Q1. SOS DP#1

Regarding SOS DP choose the correct option :

It stands for sum over subset dynamic programming.

It involves good understanding of dp and bitmasks to master.

It introduces a smart way of partitioning a set into disjoint subsets.

All of the above.

Q2. SOS DP#2

Let F[mask] = sum of A[i] for all i such that i is a submask of mask. Then which of the following is true?

i & x == i

i ^ x == x

i & x == x

i ^ x == i

Q3. SOS DP#3

Given a fixed array A of 2^N integers. For all possible bitmasks M of length N find F[mask] = sum of A[i] such that i & x == i.

Algorithm :

```
For mask = 0 to 2^N - 1
    For i = 0 to 2^N - 1
        if (i & mask) == i
            F[mask] += A[i]
```

What is the time complexity of this brute force algorithm?

Q4. SOS DP#4

Given a fixed array A of 2^N integers. For given mask M of length N find F[mask] = sum of A[i] such that i & x == i.

Algorithm :

```
DP[0] = A[0]
solve(mask)
```

```
    if DP[mask] is already calculated
            return DP[mask]
    else
        DP[mask] = A[mask]
        for each set bit i of mask
              DP[mask] += solve(mask with i off)
        return DP[mask]
```

which of the following is true?

The time complexity for this algorithm is O(2^N) and it calculates the F[mask] as required.

The time complexity for this algorithm is O(2^N), however it is not a correct algorithm to solve the problem.

The time complexity for this algorithm is O(3^N), however it is not a correct algorithm to solve the problem.

The time complexity for this algorithm is O(3^N) and it calculates the F[mask] as required.

Q5. SOS DP#5

Let us try to partition a mask S into disjoint submasks.

Define :

```
F(mask, i) = submasks of S which only differ from S in the first i
bits.
DP[mask][i] = sum of all A[x] such that x belongs to F(mask, i).
```

Choose the correct recurrence :

DP[mask][i] = DP[mask with ith bit off][i - 1] + DP[mask][i - 1] if ith bit of mask is on.

DP[mask][i] = DP[mask with ith bit off][i - 1] + DP[mask][i - 1] if ith bit of mask is off.

DP[mask][i] = DP[mask with ith bit on][i - 1] + DP[mask][i - 1] if ith bit of mask is off.

DP[mask][i] = DP[mask with ith bit on][i - 1] + DP[mask][i - 1] if ith bit of mask is on.

Q6. SOS DP#6

In reference to SOS DP - special pairs.

Define :

```
Freq[1..N] : Freq[i] represents the number of times i occured.
DP[mask] = sum of all Freq[x] such that x & mask = x.
```

Now we try to find the number of elements A such that for a given B, A & B = 0.

choose the correct recurrence :

number of such elements = DP[B & B^2]

number of such elements = DP[two's complement of B]

number of such elements = DP[B + (1 << (B & (-B)]

number of such elements = DP[one's complement of B]

Q7. SOS DP#7

In reference to SOS DP - special pairs.

Let the number of bits required to represent the largest number in input be B.

Optimal time and space complexity respectively for solving the problem is:

O(B^2) and O(B^2)

O(2^B) and O(2^B)

O(B * 2^B) and O(2^B)

O(B^B) and O(B^2)

Q8. SOS DP#8

In reference to SOS DP - vowels.

Say we represent each input string as a 24bit integer **(if 'a' is present in the string -> 0th bit is on, 1st bit is on if 'b' is present in string and so on)**.

What is the correct integer to represent "ada" and "def" respectively?

7 and 54

9 and 54

9 and 56

7 and 56

Q9. SOS DP#9

In reference to SOS DP - vowels.

Define :

```
Convert each input string to 24bit number as described in previous
question.
Freq : Freq[i] represents the number of occurences of integer i.
DP[mask] : number of integers i such that i & mask = i.
```

Consider the Algorithm :

```
ANS = 0
N = number of strings in input.
For i = 0  to (2^24 - 1)
    X = DP[i];
    X = N - X
    X = X^2
    ANS = ANS xor X
print ANS
```

choose the correct option :

Algortihm is incorrect as DP[i] is the number of integers which have a non-zero AND with i. Therefore X = N - X should not be done to make the answer correct.

Algorithm is correct because DP[i] is the number of input strings that have a zero AND with complement of i and therefore N - DP[i] is the number of strings that are included in the answer to complement(i).

Algortihm is incorrect as DP[i] is the number of integers which have a zero AND with i. Therefore X = N - X should not be done to make the answer correct.

None of these

Q10. SOS DP#10

Choose the correct time and space complexity respectively for the algorithm as described in the previous question.

Let there be N input strings.

Let 'B' be the alphabet size(24 in the problem statement of vowels).

O(N + B*2^B) and O(2^B)

O(B*2^B) and O(N + 2^B)

O(N + Bx2^B) and O(Nx2^B)

O(B*2^B) and O(2^B)