Bitmask DP in DSA: Beginner to Advanced Guide

Subtitle: Learn how to use bitmasking with dynamic programming to efficiently solve subset and combinatorial problems.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Bitmask DP combines **dynamic programming with bitmasking** to handle problems involving **subsets, combinations, and states** efficiently. It is widely used in **Traveling Salesman Problem (TSP), assignment problems, and combinatorial optimizations**.

Step 1: Core Concepts

- Represent subsets of size n using **integers from 0 to 2^n - 1**
- Each **bit** in the integer indicates whether an element is included (1) or not (0)
- Use DP array dp[mask] to store **optimal solutions for each subset**
- Combine previous subset states to build solutions for larger subsets

Why it matters: Bitmask DP can reduce **exponential complexity** to manageable O(n * 2^n) for small n (≤20).

Step 2: Example Problem — Traveling Salesman (TSP)

- **Problem:** Given n cities and distances, find the shortest path visiting each city exactly once and returning to start
- **State:** dp[mask][i] = minimum cost to visit cities in mask, ending at city i

Step 3: Implementation in JavaScript

```
let n = 4;

let dist = [

  [0,10,15,20],

  [10,0,35,25],

  [15,35,0,30],

  [20,25,30,0]

];
```

```
let dp = Array(1<<n).fill(0).map(()=>Array(n).fill(Infinity));

dp[1][0] = 0; // start at city 0


for(let mask=1; mask<(1<<n); mask++){

  for(let u=0; u<n; u++){

    if(!(mask & (1<<u))) continue;

    for(let v=0; v<n; v++){

      if(mask & (1<<v)) continue;

      dp[mask | (1<<v)][v] = Math.min(dp[mask | (1<<v)][v], dp[mask][u] + dist[u][v]);

    }

  }

}


// Find minimum cost to return to start

let ans = Infinity;

for(let i=1;i<n;i++) ans = Math.min(ans, dp[(1<<n)-1][i]+dist[i][0]);

console.log("Minimum TSP cost:", ans);
```

Step 4: Applications

1. **Traveling Salesman Problem (TSP)**
2. **Assignment / Job allocation problems**
3. **Subset sum / combination optimization**
4. **DP problems with state constraints**

Step 5: Mini Exercises

1. Count subsets with sum divisible by k using bitmask
2. Solve small TSP problems ($n \leq 16$)
3. Job assignment: n workers, n jobs, minimum total cost
4. Maximum weight independent set in small graphs

Step 6: Common Mistakes

- Not understanding bit representation → wrong states
- Forgetting to initialize dp array correctly
- Incorrectly updating mask → overwriting previous results
- Using bitmask DP for n > 20 → exceeds feasible computation time

Step 7: Practice Plan for Bitmask DP

Day 1:

- Simple subset sum / state problems → 3–5 problems

Day 2:

- TSP and assignment problems → 3–5 problems

Day 3:

- Mixed combinatorial problems → 5–7 problems

Key Takeaways

- Bitmask DP solves **subset/state-based problems efficiently**
- Practice **state transitions** and bit manipulation
- Essential for **combinatorial optimization and advanced DP problems**
- Helps develop **problem-solving intuition for exponential state spaces**

Visit haas.dev for more Bitmask DP guides, problem sets, and interview preparation resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app