Week 27: Advanced Dynamic Programming Techniques

Topics: - DP on Bitmasks (Subset DP) - DP with Convex Hull Trick / Slope Optimization - DP with Knuth Optimization and Divide & Conquer Optimization - DP on Grids (2D DP with Transitions) - DP with State Compression and Memoization - Longest Increasing Subsequence Variants (LIS, LDS, k-LIS)

Weekly Tips: - Bitmask DP is useful for small sets, like TSP and subset problems. - Convex Hull Trick speeds up DP with linear recurrence and monotone slopes. - Knuth and Divide & Conquer optimizations reduce DP from O(n^3) to O(n^2) or O(n log n). - State compression helps reduce memory for large dimension DP. - Understand monotonicity and convexity properties for optimizations.

Problem 1: DP on Bitmask (Traveling Salesman) Link: CSES TSP **Difficulty:** Advanced

C++ Solution with Explanation Comments:

```
#include <bits/stdc++.h>
using namespace std;
const long long INF=1e18;
int n; vector<vector<long long>> cost;
vector<vector<long long>> dp;
long long tsp(int mask,int pos){
    if(mask==(1<<n)-1) return cost[pos][0];
    if(dp[mask][pos]!=-1) return dp[mask][pos];
    long long res=INF;
    for(int nxt=0;nxt<n;nxt++){
        if(!(mask&(1<<nxt))) res=min(res,cost[pos][nxt]+tsp(mask|(1<<nxt),nxt));
    }
    return dp[mask][pos]=res;
}
int main(){
    cin>>n; cost.assign(n,vector<long long>(n));
    for(int i=0;i<n;i++) for(int j=0;j<n;j++) cin>>cost[i][j];
    dp.assign(1<<n,vector<long long>(n,-1));
    cout<<tsp(1,0)<<endl;
}
```

Explanation Comments: - Bitmask represents visited nodes. - Recursively calculate minimum cost visiting remaining nodes. - Memoization avoids recomputation; feasible for n<=20.

Problem 2: DP with Convex Hull Trick Link: Codeforces Convex Hull Trick **Difficulty:** Advanced

C++ Solution with Explanation Comments:

```cpp
#include <bits/stdc++.h>
using namespace std;
struct Line{
    long long m,b;
    long long eval(long long x){ return m*x+b; }
};
vector<Line> hull;
bool bad(Line l1,Line l2,Line l3){
    return (l3.b-l1.b)*(l1.m-l2.m)<=(l2.b-l1.b)*(l1.m-l3.m);
}
void add(Line l){
    while(hull.size()>=2 && bad(hull[hull.size()-2],hull.back(),l))
hull.pop_back();
    hull.push_back(l);
}
long long query(long long x){
    int l=0,r=hull.size()-1;
    while(l<r){
        int m=(l+r)/2;
        if(hull[m].eval(x)>=hull[m+1].eval(x)) l=m+1; else r=m;
    }
    return hull[l].eval(x);
}
int main(){
    int n; cin>>n; vector<long long> dp(n+1),a(n+1),b(n+1);
    for(int i=1;i<=n;i++) cin>>a[i];
    for(int i=1;i<=n;i++) cin>>b[i];
    add({b[1],0});
    for(int i=1;i<=n;i++){
        dp[i]=query(a[i]);
        if(i<n) add({b[i+1],dp[i]});
    }
    cout<<dp[n]<<endl;
}
```

Explanation Comments: - Convex Hull Trick optimizes DP of form dp[i]=min(dp[j]+b[j]*a[i]). - Add lines to hull maintaining convexity. - Binary search query finds minimum efficiently. - Reduces DP from O(n^2) to O(n log n).

End of Week 27 - Advanced DP techniques allow efficient solutions for otherwise slow algorithms. - Practice Bitmask DP, Convex Hull Trick, and DP optimizations to excel in ACM-ICPC contests.