Longest increasing subsequence

08 October 2023 13:01

→ 8 4 6 1 9 10 2 5 7 0

[1, 2, 5, 7]
[4, 6, 9, 10] } → 4 ans = 4

[3 1 5 2 7]

ans = 3

(1, 5, 7) or (3, 5, 7)
(1, 2, 7)

(1, 5)
(1, 2)
(1, 7)
(3, 7)

Brute force

① Generate all subsequences. Time = O(2^N)

② Using 2 for loops. Time = O(N^2)
Space = O(N)

i i i i i
nums = [3, 1, 5, 2, 7]

[3, 100, 4, 5, 6]

curCount = ~~2~~ 3 0
maxCount = 3

count = ~~1~~ 2
min = 3
max = 100

3, 100, 101, 102, 4, 5, 6, 7

arr: [3, 1, 5, 2, 7]

dp: [1, 1, 2, 2, 3]
(length)

for(i=0; i<n)
  for(j=0 to i)

③ Binary search. O(N log N)

ans = 5

nums: 9 2 4 10 1 6 7 15 3

[2, 4, 6, 7, 15]

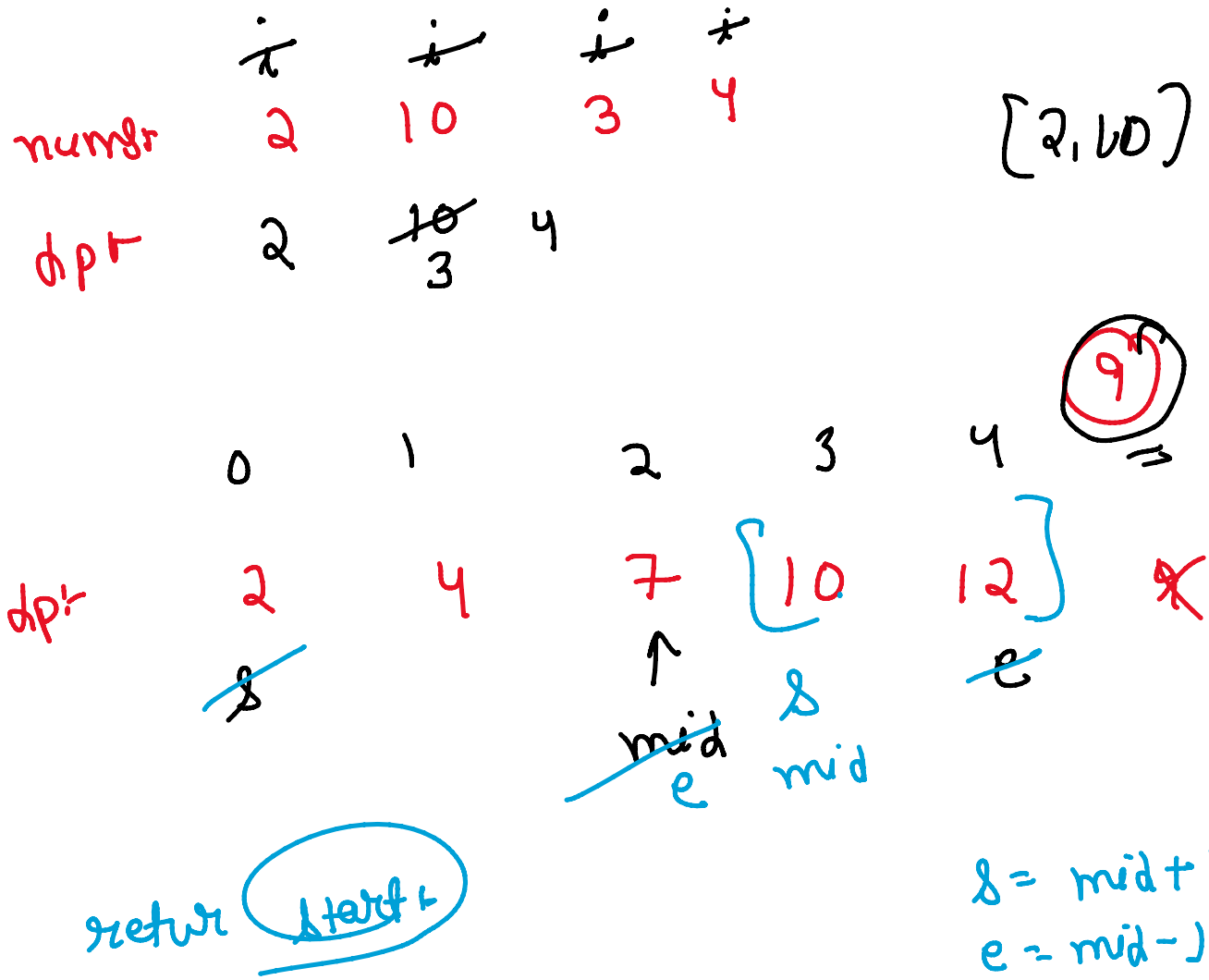
dp: 9 4 10 7 15
2 3 6
1

3 ans

ans = dp.length
= 5

nums: 2 10 3 4 7
dp: 2 10 4 7
3

ans = [2, 3, 4]
= 3


i i i i
nums 2 10 3 4
[2,10]
dpr 2 10 4
3
9
0 1 2 3 4
dp: 2 4 7 10 12
s
mid
e
s
e
return start
s = mid+1
e = mid-1

01 Knapsack

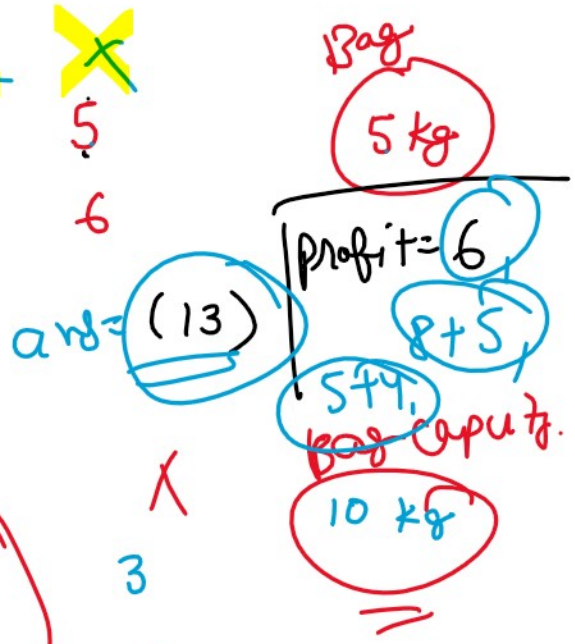
08 October 2023 13:53

n=4

weights : 1 2 4 5

values : 5 4 8 6

get max profit/value.



max ratio? (greedy) ✗

weights 5 8

values 9 10

✗ 3 7

ans = 10

Greedy.

~~1. Take ratio~~

~~2.~~ Take max value

DP

1. Explore all possibilities

Time : $O(2^n)$ optimize ↓

~~general~~ max (include, exclude).

weights : 1, 2, 4, 5

values : 5, 4, 8, 6

w = Bag 5 kg

Base case :

if (w == 0 || n == 0)

return 0;

w = sum capacity of bag

n = no. of items

Recursive code

```
if (cwow <= totalW)
include = value[i] + fun(weights, values, W_T - W_c, n-1);
exclude = 0 + fun(weights, values, W_T, n-1);
```

no. of items

capacity of the bag

fun(4, 5)

fun(3,0) fun(3,5)

fun(2,1) fun(2,5)

fun(1,1)

fun(0,0) fun(0,1)

weights

1, 2, 4, 5

values

5, 4, 8, 6

max(include, exclude)

[N] [W]

[4] [6]

24

Time - O(N*W)

Space - O(N*W)

n=3

capacity = 4

weights [3, 2, 2]

[1.8 1, 1]

values: [1.8, 1, 1]

value/weight = 0.6 0.5 0.5

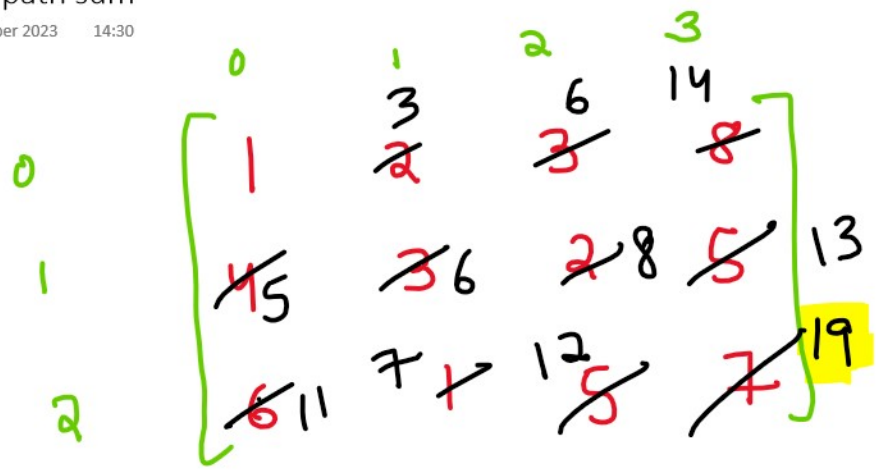
ans = (1+1) = 2

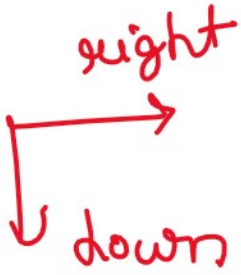
(greedy)

ans = 1.8

Min path sum

08 October 2023 14:30





(1+2+3+1+5+7) = 9

M.W. Top down (Bottom up)

grid[i][j] += min(grid[i-1][j], grid[i][j-1])

(top) (left)

Time: O(N*M)

Space: O(1)

LCS (Longest common subsequence)

08 October 2023 15:10

text1 :- a b c d e

text2 :- a a z y c

'ac' ans = 2

text1 :- a b c d e

text2 :- a e c

ae / ac ans = 2

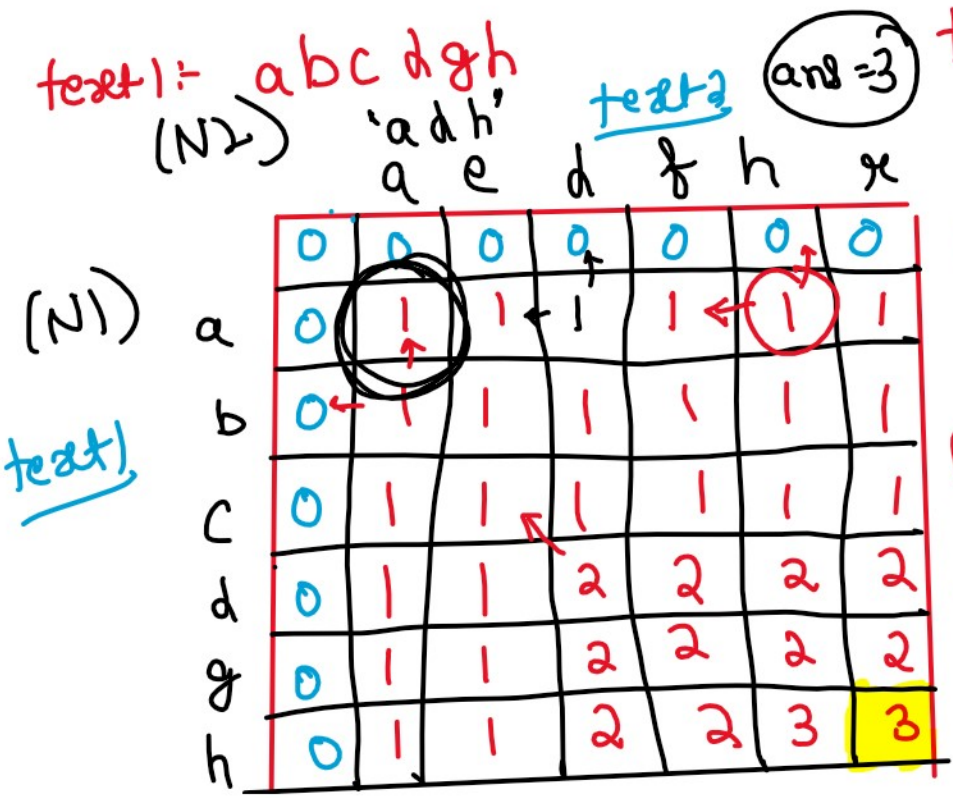
2^{N1} 2^{N2}

$2^{N1} * 2^{N2} = 2^{N1+N2}$

text1 :- a b c d g h

(N2) 'a d h' ans = 3

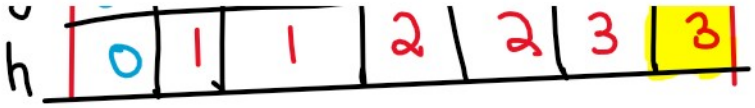
text2 :- a e d f h r



1. if text1[i] == text2[j]
   go to diagonal (+1)
2. In other cases
   max(left , top)

'abcd'

'aed'



'aed'

(1,0)

'ab' '~~a~~'

Time - O(N1 * N2)

Space - O(N1 * N2)

optimise ?

O(N2)

2 rows

Wildcard Matching

08 October 2023 15:44

? → single char

* → zero or more chars

a?b* →
- aabbb (true) — ? = a, * = bb
- aaabb (false)
- aebc (true) — ? = e, * = c
- abcc (false)
- acbbz (true) — ? = c, * = bz
- abb (true) — ? = b, * = ''

s = albmnc

p = a?b*c

? = l   * = mn

a?b*
albmn

(p) columns, (s) rows; (N*M)

		a	?	b	*	c
	T	F	F	F	F	F
a	F	T	F	F	F	F
l	F	F	T	F	F	F
b	F	F	F	T	T	F
m	F	F	F	F	T	F
n	F	F	F	F	T	F
c	F	F	F	F	T	T

"abxy"

① If s[i] = p[j] or p[j] = ? (check diagonal)

② '*' check top or left
- top → one or more chars
- left → empty sequence

Time → O(N * M)

Space → ~~O(N * M)~~ O(N)

al b mn

a?b *