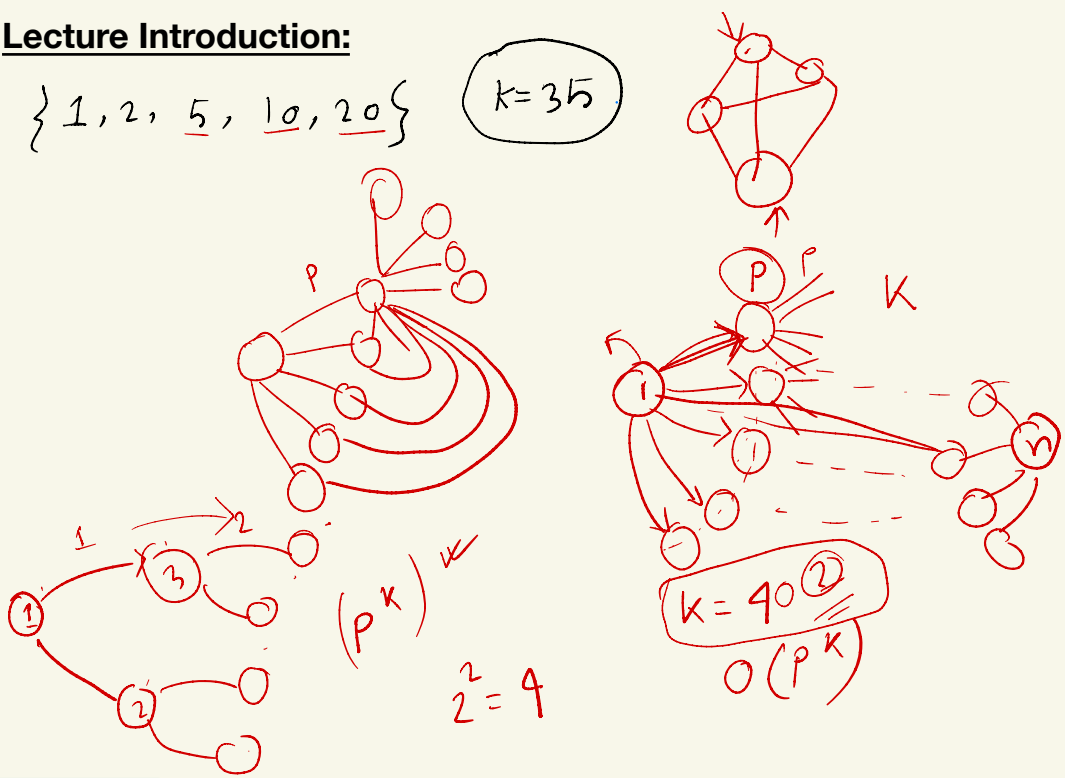
Session 3: Meet in the Middle

Problem 1 (Four Sum): Given **4** arrays **A, B, C, D** with size **n**. Your goal is to take **1** element from each of the arrays, let's say **a, b, c, d** such that **a + b + c + d** = 0. Here, n <= 2000. Problem Link: https://www.spoj.com/problems/SUMFOUR/

Problem 2 (coin change): Given a set **S** containing **n** coins and a number **k**. You have to determine whether it is possible to take some elements from **S** and make the summation of the coins to be **k**. Here, the value of **S_i** and **k** can be very large **(10^9)** and **n <= 40**. Problem Link: https://lightoj.com/problem/coin-change-iv

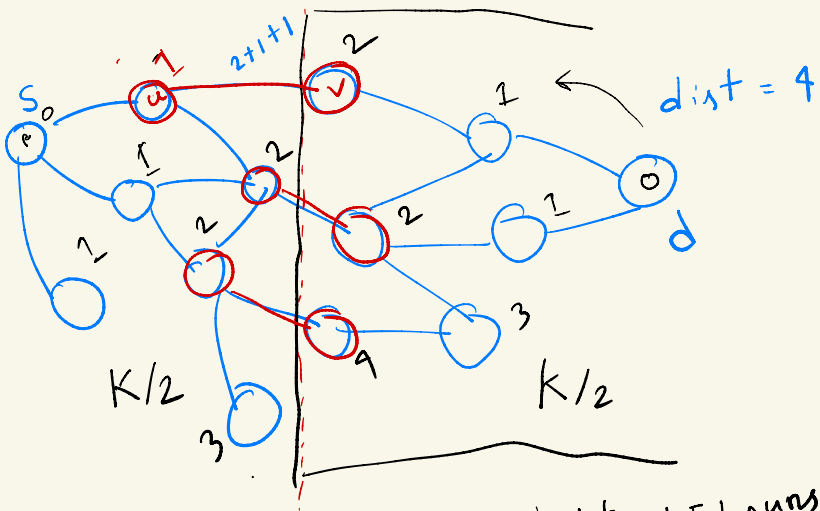
Problem 3 (Bidirectional search): Given a graph containing **n node**s and **e edges**, and 2 nodes, **a** and **b**. You have to determine the distance between **a** and **b**. Keep in mind that the graph is very **dense**, so, BFS doesn't work.

Lecture Introduction:



Perform roughly (2×10^8) ops in 1 sec.

$2^{40} \approx 2 \Rightarrow$



$K = 40$

p

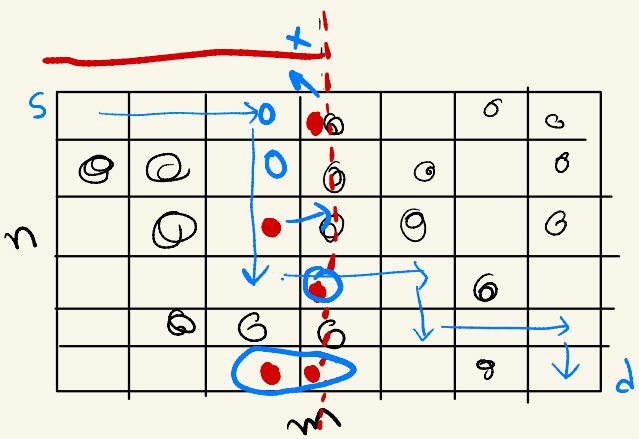
$O(p^{K/2}) \Rightarrow p^{40/2} = \boxed{2^{20}}$

$K/2$

plain bfs: 1.5 hours → meet in the middle.

$\approx$ 1 s

0.00 s

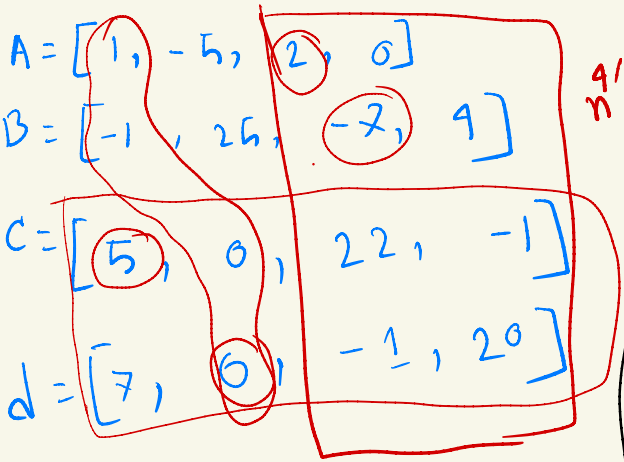


→ pruning technique.

arr[ ], 3 elements s.t. $a+b+c=k$ → n^3

$a+b+c+d$ → n^4

Problem 1 (Four Sum): Given **4** arrays **A, B, C, D** with size **n**. Your goal is to take **1** element from each of the arrays, let's say **a, b, c, d** such that **a + b + c + d** = 0. Here, n <= 2000. Problem Link: https://www.spoj.com/problems/SUMFOUR/



$n^{4/2}$

$O(n^4)$

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

$O(n^2)$

$a+b = -c-d$

$= -(c+d)$

$5+0, 5+7, 0+7, 0$

5 12 7 0

$a+b = -(c+d)$

0 0

[ ]

→ if(---):

→ for - - - ntime 1

$O(n)$

$n \times m \Rightarrow n^2$

$n \times 1 = n$

1000 10^5

$O(nm)$

i - - - - n

j - - - - - - m

[2, 1, 4, 6, 8, 3, 1, 5]

[1, 6, 8]

subsequence: remove 1 or more elements without changing the order

[2, ~~1~~, 4, ~~6~~, 8, ~~3~~, 1, ~~5~~] => [2, 4, 8, 1] ✗

→ Given this array, how many 3-length increasing subsequences are there?

3 nested loops: i, j, k

i < j < k

a[i] < a[j] < a[k]

$O(n^3)$

2 binary indexed tree

4

[2, 1, 4, 6, 8, 3, 9, 5]

3 a[j] 4 2 2

$O(n^{\lfloor 3/2 \rfloor})$

$O(n \log^2 n)$

$O(n^2)$

? 6 ? ⑥

6

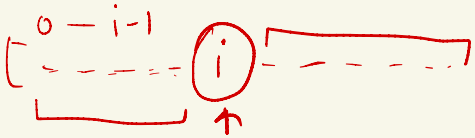
[3 | 1 | 2]

n - loop

~~n~~ loop

for i = 0 → n-1 ;

mid = a[i]



→ for j 0 → i-1 ;
check if a[j] < a[i] → cnt1++

for j i+1 → n-1 ;
- - - - a[j] > a[i] → cnt2++

ans += (cnt1 × cnt2) → permutations
combinatorics

competitive programming

*:

- Recursion
- Dynamic programming
- combinatorics. → 0 - 15
  - a lot of problems → maturity
  - → backtrack pruning