Mert Karobobo melt | Undergraduate |
20160807017

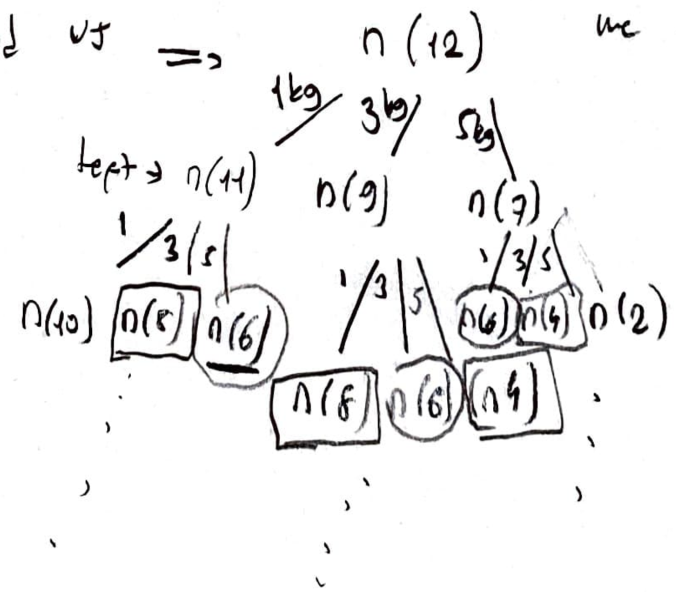
Q-1) a-) $c(n)$ denotes the smallest number of items thief can steal using a bag capacity of n

<u>answer a-)</u> Smallest number of items of denominations w_1, w_2, w_3, w_k needed to take from house for n weight thief have in bag there must be item $w_i \leq n$ (total bag capacity) for the remain items the thief should take $n-w_i$ should be optimal. So $c(n) = 1 + c(n-w_i)$, we don't know which item w_i lead us to optimal solution, we may check all item possibilities. while $w_i \leq n$.

$$c[n] \Rightarrow \begin{cases} 0 & \text{if } n = 0 \\ \min\limits_{1 \leq w_i \leq p} \{1 + c[n-w_i]\} & \text{if } n > 0 \end{cases}$$

b-) Show that this problem has overlapping subproblems property

To show that, we need to assume the thief bag n = some value, for example $n = 12$ kg so recursive approach is lead us $\Rightarrow$ $n(12)$ we have {1kg, 3kg, 5kg} 3 different type items



$\Rightarrow$ at the second iteration we already proved that there are many overlapping subproblems. ie $n(6)$, $n(4)$ $n(8)$ ...

Mert Karababa
20160807017

mert

Undergraduate

c) Write recursive algorithm pseudocode.

before write code I need to define what we have in question

n = bag of thief capacity. / N = size of the items array in have / W = weight of items

MinItems (W[ ], N, n)

if n = 0

return 0

else

total = 0

start ← ∞

for i 1 to N do

if W[i] ≥ n then

total = 1 + minItems (W, N, n - W[i])

start = min (start, total)

End if

return start

Mert Kordobo meef Undergraduate, 20160807017

Q-1) d) Greedy

We claim that the Weight of items are (1, 3, 4)

for example if bag of thief n can have 10

n=10, greedy will choose first 4, then 4 and 1 and, 1

but this can be done better with only 4, 3, 3 weight items.

So if we use greedy algorithm it will cause us to get

more items which we don't want here.

Mert Karabobo mert (Undergraduate)

20160807017

Q-1) $\boxed{e.)}$ Dynamic algorithm

minItems (W[ ], dp[ ], n, N)

for i 1 to N do
dp[i] = ∞

initalize all values to infinity

dp[0] = 0 // base case

for i 1 to n do

// inner loop denotes the index of item array (w)

for j 0 to N do

// i → sum weight

// j → next item index

if (W[j] <= i)

// might include new items

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

for i 1 to N do

return dp[N]

$\boxed{F.)}$ Complexity ⟹ N → length items

w → weight of items

$O(N^* w)$

if the item is greater than 1 the inner loop will run (w - wi) iterations instead w, but need to consider worst case and ignore factors.

Mert Karababa ~~Mert~~ Undergraduate
2016 08 07 017

Q-2) ⟹ We can back up the array after every n insertions.

a) Let c_i' be the charge i-th operation and c_i true cost then

$\sum_{1\le i\le n} c_i \le \sum_{1\le i\le n} c_i'$ for all n that the amortized time $\sum_{1\le i\le n} c_i'$ for that sequence of n operations is a bound on the true time $\sum_{1\le i\le n} c_i$.

b) Let $\Phi_i = i \bmod n$ then when $i \bmod n = 0$, $a_i = n + 0 - (n-1) = 1$.

when $i \bmod n \neq 0$, $a_i\ 1 + (i \bmod n) - ((i-1)) \bmod n = 2$.

Q-3) find Common ID (Array A, Array B)

Array A length ← m
Array B length ← n
(if m≤0 or n≤0 break;) → base case check.
Quicksort(Array B, p, r)
// Defined in class Meterials So i don't write again.

⟹ We can sort the larger array in this problem it is Array B with quicksort would lead us $O(n \log n)$ solution.

for i 0 to m do
binary_search in Array A[i]
if Array B[i] = Array A[i]
print (Array B[i], "common value)
or return

⟹ we iterate in the smaller array, Array A and do binary-search of that element selected Array B.

$O(n\log n) + O(m\log n)$
⟹ $\boxed{= O\,m(\log n)}$

Time Complexity ⟹ We get $O(n \log n)$ complexity from quick sort $O\,n(\log n)$ and $O(m \log n)$ complexity from binary search m times iterate