HW #5 – Nov 6 midnight PDF

1.
2. } midterm Exam

3.

1) input: A, B, C (nxn matrix)
output: A·B = C
if this is the case then yes
else NO

목표: $O(n^2)$ (nxn, nxn)

$O(n^3)$ (고등 곱했을 때) — pxq, qxr → pxqxr

a randomized algorithm

1. select a nx1 matrix r whose entries are 0's or 1's → random

$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 1 \\ 11 & & 4 \end{bmatrix}$ $C = \begin{bmatrix} 2 & 1 & 10 \\ 1 & 5 & 11 \\ 2 & 1 & 1 \end{bmatrix}$

$B = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 1 & 4 \\ 1 & 5 & 10 \end{bmatrix}$ $A \cdot B = C$?

$O(n^3)$

$r = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ 0 or 1 ex) $\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$

(nxn)·(nxn)·(nx1) (nxn)·(nx1)

$A \cdot B \cdot r \stackrel{?}{=} C \cdot r$

nx1 nx1

Ex) $A \cdot B \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \stackrel{?}{=} C \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$

nxn · nx1

$A \cdot (B \cdot r) = (C \cdot r)$

$B \cdot r$: nxnx1, $O(n^2)$ — $C \cdot r$: nxnx1, $O(n^2)$

$O(n^2)$

$= O(n^2) + O(n^2) + O(n^2) = \boxed{O(n^2)}$

3 cases.

1. if $A \times B = C$ 100% correct → $AB = C$, $AB - C = 0$

nxn $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

$ABr - Cr = 0$

$(AB - C)r = 0$

2. if $AB \neq C$ 적어도 $\leq \frac{1}{2}$인지

prob[error] $\leq \frac{1}{2}$

if $A \cdot B \cdot r - Cr = 0$
then yes

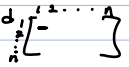
We need to show that the probability that the randomized algorithm return yes even though AB is not the same as C is at most 1/2

Assume that $D = AB - C$

(1) if $AB = C$, then regardless of entries in r, $(AB - C)r$ is a matrix that consists of $n \times n$ 0's.
In this case, the algorithm returns the correct answer.
AB-C를 이루게 각 0으로 이루어진 매트릭스이므로.

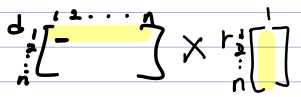
(2) what is the probability that the algorithm returns YES even though AB is not te same as C?
In other words, we want to compute the probability of the following:
- AB is not the same as C, but $Dr = 0$
D를 0으로 이루어진 매트릭스가 아니지만 r과 곱해지면서 0으로 바뀔 경우

Since $D = AB - C$ is a matrix that has at least 1 non-zero entry, let's assume that the non-zero entry is located in the 1st row, 1st column - lets call that entry, d_{11}.



Because of $Dr = 0$, the following should be true:

$(d_{11} \times r_1) + (d_{12} \times r_2) + \ldots (d_{1n} \times r_n) = 0$



we can rewrite this as follows:

$r_1 = -(d_{12} \times r_2 + \cdots d_{1n} \times r_n) / d_{11}$ ⟶ Equation A

Here, we know that there are two possible choices for r_1 (0, 1)

which means that either one of these two can make the equation A true, not both.

In other words, Equation A has 2 quantities which are equal and even though we do not know whether r_1 is 0 or 1, we know for sure that one of these should make the equality in Equation A hold.
Therefore the probability that the algorithm returns an incorrect answer is bounded above by 1/2.

2) input: $A = \{ \quad \}, x$

a set of n distinct integers. $n \geq 2$

output: if $\exists u, v \in A$ $u + v = x$ then yes
else no

Ex) $A = \{-10, 6, 4, 2, 6, 11, 15\}$, $x = 10$

$6 + 4 = 10$

$O(n \lg n)$ 이상인 증명

1. Sort the numbers in set A in an ascending order.
(Note that we may use Heapsort or Mergesort, but neither Insertion sort nor Quicksort can be used because their worst case time complexity is $O(n \times n)$)

Heap sort / Merge sort runtime $O(n \lg n)$

2. Use the binary search Algorithm with the following argument n times, where n is the number of elements in A (that is sorted.)

- for each element a in A, run binary search (A, x-a)

if the result is not 0, then return yes,
otherwise (that is, the index returned by the algorithm is 0) then return no

Since the time complexity of binary search algorithm is $O(\log n)$ and it is repeated run for the number of elements in A (which is n), the runtime of this algorithm is (in step 1, $O(n \lg n)$, in step 2, $O(n \lg n)$)
$O(n \lg n + n \lg n) = O(n \lg n)$.

Binary search

Input: n *[a positive integer]*, $a[1], a[2], \ldots, a[n]$ *[an array of data items given in ascending order]*, x *[a data item of the same data type as the elements of the array]*

index:=0, bot:=1, top:=n

while ($top \geq bot$ and $index = 0$)

$$mid := \left\lfloor \frac{bot + top}{2} \right\rfloor$$

if $a[mid] = x$ **then** $index := mid$

if $a[mid] > x$

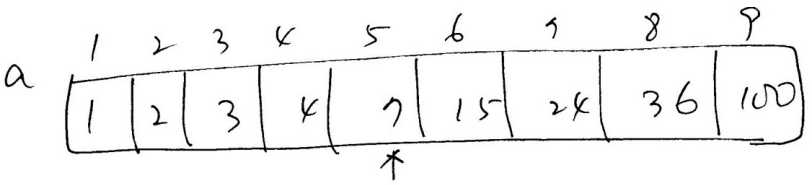
then $top := mid - 1$

else $bot := mid + 1$

end while

[If index has the value 0 at this point, then x is not in the array. Otherwise, index gives the index of the array where x is located.]

Output: *index [a nonnegative integer]*



BinarySearch(a, 36)

mid := $\lfloor \frac{1+9}{2} \rfloor$ = 5 [1,9]

a[5] < 36
"
7

mid := $\lfloor \frac{6+9}{2} \rfloor$ = 7 [6,9]

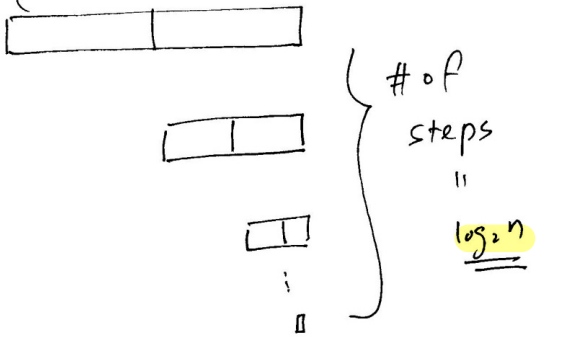
a[7] < 36
"
24

mid := $\lfloor \frac{8+9}{2} \rfloor$ = 8 [8,9]

a[8] = 36

In general,

Assume that the length $= 2^n$, $n \geq 1$



Worst case complexity:

$\log_2 n$

3) ↙ ch 15, 16

Compare the following 2 problems

① 0/1 knapsack problem (→ carry a item or not.)
② fractional ″ (↓ 20% of something)

Greedy) DP.

input: n distinct items, values ($) ○ (↙ positive R), weights (lb) ○ (↙ positive R), W → weight ↑ most weight. knapsack ○

output: optimal load ≤ W

Ex) 5 items

	$ value	(lb) weight
a_1	50	100
a_2	25	75
a_3	100	100
a_4	40	200
a_5	30	60

≤ 250 lb

knapsack

100 75 60
a_1, a_2, a_5
↑ ↑ ↑
50 25 30 = 105
235 ≤ 250

2) 50% a_4 20 20)

3. 0/1 knapsack P → DP (↖ optimal substructure property; ↗ GA X)
fractional knapsack P → Greedy choice property (Need a criteria) (↖ GA)

P2 slide

① items n ≥ 1
- weight positive R^+
- value positive R^+

knapsack ≤ w (lb)

select an optimal load.

P3 slide

	$	lb
item 1	60	10
item 2	100	20
item 3	120	30
		60

$/lb
6 ← item 1
5 ← item 2
4 ← item 3
60+100+80 = 240 $
80/3


≤ 50 lb

1) 0/1 knapsack problem

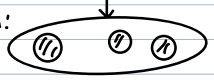
two constrain
① maximize value
② ≤ lb

(1,2) → 160
(1,3) → 180
(2,3) → 220

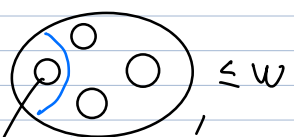
2) fractional knapsack
① optimal value
② ≤ b

$/lb

P4 optimal substructure property

1. an optimal sol
2. claim:

if ① optimal sol is given
this property also composed with optimal solution
3. what if not = ① ②  contradiction

Ex)
1) 0 1 knapsack problem

1. opt sol (definition) : an opt load ≤ w
2. 
≤ w
item$_i$, $1 \le i \le n$

we take any item that uses in optimal solution.
without item$_i$ (from optimal solution), it would no more optimal solution.

Ch15
MCM
ALSP
ASP
2
1
ch 16

item 1	w_1	v_1
item 2	w_2	v_2
item$_i$	w_i	v_i
item n	w_n	v_n

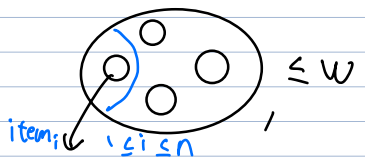
the remaining one is an optimal solution for | some solution

① n-1 items (item i)

② opt load ≤ W - w_i

3. what if not? ① X not the
② contradiction, better solution.

for fractional knapsack problem
① $item_i$ 가 전체일때와 fraction 일때같은값인지 생각한다.



we take any item that uses in optimal solution.
without $item_i$ (from optimal solution), it would no more optimal solution.

item 1	w_1	v_1
item 2	w_2	v_2
⋮	⋮	⋮
$item_i$	w_i	v_i
⋮	⋮	⋮
item n	w_n	v_n

$w_i - W$

① $item_i$ 가 전체일때와 fraction 일때같은값인지 생각한다.

the remaining one is an optimal solution for | some solution ②

after $item_i$ is gone, optimal solution for

① n-1 items, | part of $item_i$ | ↓
that was not selected initially

② W - w (part of $item_i$ initially selected)