Editorials

for

Insomnia'13

<15th March 2013>

1. Visit to biolab

Topic: Convex Hull
Problem Setter: Shubham Kansal
Problem Tester: Abhay Prakash

The problem mentions that the bacteria has a very less radius i.e. they will cover a finite area.
If we see the pattern of growth, at each reproduction a new bacteria is produced in mid of all possible pair. Thus, after a long time, the whole area inside the Convex Hull formed by initial positions of bacteria, will be covered.

Thus the problem reduces to finding the area of the convex hull formed by the initial coordinates of the bacteria.

Method to get convex hull:
There are several algorithms mentioned here<http://en.wikipedia.org/wiki/Convex_hull_algorithms> along with the complexity.

According to the given constraints O(Nlog(N)) or O(Nlog(H)) will pass in the given time limit. So we can use Graham Scan or Chan's Algorithm.

Method to get the Area of Hull:
After getting the hull, we can use idea of 'cross product' to get the area of Polygon. Its complexity is O(H).

2. Running late for office

Topic: Dynamic Programming & Counting
Problem Setter: Sharat
Tester: Surya Kiran

Given a rectangular grid of lattice points, our task is to find the number of ways in which we can go from the point (0,0) to (X,Y) using X+Y steps and never crossing any junction which is blocked. A crucial thing here was to observe that the number of ways in which we can go from (x1,y1) to (x2,y2) is (y2-y1+x2-x1 choose y2-y1). Call this value W(P1,P2). If we sort the blocked points in the order of increasing x and break the ties in the order of increasing y then we can actually find the number of ways of reaching (X,Y) as follows: Store the value of number of ways in which starting from the point (0,0) we can reach every blocked junction and the destination. Then visit the first blocked junction and see how it effects the number of ways to reach the points under consideration. It is fairly simple to see that if the value stored currently at blocked[1] is V then we need to subtract V*W(blocked[1],blocked[i]) from the value stored at blocked[i] if blocked[i].x >= blocked[1].x and blocked[i].y >= blocked[1].y Continuing with this procedure we can find all the values in O(N^2).

3. Let's play a game

Topic: Grundy Numbers
Problem Setter: Abhay Prakash
Problem Tester: Sharat & Surya Kiran

Conventions:
V : Maximum value a cell can have.
M : Number of Columns
N : Number of Rows

For the given matrix the grundy number is calculated as described below. If it comes equal to 0 then the second player will win else the first player will win, given that both the players play optimally.

Method to get Grundy Number of the given Matrix:
The game proceeds independently for each row so we just need to find the grundy number for each row and in the end just xor them.

For finding grundy for a single row, take a row with N nonzero elements(a1 a2 .. aN). Initially numbers in any of the cell is between 1 and 9 inclusive. Note that at any turn, at most one of the element, either at front or back in the row can be made zero. So at any time all the cell in between must be having non zero values.

Let grundy[i][j][x][y] denote the grundy number of the configuration (i,j,x,y) where we are left with nonzero elements between ith and jth element(j >= i). x is the value at ith column and y is the value at jth column.

We are interested in knowing the transitions possible from this state. They can be represented as follows :
(i,j,x,y) -> (i,j,u,y) for all u such that 0 < u < x
(i,j,x,y) -> (i,j,x,v) for all v such that 0 < v < y
(i,j,x,y) -> (i+1,j,a[i+1],y) if we remove the number completely from ith place
(i,j,x,y) -> (i,j-1,x,a[j-1]) if we remove the number completely from jth place

Also we have the following knowledge about the grundy numbers:
grundy[i][i][x][x] = x for all i between 1 and M, and x between 0 and a[i].

Complexity Analysis:
The number of transitions for any state is O(V). Total number of states that it can reach is O(M^2*V^2) so for each row the complexity will be O(M^2*V^3).

Finally repeating the procedure for all the rows and taking the xor of grundy values for each row gives the overall grundy value for the whole game.

Hence overall complexity will be O(N*M^2*V^3) which fits well in time for the given constraints.

4. Let's destroy them

Topic: Data Structures
Problem Setter: Surya Kiran
Problem Testers: Harsh and Sharat

This problem can be viewed as a Range-Query Problem which can be handled using Segment Trees or Binary Indexed Trees.
Consider an array A[1..N] which stores the damages done to the cities[1..N]. Let us consider a request of type 0. The damages on the cities are of the form " … 0 0 0 0 0 1 2 3 … (D-2) (D-1) D (D-1) (D-2) … 3 2 1 0 0 0 0 … ".
If you consider the differences of the damages caused to a city with the immediate predecessor then the difference array looks like " … 0 0 0 0 0 1 1 1 … 1 1 1 -1 -1 … -1 -1 -1 -1 0 0 0 … ". Let us say that these are stored in array B[1..N]. ".
If you consider the differences of the values stored in B[1…N] with their immediate predecessor then the difference array looks like " … 0 0 0 0 0 1 0 0 … 0 0 0 -2 0 … 0 0 0 0 1 0 0 0 … ". Let us say that these are stored in array C[1…N].
Now if we look for every query only 3 of the array values of C[1…N] are changing. So, we will be creating a binary indexed Tree for C[1...N] so that we will be updating it in O(logN) time for each query.
Let us now consider the query of type 1. The value of A[i] will be the sum of values of B[j] for all 1 <= j <= i. This can be proved easily by writing B[j] = A[j] – A[j-1] for each j and when we sum these we get A[i] – A[i-1] + A[i-1] – A[i-2] + A[i-2] - A[i-3] ….A[1]-A[0]. A[0] will be zero every time so the adjacent terms cancel each other and we get A[i] = sum of (B[j] for all 1 <= j <= i). Similarly as C[1..N] is the difference array of B[1…N] B[j] = sum of (C[k] for all 1 <= k <= j).
Therefore A[i] = sum of (sum of (C[k] for all 1 <= k <= j) for all 1 <= j <= i). It can be reduced into the form of A[i] = sum of ((i+1)*(sum of (C[k] for all 1 <= k <= i) – sum of ((k*C[k]) for all 1 <= k <= i). As we can see we can obtain sum of (C[k] for all 1 <= k <= i) in O(logN) time by using Binary Indexed Tree created on C[1…N].But to obtain sum of ((k*C[k]) for all 1 <= k <= i) we need to create one more array D[1…N] such that D[k] = k*C[k]. Now whenever we update C[k] we need to update D[k] also. So we create Binary Indexed Tree for D[1…N] also. Now our updating time is still order of O(logN) time.
To obtain sum of (A[i] for all x <= i <= y) we need to reduce the equation we get when we convert all the values in terms of C[k]. When we reduce it we get a quadratic equation of the form A[i] = (sum of (C[k]*k*k + C[k]*k*(2*i+3) + C[k]*(i*i+3*i+2) for all 1 <= k <= i) / 2). So we need one more array E[1..N] to store C[k]*k*k. We create Binary Indexed Tree for E[1…N]. Similar to D[1..N], E[1..N] should also be updated whenever C[k] is being updated. So our updating time is still of the order O(logN).
All the sum of (C[k]*k*k*Const1) , sum of (C[k]*k*Const2), sum of (C[k]*Const3) can be obtained from respective BITs of E[1..N], D[1…N], C[1….N] in the order of O(logN) time thereby the complexity of our solution is O(Q log N).

5. Lab

Topic: LCA(least Common Ancestor)
Problem Setter: Harsh Jhamtani
Problem Tester: Sharat Ibrahimpur

Choose a root and run a dfs assigning levels and cumulative lengths of wire from the choosen root. Then we can use dynamic programming to get the table dp[N][lg N], where dp[i][j] denotes the 2^j th ancestor of i. For more details refer to http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor#Another easy solution in O(N logN, O(logN)

For the query x,y,z, consider:
Let x1=lca(x,z)
y1=lca(y,z)
u=lca(x,y)

Condition for failure:

either : (x1==z && lca(z,u)==u)
OR (y1==z && lca(z,u)==u)

Also, Length=len_from_root[x]+len_from_root[y]-len_from_root[u]

Time complexity: O(nlgn+qlgn)

6. Game

Topic: Number-Theory, Dynamic Programming
Problem Setter: Harsh Jhamtani
Problem Tester: Sharat Ibrahimpur

For a given n, it is important to observe that most of the states don't play any role. We can find all prime factors of a given n in O(sqrt(n)). Then using them generate all the factors for the given n, which will be O(lgn). Let their no. be x. x=O((lg n)) .Let the x states are stored in states[max_states].

Then for x states we can have the result in O(x^2) using dp.
Consider dp[N][2]. Let start_state=index of k in states[]. If k is not in states, it is not possible to reach nth state

dp[i][0]=minimum cost to reach i in odd number of steps
dp[i][1]=minimum cost to reach i in even number of steps
Initially set all of dp[i][j] to be INF, where INF denotes a vey large value.
Set dp[state_index(k)][0]=0
Then for all j s.t j<i and i%j==0, then dp[i][0]=min(dp[i][0],dp[j][1]+dist(i,j)). Similar for dp[i][1].
Now if dp[n][1] is INF, then it is not possible to reach n from k in odd number of steps. Otherwise, the answer is dp[n][1]

Time complexity: O(sqrt(n)+(lgn)^2)

7. Find The Largest Rectangle

Topic: Computational Geometry
Problem Setter: Shagun Sodhani
Problem Tester: Abhay Prakash

Conventions:
area_max : Maximum area.
n : Number of elements

The basic idea behind the question was to find the largest non empty rectangle with in a large rectangle containing various points such that the sides of the chosen rectangle are parallel to given rectangle. We first sort all the given points in ascending order with x being compared first and y compared in case x is equal. This takes O(n)long(n) time.We sweep the rectangle with horizontal and vertical lines , maintaining a list of those intervals, where the lines intersect each other.Then given a rectangle of some area 'area_max' we try to find the next largest rectangle by moving horizontally and vertically in the same loop. Starting with any point we cover all the points greater than the chosen point. This takes O(n^2) time. Also we swap height and width to account for the difference in height and width of the rectangle. The loop always runs from smaller point to larger points so we run the outer loop 4 times rotating the rectangle by 90 each time. The total time turns out to be n*n+n*log(n). The complexity of the code is O(n^2).

8. Digo wants burgers

Topic: DP and Counting
Problem Setter: Shubham Kansal
Tester: Sharat

To obtain an amount of N using coins of value V[i] and which can be used at most K[i] times we just to evaluate the coefficient of x^N in product(1 + x^V[i] + x^2V[i] + x^3V[i] + … + x^(K[i]*V[i])) over i from 1 to N.
As we had multiple queries and maximum value of N was 2000 we just had to preprocess the required answer for each N from 0 to 2000 and then print the answers in O(1). This can be done by considering a polynomial f(x) = 1 initially and multiplying it with the polynomials mentioned above and keep only the powers of x less than or equal to 2000.
Expected complexity: O(N*N log N) where N is at max 2000

9. Count them if you can

Topic: Maths
Problem Setter: Surya Kiran
Tester: Shubham Kansal

We expected this problem to be the easiest among the whole problem set but it was surprising to see that it was not solved by many. If you draw the configurations using a pen and paper then it is fairly obvious to note that at any given point of time, either a cell has no bacteria or the number present in it can be expressed as a product of binomial coefficients.
For a given x, y, z the answer is same for abs(x), abs(y), z. So we will just have to deal with nonnegative x and y. If parity of (x+y) is equal to parity of z then the answer is nonzero otherwise it is zero. When the answer is nonzero it can be expressed as (z-1 choose (z-1-(x+y))/2) * (z-1 choose (z+x-1-y)/2). The value of these binomial coefficients can be obtained in O(1) by pre-calculating the factorials and their inverses modulo 10^8+2013 upto 2 million. This can be achieved in O(N) time.

10. Catch me if you can

Topic: Precomputation, Graphs.
Problem Setter: Sunit Kumar Singh
Problem Tester:

Due to the third move possible, it can easily be observed that putting a dollar bundle at place (i) is equivalent to putting it at place (i+n*3), where n is integer. This means we don't have to wonder about the exact arrangement but the total number of bundles at place (1+n*3), place(2+n*3) and place(n*3). This reduces to a total number of states to be 35*34*34 states. These states can be precomputed by doing a dfs over a 3d array.

11. Can you sort them?

Topic: Ad-hoc, Bitwise operations

Problem Setter: Sharat
Problem Testers: Harsh Jhamtani, Surya Kiran

The basic idea is to traverse the numbers bit by bit starting from the most significant bit. There can be 4 cases for the ith bit position
1) all 0s i.e. all numbers have 0s in there ith bit positions: xoring this bit position will not make any difference.
2) all 1s: xoring this bit position will not make any difference
3) first some 0s then remaining are 1s: Again xoring this bit will not help us. But note that in this case let numbers from index=1 to index=x have 0 as their ith bit, while remaining i.e. from index=x+1 to index=n have 1 as their ith bit. Now no matter whatever happens at bit positions lesser significant than i, the numbers at indexes 1 to x will always be smaller than the numbers indexes from x+1 to n. Note that this info is important since for successive iterative steps for lesser significant bits, you need to consider the above mentioned 2 groups independently since each element of the 1st group is already known to be smaller than each element of the 2nd group.
4) first some 1s then remaining are 0s: We have to xor this bit. Make the ith bit of answer as set.
5) Any other combination would mean the answer is not possible i.e. it is not possible to have an answer for the given sequence of numbers.

Time Complexity: O(n*60) [Here max. no. of bits of a number is less than 60]

12. Can you see them?

Topic: Sorting, Memoization.
Promblem Setter : Shagun Sodhani
Problem Tester : Harsh Jhamtani and Surya Kiran

This problem is based on sorting and DP. Consider the problem of solving for two ships. First we sort the ships with respect to distance between ship and first light house and then with respect to distance between ship and second light house. We have to move along the array and include the ships one by one in order into the first light house. When we include i'th ship all the ships with index 1 to i-1 will be inside the radius of first light house. We need to find the maximum of distances of ships with index j > i with second light house. This can be achieved in O(N) time with brute and in O(1) time with memorization.So minimum of all these values are found in O(NlogN + N) time. Now for 3 lighthouses we will sort the ships w.r.t first light house and then equal ones are sorted w.r.t second light house and then equal ones w.r.t third light house. We will find minimum power by same method used for two light houses. But it takes O(N^3) by brute implementation. But with some optimizations using STL SET it can be achieved in O(N^2 + NlogN) time.There can be solutions with Complexity of O(N^2*logN) but time limit was set so tightly that all solutions other than O(N^2) are timed out.

13. Bribe The President

Topic: Dijkstra algorithm for directed graphs
Problem Setter: Harsh Jhamtani
Problem Tester: Sharat Ibrahimpur

Its basically a modified dijkstra question. We have 1 to n states in board. Lets model it as an undirected graph. There would be n nodes. An edge is there between 2 nodes a and b iff gcd(a,b)=1 i.e. co-prime. For some n, phi(n)=O(n), so there will be a max of O(n^2) edges.

Now starting node is k. We know the cost of edges as (a+b)%m for any edge (a,b). We can run a Dijkstra from k, and find min. cost for a path to n. But since Digo is the first as well as the last one to get a turn, some minor changes to dijsktra will have to be applied. Let the original graph be G. We will create another graph G2. For this, we add duplicate nodes and edges. i.e. for every node u in G, create nodes u1 and u2 in G2. Now for every edges (u,v) in original graph(G), add (u1,v2) and (u2,v1) in G2. This facilitates that the two players will alternate turns. Make dist[i1]=INF,dist[i2]=INF for all 1<=i1<=n, 1<=i2<=n. Now make dist[k1]=0. Run the dijkstra's algorithm in G2. At the end find value of dist[n2].

Time complexity: O(n^2 lgn)
Space complexity: O(n^2)

14. Baywatch

Topic: Min-Cost Bipartite Matching, Ternary Search.
Problem Setter: Surya Kiran
Testers: Harsh and Sharat

An experienced programmer would easily be able to reformulate the given problem as a min-cost bipartite matching or the assignment problem. We create a bipartite graph with N hotels each on each of the partitions. As every tourist wants to change his/her hotel so we assign a cost of infinity to the edge joining hotel i to i. Now we need to assign the cost of the edge connecting hotel i to hotel j. This is precisely the Euclidean distance the tourist travels in order to go from hotel i to hotel j by visiting at least one coastline. As we need to minimize the costs we should find the shortest possible route to achieve this. Here is the most interesting part, given a pair of points P1 and P2 and a line segment L then the function cost(P1,P2,P) = dist(P1,P) + dist(P,P2) for P on L is a convex function. So to find the minima we can use ternary search. After finding the minima we can assign this value to the edge connecting i and j. After this you just need to use the Hungarian Assignment algorithm or any other algorithm which can find the min-cost matching in O(N^3) to squeeze through the time limit. The expected time complexity was O(N^3 + N^2 * M * constant for ternary search).

The test files had many corner cases and cases which would time out a O(N^4) solution.
Anton Lunyov was stuck with our second test file having the following input:

```
4
0 0 3 0
3 0 3 3
3 3 0 3
0 3 0 0
4
0 1
0 2
3 1
3 2
```

For this file Anton's answer was 12 but we can clearly see that the answer is 4 which can be achieved by letting the 1st and 2nd tourists swapping their positions and similarly for 3rd and 4th. He missed to take into consideration that the hotels may actually lie on the coastline.