ACM ICPC Amritapuri Regional 2016
Problem solving session

D - ATM Queue (by Praveen)

1. **Strictly** adhere to the input/output format described in the problem
2. How online judges evaluate your problems ?
3. How to write and and check your program ?
4. The problem - Observation:
   a. Front - [p_1 p_2, p_3, …,$\mathbf{p_k}$, p_{k+1} ….p_n] - End of the queue
   b. Front - [(p_1 p_2, p_3), (p_4, p_5, p_6), …. (p_{n-2}, p_{n-1}, p_n)]
5. If k is divisible by 3 then "No" otherwise "Yes"
6. Complexity
7. T <= 100, N <= 100, K < N.
8. Approx Complexity : T * 2 = O(T).

H - Netcoin Verification (by Praveen)

Simpler Version : Permute an array such that total verification is maximum

Solution : Sort the array is descending order. Example : [5, 1, 2, 3, 4] => [5, 4, 3, 2, 1] : Ans = (5) + (5+4) + (5+4+3) + (5+4+3+2) + (5+4+3+2+1)

Observation we can not get the largest element first! So we can not achieve the above. Infact, the largest element is always going to be at the end ! Ex : [5, 1, 2, 3, 4] => [?, ?, ?, ?, 5] or [{1, 2, 3, 4}, 5]. We note that Actual Ans <= Ans ([4, 3, 2, 1, 5]).

Claim : Actual Ans = Ans ([4, 3, 2, 1, 5])

Can we always achieve that ??

C - Influence on Social media (by Prof. Chellam)

Key observations:

- For prime factorization $p1^{a1} * p2^{a2} * p3^{a3} \dots *pn^{an}$, # of divisors equals $(a1 + 1) * (a2 + 1) \dots.. * (an + 1)$
- From above, the number must be of form p^x where p and x + 1 are primes.
- For p^2 check if sqrt(n) is prime.
- For bigger powers check for primes $<= 10^3$.

I - Mancunian Hoards Black Money (by Satyaki)

- Greedy algorithm
- Subset sum of superincreasing sequence, $\mathbf{S_n = \Sigma a_i, i<=n \; ; a_n > S_{n-1}}$
- Avoid overflow - common mistake
- Complexity - O(n*logn)
- Main Idea : If target ≥ a[i + 1], you must take that item since you can't obtain that sum using all of the first i items.
- .

B - A Historic Discussion (by Misha Chorney)

-> From samples and small examples, it looks like answer is n / 2?

- N = 2, 1 game, [1] vs [2].
- N = 4, 2 game, [1 2] vs [3 4]   and [1 3] vs [2 4]
- N = 6, 3 games,
  - [1 2 3] vs [4 5 6]
  - [1 2 4] vs [3 5 6]
  - [1 4 5] vs [2 3 6]
- N = 8, 3 games? How ? (its construction we will see it later :))
- Looks like answer is something of order of O(log n)?
- Intuition into it?

Construction

- Consider the numbers in base two. Assume indexing as zero.
- How to solve when N is power of 2?, i.e. $N = 2^K$
- Write numbers in the binary.
- There are K bits of numbers.
- Single game per bit.
- You can notice that for each bit, there are N/2 zeros and N/2 ones,
- Proof why will everybody play a game in this way?

What if $N \neq 2^K$

- Can we re-number the numbers in such a way that for each bit, there are equal number of zeros and ones?
- Yes, how?

lower bound of number of games?

- T(n) = min(T(a), T(b)) + 1, where a + b = n.
- You can prove that it will be O(ceil(log n)).
- This case will be achieved when a and b are as close to n/2 as possible.

- Use this idea to write a recursive solution to find the games !!

J - Bob vs ATM (by Balajiganapathi)

- Given a string of valid bracket sequence, a valid move is erase a valid bracket substring which are not formed by concatenating 2
- (())() -> ATM wins by taking out (***()***)() -> ()() and Bob loses
- () - nim pile with 1 stone, (()) - nim pile with 2 stones - nimber
- Apply Sprague-Grundy Theorem for finding combined nimber
- Algorithm:
- A has *n, B has *m
- <empty> - *0
- (A) -> *(n + 1)
- AB -> *n xor *m

One liner by Kevin

```
void solve() {
    scanf("%s", &s);
    a[t] = 0;
    for (char *c = s; *c; c++) {
        a[t] ^= *c == '(' ? a[++t] : ++a[t--];
    }
    puts(a[t] ? "ATM" : "Bob");
}
```

G - Havala Arrests (by Arjun Arul)

- Greedy algorithm
- Process leaves turn by turn
- Add leaf to set iff it belongs to some havala and its parent doesn’t
- Complexity - O(n*logn)

E - Black Money Island (by Animesh)

- Statement: Given graph G whose edge weights are A or A + 1, find a spanning tree for given X.
- Reduced to 0/1 edge weights, reduce X by A * (n - 1)
- Answer is YES iff :
- Weight of Minimum Spanning Tree <= Query (Denoted by X) <= Weight of Maximum Spanning Tree.
- T1(Min ST), T2(Max ST), E = One(T2) - One(T1)
- Take T1, Add edges from E one by one till we get X. While adding edges, keep removing an edge that is not in T2 from the simple cycle formed.
    - Binary search - N * log(N) * alpha(N) per query ---> **TLE**
- Construct a maxST having all edges in One(T1) : This is **always** possible!
- Now whenever we add e from E into T1, the cycle formed will have a 0 edge.

E Continued.

These observations give us a simple method to construct the spanning tree :-

Step 1) Find One(T1) (There can be multiple possibilities, any one is fine). Intuitively, One(T1) is a set of **necessary** one-edges.

Step 2) Simulate Kruskal's again, but this time start with edges in One(T1). Then, add more weight 1 edges (ensuring that you don't create a cycle at each step) until you've reached the desired weight X.

Step 3) Finally, finish up the tree by adding 0 edges, if needed.

Complexity : O(n * alpha(n)) per query. Can also be done in O(n) per query, but that’s not required for this problem.

F - Notes and GPS Chips (by Akash)

Key observations:

- For functions like $F(x) = f(0) + f(1)*x + f(2)*x^2 + f(3)*x^3 + \dots$, multiplication gives discrete convolution.
- Generating function for square shaped chip:
  - $F(x) = x + 2^2x^2 + 3^2x^3 + 4^2x^4 + \dots..$
- Generating function for hexagonal shaped chip:
  - $G(x) = 6x + 18x^2 + 36x^3 + \dots..$
- In $F^{\mathbf{A}}(x) * G^{\mathbf{B}}(x)$, coefficient of $x^{\mathbf{i}}$ denotes the required summation.
- Closed form: $6^{\mathbf{B}}\ x^{\mathbf{A+B}}\ (1 + x)^{\mathbf{A}}\ (1 - x)^{\mathbf{-3A - 3B}}$
- Challenge problem: Solve for the case when hexagonal chips don't have a hole in center.

A - Tim and BSTs (by Kevin)

- For each node **nd** do:
  - For l = 0 to nd.left.Size
    - For r = 0 to nd.right.Size
      - nd.ans[someVal] += (Something * someOtherThing) (nd.left.ans'[l], nd.right.ans'[r])

Complexity ?

A contd.

- DP[i][j] = number of ways to assign the subtree rooted at i from values in the range [1, s[i]] (where s[i] is size of subtree of i) such that label of i^{th} node is j
- Consider the intervals [1..j-1] and [j+1….s[i]]
- stayL = number of values from first range which lie in left subtree of i
- moveL = number of values from first range which lie in right subtree of i
- stayR = number of values from first range which lie in right subtree of i
- moveR = number of values from first range which lie in left subtree of i
- stayL + moveL = j-1, stayR + moveR = s[i]-j
- stayL + moveR = s[L[i]], stayR + moveL = s[R[i]]

A contd.

- Restriction - label[L[i]] has to lie in [1...j-1] and label[R[i]] has to lie in [j+1….s[i]]
- l = rank of label[L[i]] in the subtree rooted at L[i]
- r = rank of label[R[i]] in the subtree rooted at R[i]
- C(j-1, stayL) * C(s[i]-j, stayR) * DP[L[i]][l] * DP[R[i]][moveL+r]

K - Paint for Vernon (by Kevin)

Key observations:

- Maximum beauty possible for r x c grid is, $r(c-2) + (r-2)c + 2(r-1)(c-1)$.
- 15874 for 64 x 64.
- Using the above formula, find r and c such that beauty of that grid <= N. The use rest of the rows to get exact N. This should work for MAXN - 128.
- Only MAXN - 1 is not possible. MAXN - 128 to MAXN except MAXN - 1 can be formed by changing elements of last 2 rows of MAXN matrix.