code jam
hello, world!

Practice Mode

Round 1C 2009

Contest Analysis

Questions asked 1

Submissions

All Your Base

8pt	Not attempted **2176/2473 users** correct (88%)
15pt	Not attempted **1441/2203 users** correct (65%)

Center of Mass

10pt	Not attempted **823/1428 users** correct (58%)
17pt	Not attempted **737/913 users** correct (81%)

Bribe the Prisoners

15pt	Not attempted **1061/1579 users** correct (67%)
35pt	Not attempted **302/735 users** correct (41%)

Top Scores

tikitikirevenge	100
Progbeat	100
Zeroline	100
maojm	100
WSX	100
Onufry	100
Imba	100
ZhukovDmitry	100
Al.Cash	100

Contest Analysis

Overview | Problem A | Problem B | Problem C

This problem is solved by dynamic programming. For each pair of cells a ≤ b, we want to compute dp[a][b], the best answer if we only have prisoners in cells from a to b, inclusive. Once we decide the location of c, the first prisoner between a and b to be released, we face the smaller sub-problems dp[a][c-1] and dp[c+1][b]. The final answer we want is dp[1][P].

It is clear we only need to solve those dp[a][b]'s where both a and b are either 1, P, or adjacent to a prisoner to be released. Thus the number of sub-problems we need to solve is just $O(Q^2)$.

Here is the annotated judge's solution.

```
int p[200]; // prisoners to be released.
map<pair<int, int>, int> dp;

// Finds the minimum amount of gold needed,
// if we only consider the cells from a to b,
int Solve(int a, int b) {
  // First, look up the cache to see if the
  // result is computed before.
  pair<int, int> pr(a, b);
  if(mp.find(pr) != mp.end()) return mp[pr];

  // Start the computation.
  int r = 0;
  for(int i=0; i<Q; i++) {
    if(p[i] >= a && p[i] <= b) {
      int tmp = (b-a) + Solve(a, p[i]-1) + So
      if (!r || tmp<r) r=tmp;
    }
  }
  mp[pr]=r;
  return r;
}
```

Ostap	100