
    dp[i] += dp[i - coin[k]]
  }
}
```

Time/Mem: O(n·k)

Frog

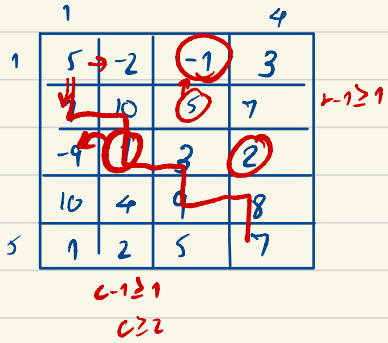
มีหิน n ก้อน แต่ละก้อนสูง $h(i)$ กบเริ่มที่ก้อนแรก $(i=1)$ สามารถกระโดดได้โดยใช้พลังงาน $|h(i) - h(j+k)|$
จงหาพลังงานที่น้อยที่สุดในการที่กบกระโดดไปยังหินก้อนสุดท้าย

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

Classical DP

Grid DP

(R-1)+(C-1) ตารางขนาด (R, C) $O(R \cdot C)$



$dp(r,c)$: = ค่าที่มากที่สุดจากเริ่มที่ $(1,1)$ และมาถึง (r,c)

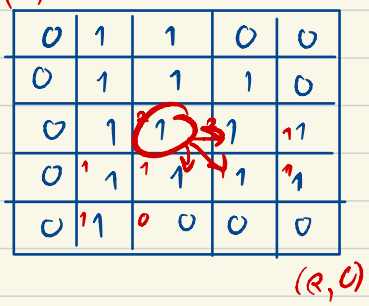
$$dp(r,c) = arr[r][c] + \max \begin{cases} dp(r-1,c) & ; r \geq 2 \\ dp(r,c-1) & ; c \geq 2 \end{cases}$$

$dp(r,c) = arr[1][1]$; $r=1$ and $c=1$

$dp(r,c) = -INF$; $r \leq 0$ or $c \leq 0$

Island

$O(R \cdot C)$



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

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

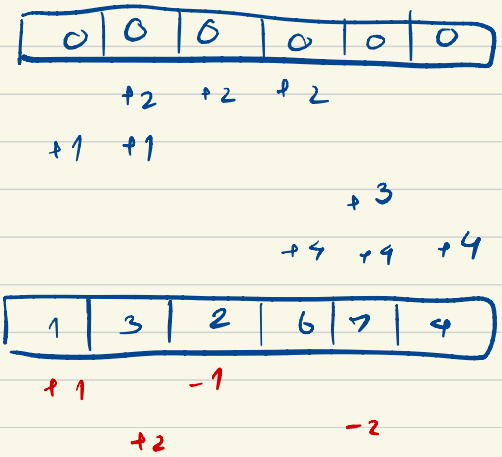
$dp(r,c) = 0$; $r \leq 0$ or $c \leq 0$ or $r > R$ or $c > C$

```
for (r : R → 1)
    for (c : C → 1)
        dp[r][c] = 1 + min(...)
```

Event Point

$O(n)$

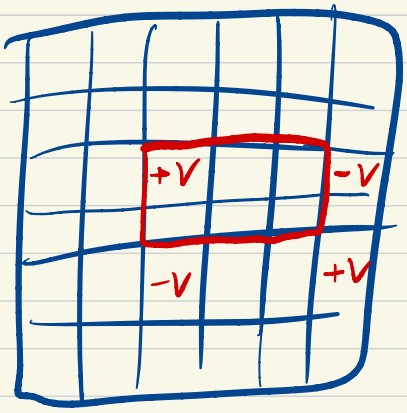
op → +V ในช่วง [a,b]
↳ ถามค่าในช่อง idx หลังจากทำคำสั่งทั้งหมด




```
for (q : 1 → Q)
    ep[a] += V, ep[b+1] -= V

int sum = 0
for (i : 1 → N) {
    sum += ep[i] → qs[i] = qs[i-1] + ep[i]
    res[i] = sum
}
```

Event point 2D



ep → qs

สุดเริ่ม (r1, c1) (r2, c2) สุดจบ

ep(r1)(c1) += V
ep(r1)(c2+1) -= V
ep(r2+1)(c1) -= V
ep(r2+1)(c2+1) += V

```
for(r: 1→R)
  for(c: 1→C)
    qs(r)(c) = qs(r-1)(c)
             + qs(r)(c-1)
             - qs(r-1)(c-1)
```

LIS (Longest Increasing Subsequence) O(n²) → O(nlogn)

↳ b-search
↳ segment tree

6 3 4 7 5 6

LIS = (3, 4, 5, 6)

dp(i) := ความยาวที่มากที่สุดถ้าตัวสุดท้ายใน seq เป็น ตัวที่ i

dp(i) = max(dp(j)) + 1 ; j < i and a(j) < a(i)

dp(i) = 1 ; any i

Satisfy 1 condition first

Segment tree → query max(l, r) → O(logn) Optimization

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

	1	2	3	4	5	6
dp	1	1	2	3	3	4
	6	3	4	7	5	6

2. find max(dp(1, i-1)) + 1

B-search

mnp(i) := เก็บค่าที่น้อยที่สุดที่เป็น i ถ้าตำแหน่งที่ i

6 3 4 7 5 6

mnp: 3 (1), 4 (2), 5 (3), 6 (4)

mnp.size() == LIS

lower_bound
หาตำแหน่งแรกสุดที่มีค่า ≥ a[i]

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

mnp	2	8	9	→ 689
idx	5	4	6	

vector<int> mnp → strictly increasing

```
for(i: 1→N){
    int k = lower_bound(mnp.begin(), mnp.end(), a[i]) - mnp.begin()
    if(k == mnp.size())
        mnp.push_back(a[i])
    else
        mnp[k] = a[i]
}
cout << mnp.size()
```