Problema A. Navalha Batal

A minha solução é basicamente criar uma "lista de adjacência"para cada linha/coluna com base nas Q coordenadas de entrada e então ordená-las. Após isso, percorre todas as listas (a soma dos elementos delas será no máximo $2Q$) e descobre todos os intervalos entre duas peças consecutivas. Como a lista estará ordenada, para descobrir o tamanho de cada intervalo basta fazer `lista[atual][i]-lista[atual][i-1]` para todo i de 1 até o tamanho da `lista[atual]`. Para descobrir a quantidade de peças que é possível colocar em um intervalo de tamanho X, basta fazer o somatório de $[1, X]$, podendo ser calculado em tempo constante com a fórmula $(X * (X + 1))/2$. Acumule todas essas quantidades tirando módulo $10^9 + 7$ e exiba como resposta.

Atentar para não contar peças de tamanho 1x1 duas vezes. Atentar para não esquecer de contar os intervalos entre as bordas e a primeira peça.

A complexidade, por conta da ordenação, é $A \log A$, sendo $A = 2Q$.

Problema B. Podando Galhos

A solução é simples. Quando um vértice se desprende do pai, basta fazer uma DFS na sua subárvore marcando o momento em que cada vértice caiu. O único truque necessário é evitar visitar vértices que já caíram. Assim, a complexidade final fica linear.

Problema C. Caça às Flores

Criar uma matriz NxMxT para fazer a soma acumulada de cada um dos 10 bits possíveis. Percorre-se a matriz fixando o vértice superior esquerdo do quadrado e então tento encontrar o menor quadrado que contenha todos os T bits ao menos uma vez. A ideia para encontrar o quadrado é através de uma busca binária chutando o vértice inferior direito.

A complexidade final fica: $O(NxMxTxLog(min(N, M))) \approx 2,2x10^7$

Problema D. Prêmios da Mineira

Problema E. Jantar das Capivaras

Seja G o grafo correspondente às alianças entre as capivaras. Se G possui algum vértice de grau um, coloque ele de um lado e os demais do outro. Se isso não ocorre, seja C um ciclo de G, que pode ser encontrado em tempo linear com uma busca em profundidade. Vamos agora encontrar um ciclo menor que intersecta C um caminho de C. Isso ocorre quando existe algum vertice em $G \setminus C$ com dois vizinhos em C. Como G tem grau máximo três, cada vértice de C possui, no máximo, um vizinho fora de C. Ou seja, encontrar um v com $d_C(v) \geq 2$ pode ser feito em tempo linear. Ainda mais, v define dois novos cíclos, sendo que um deles tem, no máximo, metade dos vértices de C.

Repetindo esse processo, temos 3 casos:

1. Não há mais nenhum v com dois vizinhos em C. Isso implica que a partição $(C, G \setminus C)$ é uma solução para o problema.
2. Existe v e $|C| = 4$. Esse caso gera uma bipartido completo $K_{2,3}$ que é indivisível, que chamamos de Q. Agora, note que os vertices do lado de tamanho três desse bipartido são os únicos com algum vizinho fora do bipartido. Se $Q = G$, responda não. Se nenhum v fora de Q tem 2 vizinhos, $(Q, G \setminus Q)$ é uma partição válida. Se não, defina $Q = Q \cup \{v\}$ e repita a análise. Porém, essa repetição só pode ser feita mais duas vezes, pois, quando o número de vertices de Q com grau dois em Q for um, é impossível não poder particionar o grafo em $(Q, G \setminus Q)$.
3. Existe v e $|C| = 3$. Fazemos uma análise idêntica ao caso do bipartido completo, porém agora temos que o triâgulo como base.

A obesrvação crucial é que a operação de redução dos cíclos é bastante eficiente, pois, a cada iteração, reduzimos em pelo menos metade o número de vértices que temos de verificar. A complexidade final é, portante, $\mathcal{O}(n \log n)$.

Problema F. Quem é o mentiroso?

O problema apresenta um grafo $G(V, E)$ onde todos os vértices $v \in V$ são os participantes, e as arestas $e(v_i, v_j)\ \forall\ v_i, v_j \in V$ representam que a pessoa v_i sorteou a pessoa v_j. É garantido que os vértices do grafo tem grau de entrada e saída igual a 1 ($g_e(v) = g_s(v) = 1$), porque cada pessoa tira apenas uma outra e é tirada apenas uma vez. É garantido também que não existem loops no grafo (uma pessoa não tira a si próprio no Amigo Oculto). Resumindo: o grafo G do Amigo Oculto é um grafo de ciclos.

Seja v_i o mentiroso, e v_j a pessoa que o mentiroso sorteou e v_k a pessoa que o mentiroso diz ter sorteado. O mentiroso então remove de G a aresta $e(v_i, v_j)$ e adiciona a aresta $e(v_i, v_k)$ tal que $g_e(v_j) = 0$ e $g_e(v_k) = 2$. Assim, para novamente corrigir o grafo e manter a propriedade $g_e(v) = g_s(v) = 1$, temos que deslocar umas das duas arestas que incidem em v_k para v_j. Logo, os únicos possíveis mentirosos são sempre os dois que dizem ter tirado a mesma pessoa. Exceto quando o outro vértice de entrada de v_k é o próprio v_j ou quando v_i é v_k, então o mentiroso é garantido ser somente v_i.

Problema G. Brincando com a formiga

Podemos calcular facilmente quantos segundos a formiga leva para subir e descer o brinquedo uma vez. Para subir, há $2^(K+1)$ segmentos subindo e $2^(K+1)$ segmentos horizontais. Portanto, ela leva $2^(K+2)+2^(K+1) = 3*2^(K+1)$ segundos para subir. Para descer, ela leva $2^K+2^(K+1) = 3*2^K$. Para subir e descer, portanto, ela leva $9 * 2^K$ segundos. Podemos primeiro reduzir n módulo este número sem alterar a solução.

Depois, podemos calcular se a formiga estará subindo ou descendo facilmente. Se $N <= 3*2^(K+1)$, ela estará subindo no segundo N. Caso contrário, está descendo. Vamos assumir que ela está subindo; a solução para a formiga descendo é similar.

Se ela está subindo, podemos facilmente calcular se ela está na primeira ou na segunda metade do trajeto. Se ela está na primeira metade, ela está percorrendo um brinquedo de ordem $K-1$. Basta chamarmos recursivamente então a nossa solução com uma ordem a menos no brinquedo. Se ela está na segunda metade, ela também está em uma parte de ordem $K-1$. Também resolvemos recursivamente, subtraindo o tempo que ela demora para percorrer a primeira metade, mas temos que adicionar um deslocamento em x e em y na resposta para considerar que ela está na segunda metade.

Com isso, o custo da solução fica $O(K)$.

Problema H. Comunicação Trocada

Problema I. Correndo em Círculos

É fácil ver que a solução ótima pode sempre caminhar entre pontos que estão na extremidade do círculo maior (dá para transformar qualquer solução numa desse tipo sem aumentar o número de lados). Além disso, o ótimo é fazer com que cada troca de direção tenha o maior ângulo possível. Basta calcular esse ângulo e pegar o teto da divisão de 2*PI por ele. Para calcular o ângulo, basta desenhar no papel as duas circunferências, centradas no ponto O, e de raios a e b. Sejam P_1 e P_2 dois pontos sobre a circunferência de raio b, de tal forma que o segmento $P1-P2$ tangencie a circunferência de raio a (assim o arco $P1-P2$ é o maior possível) num ponto que chamamos de S. Observe o triângulo $OSP1$. Ele é retângulo, e um cateto tem tamanho a e a hipotenusa tem tamanho b. O seno do ângulo $OP1S$ é a/b. Com isso, calculamos o ângulo com a função arco seno. Com isso, é fácil chegar à resposta. Calculamos o outro ângulo agudo do triângulo, e multiplicando por 2 temos o tamanho do arco $P1-P2$. Basta agora pegar o teto da divisão de $2*\pi$ por esse arco para termos o número de lados do polígono que José irá percorrer.

Problema J. Wasserman, Rei das Águas

Seja dp a representação dos sufixos de S tal que $dp(i) = -1$ é equivalente a dizer que o sufixo $S_{i:}$ não tem tradução, enquanto que para $dp(i) \geq 0$ existe tradução, e a primeira palavra da tradução termina em $dp(i)$, ou seja, $S_{i:dp(i)} \in N$ onde N é o dicionário das traduções.

Uma vez computado dp, basta seguir os ponteiros a partir de $dp(0)$ e imprimir as traduções. Para computar dp:

$$dp(|S|) = |S|$$
$$dp(i) = \begin{cases} \max(j \ \forall n_j \in N \text{ tal que } n_j = S_{i:j} \text{ e } dp(j) \geq 0) \\ -1 \text{ caso contrário} \end{cases}$$

Mesmo usando hashes, computar essa DP usando a forma acima tem custo de $O(|S||N|)$. Porém, podemos resolver usando uma variação do Aho-Corasick, em $O(|S| + |N||n_i|)$.

Problema K. Cobra de Dominós

No problema "Cobra de Dominós", dado um dominó com peças que tenham N valores distintos (de 0 até $N-1$, por exemplo) a equipe deve desenvolver o raciocínio matemático para reconhecer a construção da sequência que descreve o número máximo de peças, $P(N)$, usadas para se construir uma cobra usando as regras de encaixe (mesmo valor dos lados das peças).

Por exemplo:

- Para $N = 1, P(N) = 1$. Uma maior cobra possível de se contruir é |0-0|
- Para $N = 2, P(N) = 3$. Uma maior cobra possível de se contruir é |0-0||0-1||1-1|
- Para $N = 3, P(N) = 6$. Uma maior cobra possível de se contruir é |0-0||0-1||1-1||1-2||2-2||2-0|
- Para $N = 4, P(N) = 9$. Uma maior cobra possível de se contruir é |0-0||0-1||1-1||1-2||2-2||2-2||2-3||3-3||3-0| (não usa |1-3|)
- Para $N = 5, P(N) = 15$. Uma maior cobra possível de se contruir é |0-0||0-1||1-1||1-2||2-2||2-2||2-3||3-3||3-0||0-4||4-1||1-3||3-4||4-4||4-2|
- Para $N = 6, P(N) = 19$. Uma maior cobra possível de se contruir é |5-0||0-0||0-1||1-1||1-2||2-2||2-2||2-3||3-3||3-0||0-4||4-1||1-3||3-4||4-4||4-2||2-5||5-5||5-1| (não usa |3-5|)

Usando a mesma técnica construtiva, percebe-se que $P(7) = 28, P(8) = 33, P(9) = 45, \ldots$

Essa sequência pode ser descrita por

$P(N) = C(N, 2) + N$, (se N é ímpar) ou $P(N) = C(N, 2) + N/2$, (se N é par);

que pode ser simplificada para

$P(N) = (((-1)^N) * (2 - N) + 2 + N + 2 * N^2)/4$.

Problema L. História de Pescador

A solução esperada para o problema.

- Leitura dos inteiros M, N. - Laço de repetição para i de 1 a M: - Leitura da quantidade Q_i - Impressão do valor $N * Q_i$ na formatação solicitada

Análise assintótica da solução esperada: O(M).