(1). (a) F eg: $f(n) = n \quad g(n) = n^2$

$n + n^2 \neq O(n)$

(b) T

(c) T

(d) F $f(n) = e^n$, $f(\frac{n}{2}) = e^{\frac{n}{2}}$

$= e^{\frac{n}{2}} \cdot e^{\frac{n}{2}}$

when $n \to \infty$, can't find c, make $ce^{\frac{n}{2}} \geq e^{\frac{n}{2}} e^{\frac{n}{2}}$

(e) F

$\log_2(n!) = \log_2 1 + \log_2 2 + \cdots \log_2 n$

$\leq n \log_2 n$

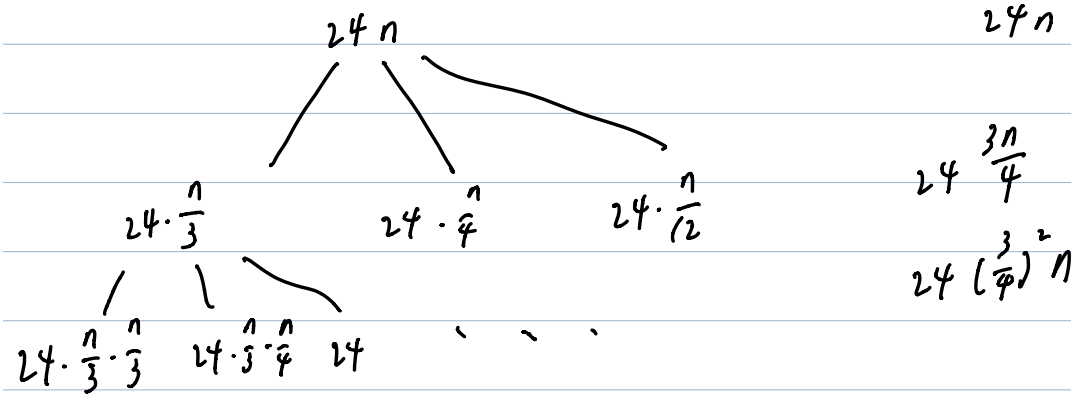
can't find c, make $n \to \infty$, $cn \log_2 n \geq n^2$

(2). (a) master method:

$\because \quad 10n = O(n^{(\log_3 16 - 1)}) = O(n^{\log_3 8})$

$T(n) = \Theta(n^{\log_3 6})$

(b) recursion



/ ⋮

$T(1)$.

$$24n \le T(n) \le 24\left(1+\frac{1}{4}+\left(\frac{1}{4}\right)^2+\cdots\right)n$$

$$T(n) = \Theta(n)$$

(c) $T(2^k) = 2^{k/2} \cdot T(2^{k/2}) + 2^{k+1}k.$

$$S(k) = \frac{T(2^k)}{2^k}$$

$$2^k S(k) = 2^k \cdot S(k/2) + 2^{k+1}k.$$

$$S(k) = T(k/2) + 2k$$

$$S(k) = \Theta(k)$$

$$T(2^k) = 2^k \Theta(k)$$

$$T(n) = n\,\Theta(\log n)$$

$$T(n) = \Theta(n \log n)$$

c d) $T(n) = 2T(\frac{n}{2}) + \frac{4n}{\lg n}$

$T(n) = 2^2 T(\frac{n}{4}) + \frac{4n}{\lg \frac{n}{2}} + \frac{4n}{\lg n}$

$= 2^k T(\frac{n}{2^k}) + \frac{4n}{\lg \frac{n}{2^k}} + \frac{4n}{\lg \frac{n}{2^{k-1}}} + \cdots \frac{4n}{\lg \frac{n}{2}} + \frac{4n}{\lg n}$

$\rightarrow k = \lg N \qquad T(n) = N T(1) + \frac{4n}{1} + \frac{4n}{2} + \cdots \frac{4n}{k}$

$= N T(1) + 4n \lg k$

$= N T(1) + 4n \lg(\lg N)$

so: $T(n) = \Theta(N \lg(\lg N))$

6. c1) use merge sort

i .... ! j ...

when merging: if $a[i] > a[j]$

num = num + len($a[i] \cdots a[j-1]$)

continue merge

if $a[i] < a[j]$

continue merge.

c2) $T(n) = 2T(\frac{n}{2}) + n$

as (1),

use master method

$T(n) = O(n \log n)$

(3) if: $i < j$, only when $a[i] > a[j]$, exchanging takes place.

so the number of inversions is the number of exchanges.

(4).

we take each leg as an exchanges.

★ satisfy all k constraints can be viewed as we should sort players by the order of prizes.

so, we just need to calculate the number of inversions

(5). similar to (4). the empty position will be filled with the unused prices in ascending order and the performing bubble sort as (4)

example: (a, 2) (b, 3) $N = 4$

a	b	c	d
2	3	1	4

ascending

correctness:

the non-constraints array is in ascending order, so their position will not be swap, they will stay their relative position. For the constraints array, the algorithm is still O(N log N).

Problem 7.

(1). just need to compute $\log_2 N$

if $\log_2 N$ is a integer, the problem is solvable.

if $\log_2 N$ is not a integer, the problem can't be solved.

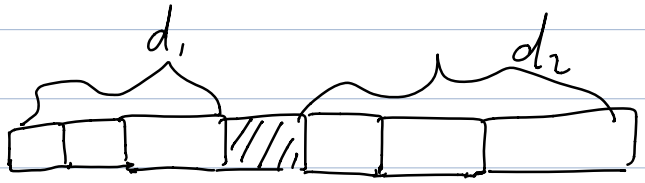
(2)

pseudo-code:

```
process (N, pos):
    if  N == 1
        return;
    if  pos % 2 == 0:
        move left
    else:
        move right
    process (N/2, ⌈pos/2⌉)
```

$(\log_2 N)$

(3).



we can move to left or right

so. possibilities is $2^{max\_movestep}$

$max\_movestep = \log_2(d_1 + d_2)$

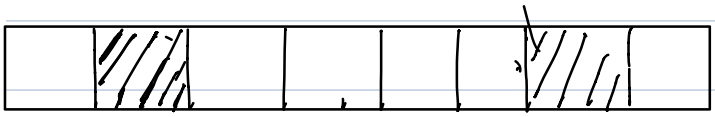
so there are $O(d_1 + d_2)$ possibilities.

(4). $dp[i] = 1$ means $1 \sim i$ can finish. 0 means not.

when meeting a block, we use its first left and right block as its left, right. (if don't have, use boundary).

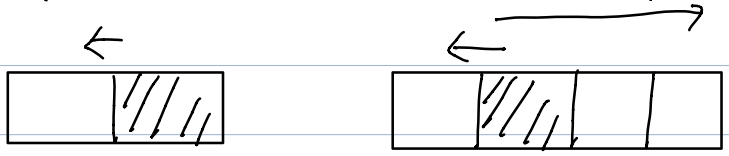
then we use (3), find this block's possibilities. of the status pick the status that meet the condition. fill the dp array.

eg:



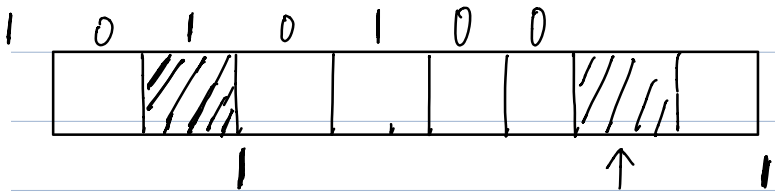
$dp[0] = 1$ eles $dp[i] = 0$ first block $d_1 = 1$ $d_2 = 4$

find all status that meet condition is



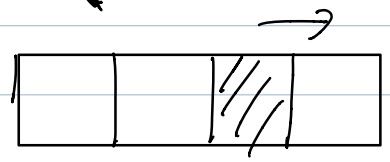
so $dp[2] = 1$ $dp[4] = 1$

second block.



$d_1 = 4$ $d_2 = 1$

find all status that meet condition is only one ①



$dp[8] = 1$

return $dp[end] : dp[8]$

$T(n) = \Theta(N)$

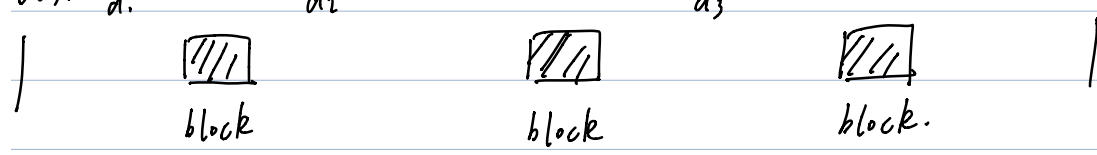
(5) when $i < j$:

if $1 \sim i$ can be solved and $i+1 \sim j$ can be solved

then $1 \sim j$ can be solved.

sub-problem: Judge whether $i-j$ can be solved.

(6).



$N \leq T(N) = O(d_1 + d_2) + O(d_2 + d_3) + O(d_3 + d_4) \cdots O(d_{n-1} + d_{n+1})$

$= O(2(d_1 + d_2 + d_3 \cdots d_n))$

$= O(2N)$

so: $T(N) = \Theta(N)$