### Week 33: Advanced Geometry – Convex Hull Tricks and Line Container

**Topics:** - Convex Hull Trick (CHT) for DP Optimization - Dynamic CHT with Line Container - Li Chao Segment Tree for Line Queries - Applications: DP with Linear Recurrence, Cost Minimization, Slopes Optimization - Handling Monotone and Non-Monotone Queries - Geometric Interpretation of Lines and Envelopes

**Weekly Tips:** - Convex Hull Trick is effective for DP of form dp[i] = min(dp[j] + m[j]\*x[i] + b[j]). - Line Container uses multiset and slope comparison to maintain convex hull. - Li Chao Segment Tree allows dynamic insertion and query of lines over integer range. - Monotone queries allow O(1) or O(log n) per query; general queries require tree structures. - Visualizing the envelope of lines helps understand optimization.

**Problem 1: DP Optimization with CHT** **Link:** [Codeforces Example](https://codeforces.com/problemset/problem/321/C) **Difficulty:** Advanced

**C++ Solution with Explanation Comments:**

#include <bits/stdc++.h>  
using namespace std;  
struct Line{  
 long long m,b;  
 long long eval(long long x){ return m\*x+b; }  
};  
struct Hull{  
 vector<Line> lines;  
 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(lines.size()>=2 && bad(lines[lines.size()-2],lines.back(),l)) lines.pop\_back();  
 lines.push\_back(l);  
 }  
 long long query(long long x){  
 int l=0,r=lines.size()-1;  
 while(l<r){ int m=(l+r)/2; if(lines[m].eval(x)>=lines[m+1].eval(x)) l=m+1; else r=m; }  
 return lines[l].eval(x);  
 }  
};  
int main(){  
 int n; cin>>n;  
 vector<long long> a(n+1),dp(n+1);  
 for(int i=1;i<=n;i++) cin>>a[i];  
 Hull hull;  
 dp[1]=0; hull.add({a[1],dp[1]});  
 for(int i=2;i<=n;i++){  
 dp[i]=hull.query(i); // example usage  
 hull.add({a[i],dp[i]});  
 }  
 cout<<dp[n]<<endl;  
}

**Explanation Comments:** - Each DP state represented as a line y = m\*x + b. - Add lines to hull maintaining convexity. - Query minimum value for given x efficiently. - Reduces naive O(n^2) DP to O(n log n) or O(n).

**Problem 2: Li Chao Segment Tree** **Link:** [CP-Algorithms Li Chao Tree](https://cp-algorithms.com/geometry/segment_tree_li_chao.html) **Difficulty:** Advanced

**C++ Solution with Explanation Comments:**

#include <bits/stdc++.h>  
using namespace std;  
struct Line{ long long m,b; long long eval(long long x){ return m\*x+b; } };  
struct LiChaoNode{ Line line; LiChaoNode \*l=nullptr,\*r=nullptr; };  
long long minX=-1e9,maxX=1e9;  
void insert(LiChaoNode\* &node,Line newLine,long long l=minX,long long r=maxX){  
 if(!node){ node=new LiChaoNode{newLine}; return; }  
 long long m=(l+r)/2;  
 bool left=newLine.eval(l)<node->line.eval(l);  
 bool mid=newLine.eval(m)<node->line.eval(m);  
 if(mid) swap(node->line,newLine);  
 if(r-l==1) return;  
 if(left!=mid) insert(node->l,newLine,l,m); else insert(node->r,newLine,m,r);  
}  
long long query(LiChaoNode\* node,long long x,long long l=minX,long long r=maxX){  
 if(!node) return LLONG\_MAX;  
 long long m=(l+r)/2,res=node->line.eval(x);  
 if(r-l==1) return res;  
 if(x<m) return min(res,query(node->l,x,l,m));  
 else return min(res,query(node->r,x,m,r));  
}  
int main(){  
 LiChaoNode\* root=nullptr;  
 insert(root,{2,3}); // example line y=2x+3  
 insert(root,{1,5}); // example line y=x+5  
 cout<<query(root,10)<<endl; // query x=10  
}

**Explanation Comments:** - Li Chao Tree supports dynamic insertion and query over range. - Useful when lines can be added in any order. - Efficient for O(log N) per operation. - Can handle DP or geometric optimization with arbitrary x values.

**End of Week 33** - Advanced convex hull tricks and line container techniques optimize DP and geometric problems. - Practice both static and dynamic line queries for ACM-ICPC contests.