DP with Bitmasking

Dynamic Programming 5
Bitmasking & Results

- Priyansh Agarwal

Array $A = \{5, 6, 4\}$

2^3

$\rightarrow$
$\rightarrow 5$
$\rightarrow 6$
$\rightarrow 4$
$\rightarrow 5, 6$
$\rightarrow 6, 4$
$\rightarrow 5, 4$
$\rightarrow 5, 6, 4$

Space $\rightarrow 2^n \cdot n$

Time to search in subset $= O(n)$

insert in subset $= O(1)$

delete $\rightarrow O(n)$

	Vector	Set	Space
Search	✓ $O(n)$ =	$O(\log n)$ =	$\to 2^n \cdot n$
Insert	✓ $O(1)$ =	$O(\log n)$ =	
Delete	✓ $O(n)$	$O(\log n)$	$O(n\log n)$

✓

$S_1 \cup S_2$ }

$\{5, 6, 4\}$

$\{5, 6\}$

$\{6, 4\}$

$\{0, 0, 0\}$

$\{5, 6, 4\}$

$O(n)$

$S_1 \wedge S_2$ $\qquad$ $S_1 - S_2$

Bitmasking

{ $\boxed{5}$, 6, 4 }

0th 1st 2nd

0 1 2 3

3 2 1 0

→ — — —	000	0
→ 5	001	1
→ 6	010	2
→ 4	100	4
→ 5, 6	011	3
→ 6, 4	110	6
→ 5, 4	101	5
→ 5, 6, 4	111	7

Search an element in susset

$\{5, \boxed{6}, 9\}$

6 is present or not

$\{5, 6\}$

011 → 3

arr[i] present in susset or not

$\boxed{3 \,\&\, (2^1)} > 0$ then

arr[i] is present

$\& \boxed{\begin{matrix} 011 \\ 010 \end{matrix}} > 0$

Subset $\rightarrow$ D

ith element of array is present or not

$$\boxed{D \,\&\, (1 << i)} > 0$$

Yes

o/w No

$O(1)$ $O(1)$

Search $\rightarrow$ $O(1)$

$\{5, 6, 9\}$
0 1 2

Insert $\rightarrow$ $O(1)$

Deletion

$D \mid (1 << i)$

011
OR
100
111

3 | 4
7

$\{5, 6\}$

$1 << 2$
4

$011 \rightarrow 3$

$\{5, 6, 9\}$

if element present take xor

$$\text{if } (D \,\&\, (1 \ll i))$$

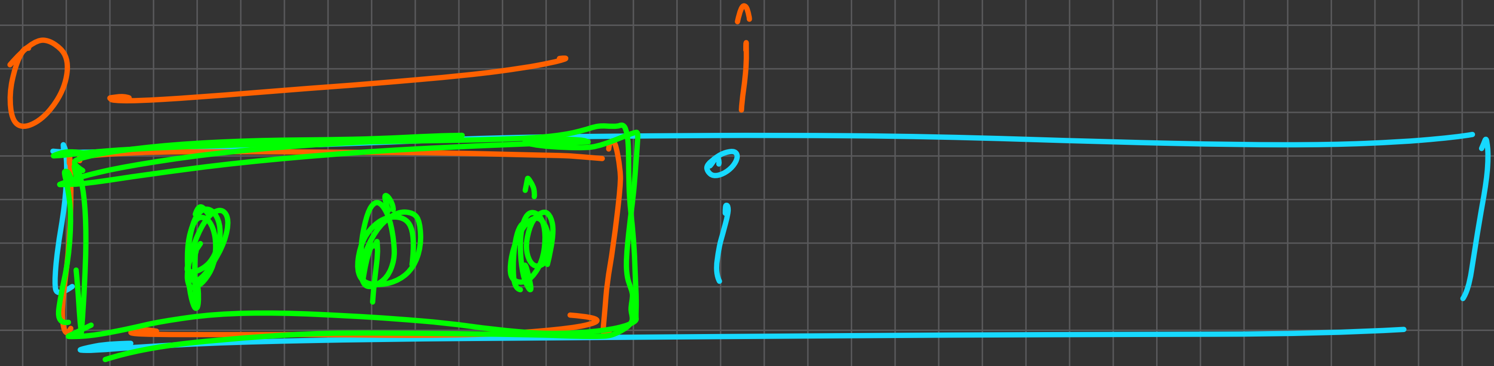
Union

$$D = D \wedge (1 \ll i)$$

$$D_1 \mid D_2$$

Intersection

&



32

$dp[i]$ [mask] [~~used~~]

$\rightarrow 0 - 10^9$

$2^{31} \rightarrow 10^9$

0 1 2 3 $\boxed{4}$ 5

current mask = current mask | (1<<4)

$\underline{\underline{\log_2(1e7)}} \longrightarrow \underline{\underline{20 \to 22}}$

$\underline{\underline{64}}$

$\underline{\underline{2^{64}}}$

$\underline{\underline{\text{map}}}$

map → [10^9 —— 10^{10}]

10^7

array → [0 —— 10^7]

DP with Bitmasking

- Bitmasks
- Basic operations on Bitmasks
- Limitations on “N” (You will need 2^n integers to represent all the subsets)

Problem 1:

Given a list of points on a 2D plane, rearrange these points in any way such that in the final permutation of points, the sum of distances of the adjacent elements is minimized.

euclidean

Constraints: [N <= 15], [-1e9 <= Xi, Yi <=1e9]

15!

=

13,07,67,43,68,00

Points : [{0, 0}, {5, 6}, {1, 2}]

Best permutation -> [{0, 0}, {1, 2}, {5, 6}]]

Ans = Dist(P1, P3) + Dist(P3, P2)

$(0,0) \quad (5,6) \quad (1,2)$

$P_1 \; P_2 \qquad P_2 \; P_3 \qquad P_3 \; P_1$

$$\text{Dist}\left[\{5,6\},\{0,0\}\right] + \text{Dist}\left[\{5,6\},\{1,2\}\right]$$

$$\text{Dist}\left[\{1,2\},\{0,0\}\right] + \text{Dist}\left[\{0,0\},\{5,6\}\right]$$

$P_1 \leftarrow P_2 \leftarrow P_3 \quad P_4 \quad P_6$

$P_1 \quad P_3 \quad P_2 \quad P_4 \quad P_6$

x

$(P_2, P_1) + (P_3, P_2) + (P_4, P_3) \leftarrow (P_6, P_4)$

$(P_1, P_3) + (P_2, P_3) + (P_4, P_2) \leftarrow (P_6, P_4)$

S_3

$P_1 \quad P_2 \quad P_3 \quad P_4$

$P_3 \quad P_2 \quad P_1 \quad P_4$

$P_2 \quad P_3 \quad P_1 \quad P_4$

$3!$

$1 \to 3!$

1 2 3 4 5

$\boxed{7}$ 9

7

7

7

7

5!

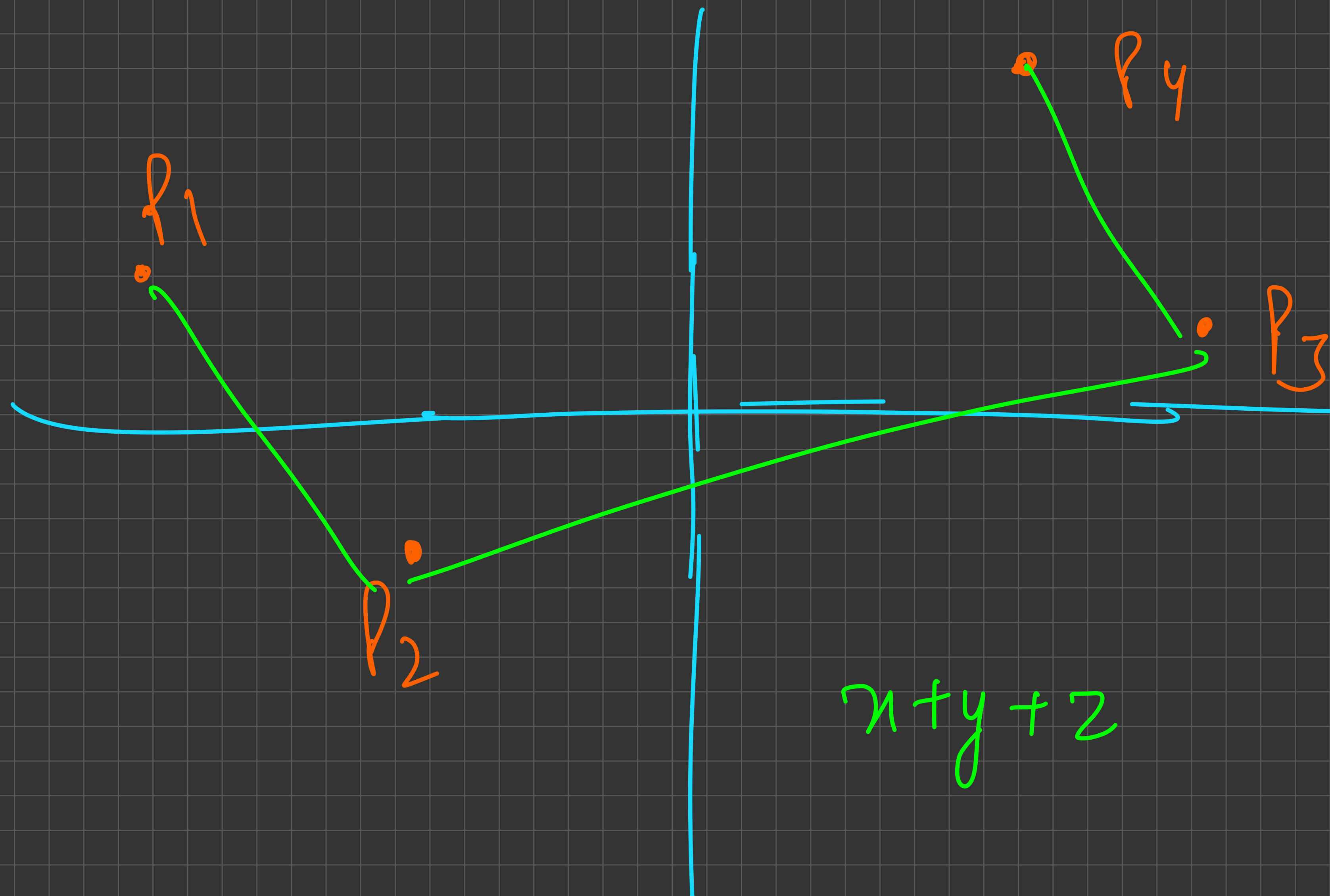
32k n^2

15

2^n

$n!$

1872


P1
P2
P3
P4
x+y+z

$dp[i][mask][last]$

= you have already fixed i points so far, in which mask represents the exact point you have used and last represents last element picked.

= min sum of distances from ith point to $(n-1)$th point provided ith point = last

[5, 4, 7, 8, 9]

$n - i$

$O(n)$

$i \to 5$

mask = picked up print

last = 9

$$dp[i][mask][last]$$

$$= \underline{\underline{\min}} \left\{ \underline{\underline{dist(last, picked)}} + dp[i+1][mask \mid (1<<picked)][picked] \right\}$$

where $mask \,\&\, (1<<picked) == 0$

Problem 1: TC: $O(n^3 2^n)$, SC: $O(n^2 2^n)$

$n^2 \cdot 2^n$

```
state:
    dp[i][bitmask][last element] = minimum sum of distances in the suffix [i... n - 1]
    such tha bitmask represents the elements in the first i - 1 elements and last elements
    represents the last point

transition:
    check for jth point from (0 to n - 1)
    can you pick the jth point as the ith element in the final array or not

    if(bitmask & (1 << j)){ whether jth bit is set or not
        continue;
    }else{
        dp[i][bitmask][last element] = min(dp[i][bitmask][last element],
        (bitmask != 0 ? dist(j, last element) : 0) + dp[i + 1][bitmask | (1 << j)][j]
    }

base case:
    dp[n][(1 << n) - 1][anything] = 0

final subproblem
    dp[0][0][anything]
```

Problem 1: TC: $O(n^2 2^n)$, SC: $O(n*2^n)$

```
state:
    dp[bitmask][last element]
    i = set_bits(bitmask)
    = minimum sum of distances in the suffix [i... n - 1] such tha bitmask represents the
    elements in the first i - 1 elements and last elements represents the last point

transition:
    check for jth point from (0 to n - 1)
    can you pick the jth point as the ith element in the final array or not

    if(bitmask & (1 << j)){ whether jth bit is set or not
        continue;
    }else{
        dp[bitmask][last element] = min(dp[bitmask][last element],
        (bitmask != 0 ? dist(j, last element) : 0) + dp[bitmask | (1 << j)][j]
    }

base case:
    dp[(1 << n) - 1][anything] = 0

final subproblem
    dp[0][anything]
```

$n^2 \cdot 2^n$

Permutation problems $\longrightarrow$ DP with Bitmasking

$n!$

$\Rightarrow$

$\boxed{n^k \cdot 2^n}$

1, 2, 3, 4

$\boxed{n \leq 20 \quad \& \quad \text{permutations}}$

Problem 2: Link (Homework)

- State
  -
- Transition
  -
- Base Case
  -
- Final Subproblem
  -

Trick to identify a DP problem?

Repeating subtasks:

- If I have the answer of state, then why should I calculate it again and waste time

Pro Tips for contests:

- Number of ways problems -> DP, Brute Force or some formula
- Look for small constraints in the problem. (Most probably it would be dp and not greedy)
- Identify states and transition time for each state.
- Calculate time complexity as (number of states * transition time for each state).
- If this number fits into your Time limit (Great), if not, try to see if you can skip some states and still get the right answer.
- Try to reduce the transition time by using some Data Structure or some clever observation if transition time is the bottleneck
- Never try to over optimize. If your current states and transition time fit into your Time Limit, just code it and do not optimize it further.

Common states and transitions with constraints

```
total operations <= 1e8

n <= 125 :

    state: O(n^3), transition: O(1), [n <= 100 O(logn) is possible]
    state: O(n^2), transition: O(n), [n <= 100 O(nlogn) is possible]
    state: O(n), transition: O(n^2)

n <= 5000:
```

Programmer

dp problem or n^2 brute force

```
    state: O(n^2), transition: O(1) [n <= 1000 then O(logn) is possible]
    state: O(n), transition: O(n) [n <= 1000 then O(nlogn) is possible]

n <= 1e6:

    state: O(n), transition: O(1), O(logn)
```

4 6 5

$4 \cdot 10^8$

```
1 second <= operations <= 4 * 1e8
4 second <= operations <= 1e9
```

10^9

1500 — 1800