Problemas e estatísticas

11 de novembro de 2017

Problema H: Hard choice

Título *Hard choice*
Autora *Inés Kereki*

Descrição: Dados três tipos de refeição, sabe-se o número de refeições disponíveis e de refeições solicitadas. Pergunta-se o número mínimo de pessoas que não vão receber a refeição solicitada.

Problema H: Ideia da Solução

Para cada uma das três refeições, somamos a diferença entre o número de pratos requisitados e o número de pratos disponíveis, se esse número for positivo.

Problema H: Estatísticas

Primeira submissão correta

Time	UEA - Wesley e os Safadões
Tempo	2 minutos

Número de submissões	76
Número de submissões corretas	72 (94%)
Número de times que submeteram	72
Número de times que acertaram	72 (100%)
Número de submissões no blind	0

Problema C: Complete Naebbirac's sequence

Título *Complete Naebbirac's sequence*
Autor *Yonny Mondelo Hernández*

Descrição: Dada uma sequência de $1 \leq N \leq 1000$ números, descobrir se podemos remover um número e/ou adicionar um número à sequência para que todo número entre 1 e K apareça exatamente a mesma quantidade de vezes.

Problema C: Ideia da Solução

Esse problema admite várias soluções com complexidades diferentes. Uma solução linear é a seguinte: Dependendo da mudança no tamanho da sequência, cada um dos K elementos vai aparecer $(N-1)/K$, N/K ou $(N+1)/K$ vezes se existir resposta. Como $K \geq 3$, o único número inteiro da lista acima pode ser obtido pela expressão $\lfloor \frac{N+1}{K} \rfloor$. Chamemos esse número de M.

Contamos então quantas vezes cada elemento aparece. Todos devem aparecer M vezes, exceto no máximo um elemento que pode aparecer $M-1$ vezes e/ou no máximo um elemento que pode aparecer $M+1$ vezes.

Problema C: Estatísticas

Primeira submissão correta

Time	UFPE - ALT
Tempo	13 minutos

Número de submissões	166
Número de submissões corretas	60 (36%)
Número de times que submeteram	69
Número de times que acertaram	60 (86%)
Número de submissões no blind	5

Problema I: Imperial roads

Título *Imperial roads*
Autor *Edwin Niño*

Descrição: Dado um grafo com pesos G com até 200.000 arestas, recebemos várias consultas. Cada consulta é uma aresta (u, v), e pede-se a árvore geradora de G de menor custo que contém a aresta (u, v).

Problema I: Ideia da Solução

Inicialmente, computamos uma árvore geradora mínima T do grafo. As consultas com arestas $(u, v) \in E(T)$ devem retornar o custo de uma árvore geradora mínima.

Se a consulta (u, v) não corresponder à uma aresta da árvore, achamos, na árvore T, a maior aresta no caminho entre u e v (o que pode ser feito com LCA e *sparse table*, por exemplo). Removemos essa aresta e inserimos a aresta (u, v) para obter a árvore mínima ótima. A prova de corretude é a mesma dos algoritmos usuais de árvore geradora mínima (ver CLRS).

Problema I: Estatísticas

Primeira submissão correta

Time	UFU - Ahozinho com Feijão
Tempo	45 minutos

Número de submissões	100
Número de submissões corretas	30 (30%)
Número de times que submeteram	42
Número de times que acertaram	29 (69%)
Número de submissões no blind	8

Problema E: Enigma

Título *Enigma*
Autor *Jeferson Lesbão*

Descrição: Dado um número S com alguns dígitos apagados, preencher os dígitos faltantes de modo a obter o menor número divisível por N.

Problema E: Ideia da Solução

Inicialmente, devemos utilizar programação dinâmica para descobrir se existe uma forma de preencher os digitos gerando um número divisível por N.

Se o numero existir, reconstroi-se o caminho percorrido pela programação dinâmica para gerar o numero esperado. Os estados utilizados pela PD são o índice na string S e o numero sendo gerado módulo N.

Problema E: Estatísticas

Primeira submissão correta

Time	USP-São Carlos - Trei Linha
Tempo	10 minutos

Número de submissões	70
Número de submissões corretas	25 (35%)
Número de times que submeteram	39
Número de times que acertaram	25 (64%)
Número de submissões no blind	1

Problema F: Fundraising

Título *Fundraising*
Autor *Paulo Cezar Pereira Costa*

Descrição: Recebemos N pessoas, cada uma representada por três valores: Sua beleza, sua riqueza e a doação que fará se for convidada a um jantar. Queremos escolher pessoas a convidar de modo a maximizar a doação, mas sem que duas pessoas briguem. Duas pessoas brigam quando uma delas é estritamente mais bela *xor* estritamente mais rica que outra.

Problema F: Ideia da Solução

Para evitar casos particulares, note que podemos primeiro juntar pessoas com mesma beleza e riqueza (somando suas doações). Para quaisquer duas pessoas do conjunto escolhido, uma delas tem que ser mais rica e mais bela que a outra.

Ordenamos as pessoas em ordem crescente de riqueza. Ao processar uma pessoa P (com beleza B), mantemos uma estrutura (BIT ou segment tree) que permite achar a maior doação possível usando apenas pessoas já processadas de beleza menor que B. Podemos então somar a doação de P a esse valor, atulizando a estrutura.

Para lidar com o caso de empate, ordenamos em ordem crescente de riqueza e, como desempate, em ordem decrescente de beleza.

Problema F: Estatísticas

Primeira submissão correta

Time	UnB-FGA - Teorema de Offson
Tempo	33 minutos

Número de submissões	98
Número de submissões corretas	24 (24%)
Número de times que submeteram	40
Número de times que acertaram	24 (60%)
Número de submissões no blind	6

Problema J: Jumping Frog

Título *Jumping Frog*
Autor *Gabriel Poesia*

Descrição: É dada uma string de N bits. Dizer quantos valores de K existem tais que se pode começar em alguma posição i e dar saltos de K em $K \pmod N$ até voltar à posição inicial passando apenas por 1s.

Problema J: Ideia da Solução

- Se $\text{mdc}(K, N) = 1$, então todas as posições serão visitadas.
- Mais geral: se $\text{mdc}(K, N) = d$, e começamos na posição p, todas as posições i com $i \equiv p \mod N$.
- Há "poucos" valores possíveis de d, já que d tem que ser um divisor de N ($\sigma_0(N) \leq 2\sqrt{N}$).
- Para cada valor de d, há d valores de p distintos, cada um visitando N/d posições.
- Para cada i, pode-se checar todas as posições com o mesmo valor mod N em $O(N/d)$.
- Verificando para cada valor de i, fica $O(N)$ (para um d fixo).
- Como há $\sigma_0(N)$ valores de D, basta verificar todos eles em $O(\sigma_0(N) \times N) = O(N \times \sqrt{N})$.
- Algoritmo: pré-computar resposta para cada d. Para cada K, computar $d_k = mdc(K, N)$ e somar 1 se for possível para d_k. $O((\sigma_0(N) + \log N) \times N)$.

Problema J: Estatísticas

Primeira submissão correta

Time USP - ¯_(”/)_/¯
Tempo 40 minutos

Número de submissões	71
Número de submissões corretas	23 (32%)
Número de times que submeteram	32
Número de times que acertaram	19 (59%)
Número de submissões no blind	12

Problema B: Buggy ICPC

Título *Buggy ICPC*
Autor *Gabriel Poesia*

Descrição: Dada uma string, dizer de quantas formas se pode digitá-la sabendo que o teclado tem um erro: após uma vogal ser digitada (sempre à direita), a string é invertida. Exemplo: "ac"pode ser digitada como "ac" e "ca".

Problema B: Ideia da Solução

- É útil pensar no último caractere que pode ter sido digitado. Observe que se uma string começa com consoante e tem vogais, é impossível digitá-la.
- Solução: repetir enquanto a string não está vazia:
  1. Se a string começa e termina com consoante, tirar a última consoante (única opção).
  2. Se a string começa com vogal e termina com vogal, inverter a string e tirar a última vogal.
  3. Se a string começa com consoante e termina com vogal, pare (impossível digitar).
  4. Se a string começa com vogal e termina com consoante, há duas opções de último caractere digitado.
     1. No caso de a vogal ter sido digitada por último, só é possível digitar o resto se não sobraram vogais. Neste caso, somar 1 na resposta.
     2. Tirar a consoante e continuar.
- Implementando de forma "esperta", fica $O(n)$.

Problema B: Estatísticas

Primeira submissão correta

Time	UFRJ - Esse é meu time
Tempo	47 minutos

Número de submissões	41
Número de submissões corretas	16 (39%)
Número de times que submeteram	23
Número de times que acertaram	15 (65%)
Número de submissões no blind	7

Problema G: Gates of uncertainty

Título *Gates of uncertainty*
Autor *Ricardo Anido*

Descrição: Dado um circuito feito de portas NAND com algumas portas quebradas (que retornam sempre 0 ou sempre 1), queremos encontrar o número de maneiras de escolher as entradas do circuito de modo que a saída do circuito comprove que há portas quebradas.

Problema G: Ideia da Solução

Pensamos no circuito como uma árvore binária. Para cada nó v, computamos 4 números: Para $r \in \{0, 1\}$ e $d \in \{0, 1\}$, contamos o número de maneiras de dar valores para as entradas da subárvore de v tal que a saída do nó v é r e deveria ser d. Os quatro valores de um nó v dependem apenas dos mesmos valores para seus filhos.

Problema G: Estatísticas

Primeira submissão correta

Time	USP - ¯_(”/)_/¯
Tempo	107 minutos

Número de submissões	29
Número de submissões corretas	14 (48%)
Número de times que submeteram	18
Número de times que acertaram	13 (72%)
Número de submissões no blind	9

Problema D: Daunting device

Título *Daunting device*
Autor *Walter Erquinigo*

Descrição: Temos L células a colorir. Recebemos várias operações do tipo "mude a cor dos números entre M_1 e M_2 para X", onde os números M_1 e M_2 são calculados do número atual de células de uma cor P. Queremos saber quantas vezes a cor mais frequente aparece depois de todas as operações.

Problema D: Ideia da Solução

Representaremos o estado atual como uma união de intervalos tal que todas as células de um intervalo têm a mesma cor.

Podemos manter os intervalos "ativos" num `set`. Ao adicionar um intervalo I, removemos todos os intervalos que contêm I e ajustamos os (até dois) intervalos que intersectam I. A cada alteração no `set`, tomamos o cuidado de atualizar a quantidade de números da cor correspondente.

Cada intervalo que passa pelo `set` é deletado apenas uma vez, de modo que a complexidade amortizada das N atualizações é $O(N \log L)$. O tempo total é $O(C + N \log L)$.

Problema D: Estatísticas

Primeira submissão correta

Time	USP - dog hits dog
Tempo	194 minutos

Número de submissões	25
Número de submissões corretas	5 (20%)
Número de times que submeteram	10
Número de times que acertaram	5 (50%)
Número de submissões no blind	5

Problema M: Marblecoin

Título *Marblecoin*
Autor *Paulo Cezar Pereira Costa*

Descrição: Dados N ($N \leq 10^5$) arrays de números entre 1 e 300 com no máximo 400.000 elementos no total, fazer o *merge* lexicograficamente mínimo dos arrays.

Problema M: Ideia da Solução

Iremos construir a resposta elemento a elemento, escolhendo o array que fornecerá o próximo número. O problema são empates! Escolhendo o array lexicograficamente mínimo, melhoramos nossas escolhas no futuro (talvez muitos passos depois). Inserimos um elemento 301 no final de cada array para tratar o caso de prefixos, pois em caso de empate acabar com um array dá menos escolhas no futuro.

Talvez não muito surpreendentemente, usaremos arrays de sufixo para comparar lexicograficamente os sufixos dos arrays eficientemente. Para isso, pensamos nos números de 1 a 301 como letras de um alfabeto. Usamos uma fila de prioridade para decidir qual será o próximo número em cada passo.

Precisamos de tempo linear para construir o array de sufixo, e tempo $O(\log N)$ por letra para atualizar a fila de prioridade.

Problema M: Estatísticas

Primeira submissão correta

Time	ITA - CalopsITA
Tempo	258 minutos

Número de submissões	35
Número de submissões corretas	11 (31%)
Número de times que submeteram	7
Número de times que acertaram	3 (42%)
Número de submissões no blind	20

Problema L: Linearville

Título *Linearville*
Autor *Guilherme A. Pinto*

Descrição: Dado um grid de ruas horizontais e verticais, com espaço de 1 ou 5 metros entre ruas consecutivas, achar o menor caminho entre duas esquinas com a restrição de que o caminho tem que ser alternado: Usar um trecho de rua no sentido Norte–Sul, depois um trecho no sentido Leste–Oeste e assim por diante (ou vice-versa).

Problema L: Ideia da Solução

Primeiro calculamos o número de quarteirões (independente de tamanho) que precisamos andar em cada uma das direções. Se a diferença entre esses números for maior que 1, precisamos "gastar tempo" na outra direção.

Para isso, para cada direção encontramos o espaço de 1 metro mais próximo em cada uma dos sentidos. Para a direção com a maior quantidade de passos, temos então duas opções: Gastar tempo com passos de 5 metros, ou andar até um dos dois espaços mais próximos de 1 metro, gastar tempo lá e depois ir até o ponto final. Tentamos todas as opções.

Problema L: Estatísticas

Primeira submissão correta

Time USP - ¯_(”/)_/¯
Tempo 195 minutos

Número de submissões	2
Número de submissões corretas	1 (50%)
Número de times que submeteram	2
Número de times que acertaram	1 (50%)
Número de submissões no blind	0

Problema A: Arranging tiles

Título *Arranging tiles*
Autor *Guilherme A. Pinto*

Descrição: Dados N ($N \leq 14$) polígonos convexos com altura H, posicioná-los da esquerda para a direita de modo que a "largura" total da menor caixa (de altura H) que os contenha seja mínima.

Problema A: Ideia da Solução

Primeiro problema: Dada uma ordem, computar a largura ótima. Para isso, é suficiente saber a largura ótima para posicionar dois polígonos.

Posicionemos os dois polígonos muito distantes um do outro. A interseção da reta horizontal de altura k com os dois polígonos são segmentos de reta, e chamaremos a distância entre esses dois segmentos de $d(k)$. d é uma função linear por partes.

Fazemos um line sweep para calcular o menor valor atingido por d quando a altura y varia de 0 a H. Podemos então aproximar os dois polígonos por essa distância.

Segunda parte: Encontrar a ordem ótima. Para isso, faz-se uma programação dinâmica similar à do caixeiro-viajante, com complexidade $O(2^N N^2)$.

Problema A: Estatísticas

Primeira submissão correta

Time	UFRN - Ginga com Tapioca
Tempo	297 minutos

Número de submissões	3
Número de submissões corretas	1 (33%)
Número de times que submeteram	1
Número de times que acertaram	1 (100%)
Número de submissões no blind	1

Problema K: Keep it covered

Título *Keep it covered*
Autor *Paulo Cezar Pereira Costa*

Descrição: Preencher um tabuleiro 20×20 tal que toda célula participe de um ciclo ou caminho. Até 15 células do tabuleiro contém círculos e correspondem a células que *tem* que corresponder a um extremo de um caminho, e as demais *não podem* ser extremos de caminhos.

Problema K: Ideia da Solução

Montamos um grafo bipartido do seguinte modo: Cada célula é representada por um vértice se ela tem um círculo e dois vértices caso contrário. Ligamos dois vértices se as células correspondentes são adjacentes. A resposta é "Y" se e só se o grafo tem um emparelhamento perfeito, e as arestas do emparelhamento formam a resposta quando marcadas no tabuleiro.

Problema K: Estatísticas

Primeira submissão correta

Time	(nenhum AC)
Tempo	(nenhum AC)

Número de submissões	0
Número de submissões corretas	0 (nenhuma submissão)
Número de times que submeteram	0
Número de times que acertaram	0 (nenhuma submissão)
Número de submissões no blind	0