Brazilian ICPC Summer School 2022

Resolução Contest 1: HCW19

- HCW19
- Problemas de Segment Tree

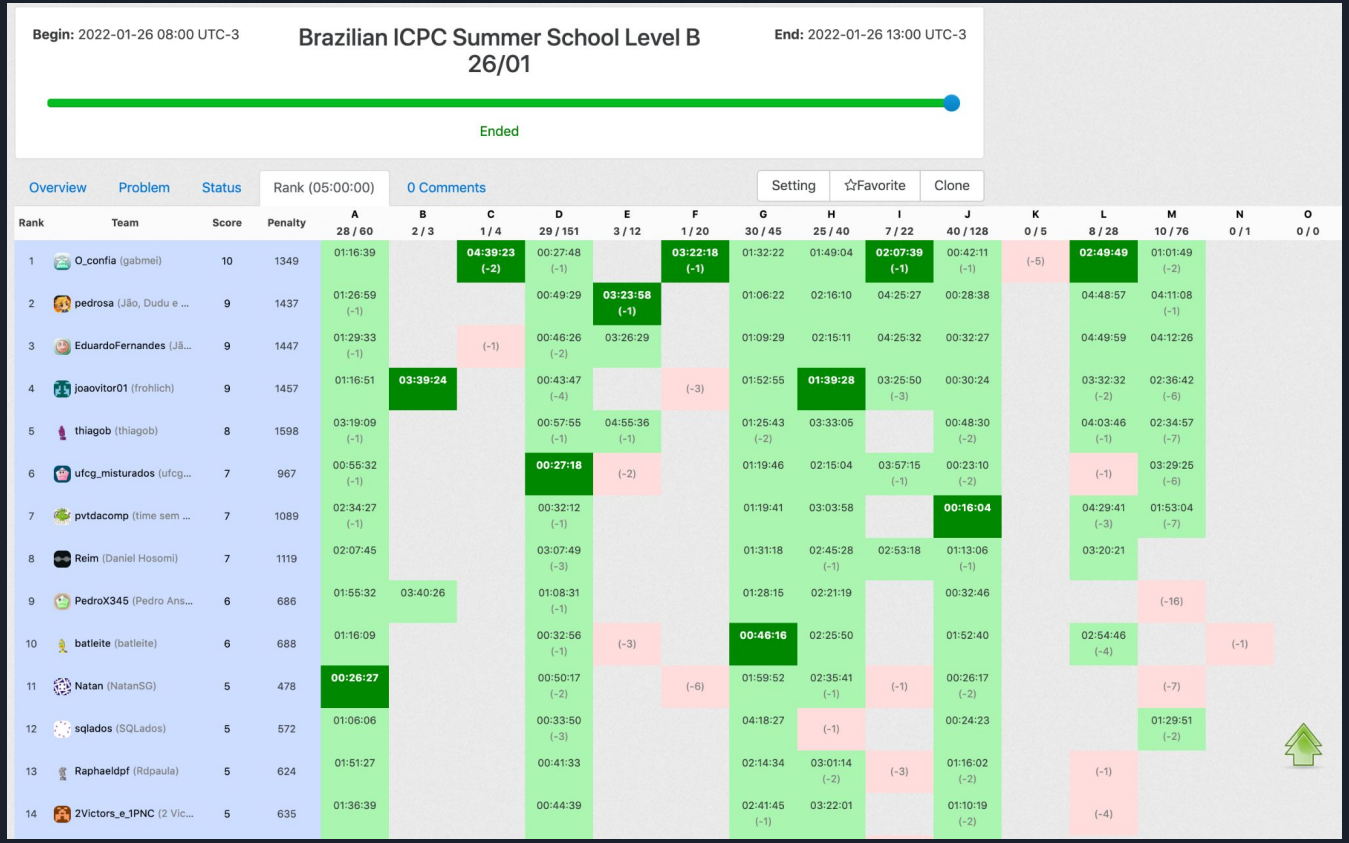
André Amaral de Sousa

Autor

Sobre

- Ex-Maratonista (ICPC 2013)
- Ex-Olímpico (IOI 2012)
- Problem Setter na Brasileira 2019 e 2020
- Engenheiro de Computação (POLI-USP)
- Professor no colégio ETAPA
- Contato:
  - andremfq@gmail.com

Placar



Begin: 2022-01-26 08:00 UTC-3

Brazilian ICPC Summer School Level B 26/01

End: 2022-01-26 13:00 UTC-3

Ended

Overview | Problem | Status | Rank (05:00:00) | 0 Comments | Setting | ☆Favorite | Clone

Rank	Team	Score	Penalty	A 28 / 60	B 2 / 3	C 1 / 4	D 29 / 151	E 3 / 12	F 1 / 20	G 30 / 45	H 25 / 40	I 7 / 22	J 40 / 128	K 0 / 5	L 8 / 28	M 10 / 76	N 0 / 1	O 0 / 0
1	O_confia (gabmei)	10	1349	01:16:39		**04:39:23 (-2)**	00:27:48 (-1)		**03:22:18 (-1)**	01:32:22	01:49:04	**02:07:39 (-1)**	00:42:11 (-1)	(-5)	**02:49:49**	01:01:49 (-2)		
2	pedrosa (Jão, Dudu e ...	9	1437	01:26:59 (-1)			00:49:29	**03:23:58 (-1)**		01:06:22	02:16:10	04:25:27	00:28:38		04:48:57	04:11:08 (-1)		
3	EduardoFernandes (Jã...	9	1447	01:29:33 (-1)		(-1)	00:46:26 (-2)	03:26:29		01:09:29	02:15:11	04:25:32	00:32:27		04:49:59	04:12:26		
4	joaovitor01 (frohlich)	9	1457	01:16:51	**03:39:24**		00:43:47 (-4)		(-3)	01:52:55	**01:39:28**	03:25:50 (-3)	00:30:24		03:32:32 (-2)	02:36:42 (-6)		
5	thiagob (thiagob)	8	1598	03:19:09 (-1)			00:57:55 (-1)	04:55:36 (-1)		01:25:43 (-2)	03:33:05		00:48:30 (-2)		04:03:46 (-1)	02:34:57 (-7)		
6	ufcg_misturados (ufcg...	7	967	00:55:32 (-1)			**00:27:18**	(-2)		01:19:46	02:15:04	03:57:15 (-1)	00:23:10 (-2)		(-1)	03:29:25 (-6)		
7	pvtdacomp (time sem ...	7	1089	02:34:27 (-1)			00:32:12 (-1)			01:19:41	03:03:58		**00:16:04**		04:29:41 (-3)	01:53:04 (-7)		
8	Reim (Daniel Hosomi)	7	1119	02:07:45			03:07:49 (-3)			01:31:18	02:45:28 (-1)	02:53:18	01:13:06 (-1)		03:20:21			
9	PedroX345 (Pedro Ans...	6	686	01:55:32	03:40:26		01:08:31 (-1)			01:28:15	02:21:19		00:32:46			(-16)		
10	batleite (batleite)	6	688	01:16:09			00:32:56 (-1)	(-3)		**00:46:16**	02:25:50		01:52:40		02:54:46 (-4)		(-1)	
11	Natan (NatanSG)	5	478	**00:26:27**			00:50:17 (-2)		(-6)	01:59:52	02:35:41 (-1)	(-1)	00:26:17 (-2)			(-7)		
12	sqlados (SQLados)	5	572	01:06:06			00:33:50 (-3)			04:18:27	(-1)		00:24:23			01:29:51 (-2)		
13	Raphaeldpf (Rdpaula)	5	624	01:51:27			00:41:33			02:14:34	03:01:14 (-2)	(-3)	01:16:02 (-2)		(-1)			
14	2Victors_e_1PNC (2 Vic...	5	635	01:36:39			00:44:39			02:41:45 (-1)	03:22:01		01:10:19 (-2)		(-4)			

HCW 19

Comentários Gerais:

- Contest disponível no GYM
- 5h, 3 estrelas
  - Muitos contests 3 estrelas estão num nível legal pra Maratona Brasileira (entre a primeira e segunda fase)
  - Mesmo contests mais fáceis são muito úteis (vários têm problemas difíceis legais, e mesmo os problemas médios servem para treinar velocidade)
- Tutorial
  - Não falarei de todos os problemas
- É muito importante upsolvar os problemas
  - Pense nos problemas que não teve tempo durante o contest
  - Se mesmo após pensar não conseguir resolver, leia o tutorial e resolva
-

B. Beggin' For A Node

Comentários Gerais: não vou resolver

- Problema bem legal
- Este problema fica fácil se você domina Centroid Decomposition
  - Se você nunca estudou, recomendo que estude
  - Se já havia visto, e não conseguiu identificar, precisa treinar mais
- O tutorial descreve uma solução que usa apenas LCA, DFS and Binary Search
  - Ideia muito legal, recomendo aprender, mas com Centroid Decomposition fica bem mais fácil

C. Countering Terrorists

Ideia

- Fazer Busca Binária no valor de w. Para um dado w precisamos verificar se é possível eliminar todas as n bombas, com P dispositivos do tipo 1 (tamanho w) e Q dispositivos do tipo 2 (tamanho 2w)
- Para responder pra um w específico, iremos fazer uma DP. DP[i][j] vai guardar a menor quantidade de dispositivos do tipo 2 necessária para eliminar todas as bombas de 1 à i (bombas ordenadas) usando no máximo j dispositivos do tipo 1.
    - Se temos essa DP calculada, basta verificar se DP[n][P] <= Q então conseguimos com esse valor de w, senão não conseguimos.
    - Procure entender bem essa DP, no fundo ela é simples, mas não é uma DP comum.

C. Countering Terrorists

Ideia

- Como calcular DP[i][j] ?
- Para eliminar a bomba i, que está em v[i], temos duas possibilidades, usando um dispositivo do tipo 1 ou usando um dispositivo do tipo 2.
- DP[k1][j - 1], sendo k1 a primeira bomba que não será afetada pelo dispositivo de tamanho w, ou seja, maior índice tal que v[k1] < v[i] - w, dá pra encontrar com Busca Binária ou manter com Two Pointers
- DP[k2][j] + 1, analogamente sendo k2 a primeira bomba que não será afetada pelo dispositivo de tamanho 2w,

C. Countering Terrorists

Ideia

- Quanto fica a complexidade ?
- O(log(10^9)) passos da Busca Binária no w
- A cada passo temos uma DP que é O(N*P)
- Mas isso dá TLE: 30 * 2000 * 10^5 = 6 * 10^9
- Observação: se P + Q >= n então a resposta é 1. Desta forma só iremos fazer a Busca Binária quando P + Q < n, assim P < 2000, e portanto a complexidade fica 30 * 2000 * 2000 = 1,2 * 10^8 que passa!

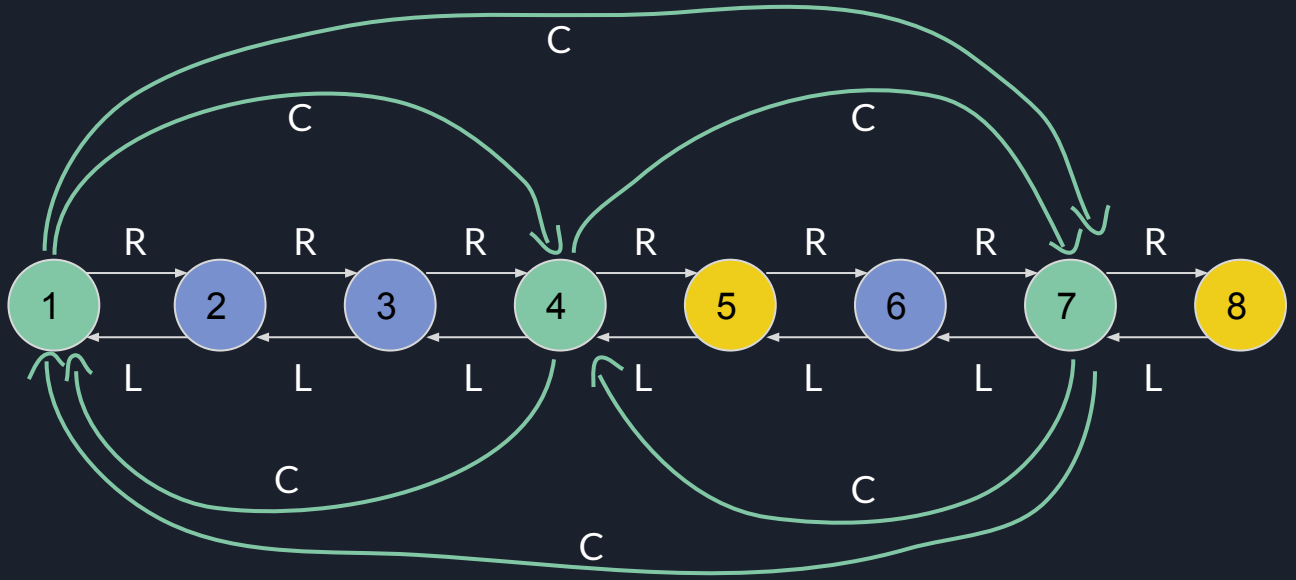
E. Elevate To Dominate

Comentários Gerais: não vou resolver

- Problema clássico de Convex Hull Trick (Otimização de DP)
  - Se você nunca estudou, recomendo que estude
  - Se já havia visto, e não conseguiu identificar, precisa treinar mais

L. Left or Right? How about neither?

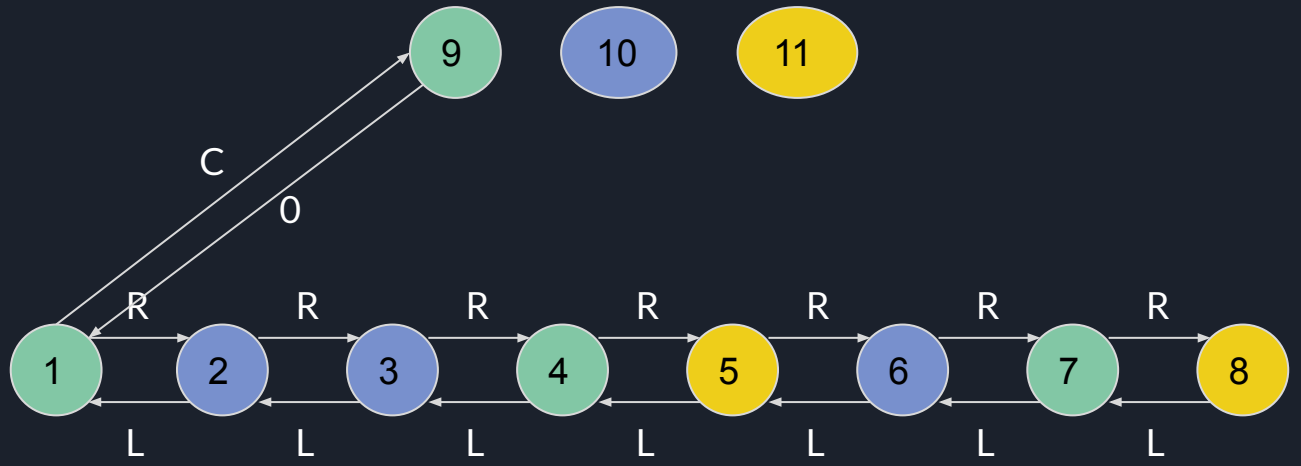
Ideia



- Problema de caminho mínimo num grafo com pesos não negativos -> Dijkstra
- Mas o problema são os teletransportes, pois podemos ter $O(N^2)$ arestas, como arrumar ?

L. Left or Right? How about neither?

Ideia



- Criar um novo vértice para cada valor.
  - Cuidado: a princípio os valores vão de 1 à 10^9, mas podemos comprimir coordenadas
- Assim o número de vértices fica no máximo 2*N, e o número de arestas no máximo 4*N, e portanto podemos finalizar usando Dijkstra

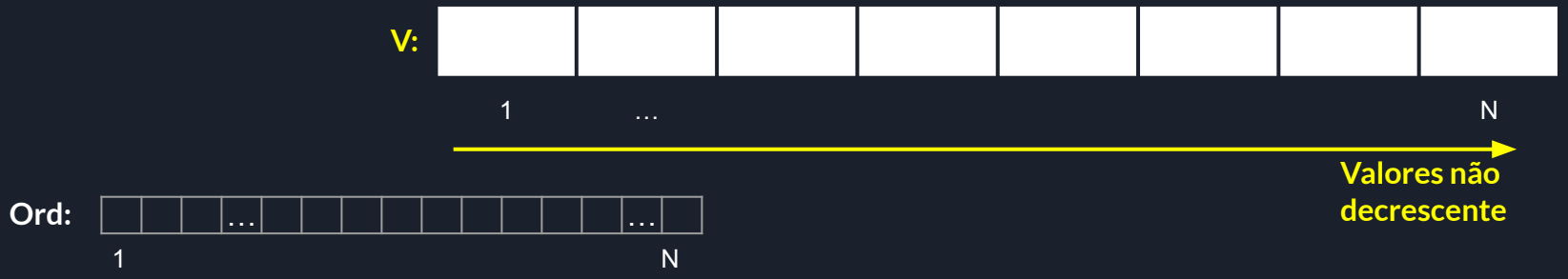
M - GSS4

Ideia

- Quantas vezes podemos tirar raiz quadrada de um número ?
  - 10^18 -> 10^9 -> 31622 -> 177 -> 13 -> 3 -> 1 -> 1 -> 1 ....
- Após chegar em 1 o número não se altera, e demora no máximo 6 operações para chegar no 1. Portanto cada número só precisa ser alterado no máximo 6 vezes.
- Manter em cada nó a soma dos valores e um booleano dizendo se todos os valores naquele nó são iguais a 1 ou não.
- Ao fazer update, se um nó já possui todos os valores iguais a 1, não precisamos entrar nesse nó pois nada será alterado. Isso garante que iremos alterar apenas o que for necessário
- Complexidade total O(6*N*logN + MlogN)

N - Temple Queues

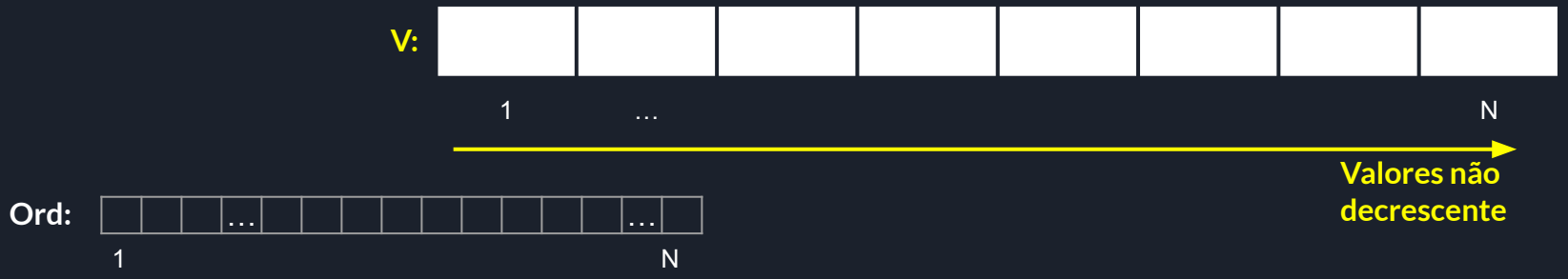
Ideia



- Imagine que iremos manter um vetor V com as quantidades de cada fila, porém ordenado de forma não decrescente.
- Para isso, mantemos para cada fila i original, em qual posição ela está. E pra cada posição também mantemos qual o índice original que ela estava.
- Na verdade iremos manter uma Segment Tree que representa esse vetor V.
- Para a operação 2 basta fazer Busca Binária na Seg (encontrar o índice do primeiro valor que seja maior ou igual a X)

N - Temple Queues

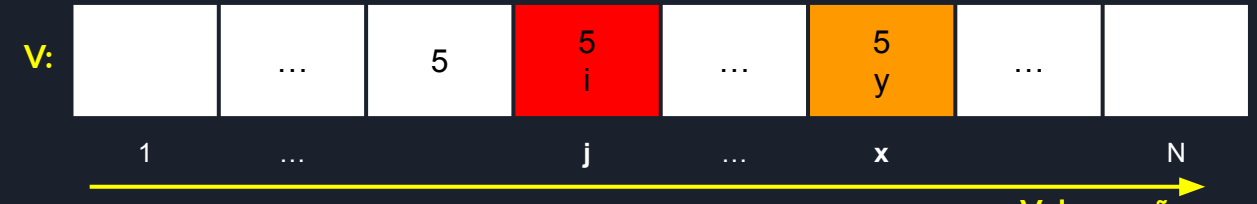
Ideia



- Para a operação 3, além da Busca Binária na Seg (encontrar o índice do primeiro valor que seja maior ou igual a Y) basta fazer update em intervalo, usando Lazy Propagation
- O grande problema é a operação 1, pois ela a princípio pode deixar os valores de forma não ordenada, como consertar ?

N - Temple Queues

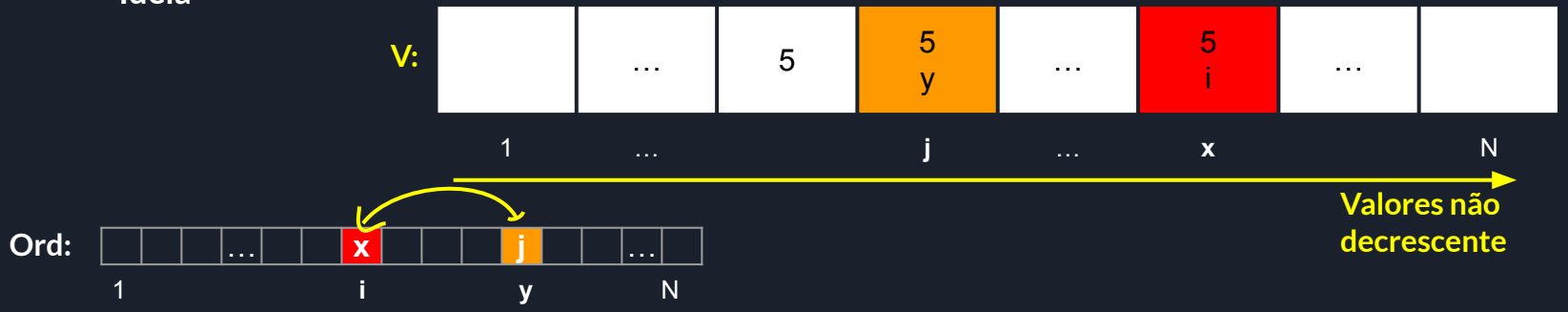
Ideia



- Suponha que uma pessoa vai entrar na fila i (índice original), e que nesse momento essa fila esteja no índice j da Seg.
- Usando Busca Binária na Seg obtemos o último índice x que possui o mesmo valor que j
- Observe que se somarmos 1 no valor de j na Seg, isso a tornará não ordenada, mas se somarmos 1 no valor de x, como se trata do último índice com aquele valor, e estamos somando apenas 1, manterá ordenado.

N - Temple Queues

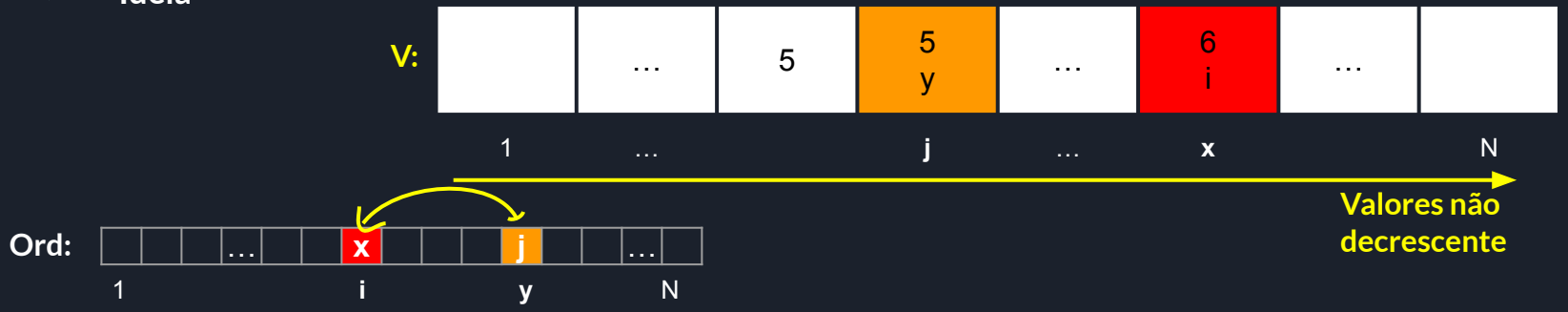
Ideia



- Portanto a ideia é trocar as filas j e x de lugar. Perceba que na Seg nada muda pois elas têm o mesmo valor, mas altera as estruturas que guardam as posições das filas (se quiser pode guardar na seg qual o índice original, mas recomendo usar outro vetor auxiliar)

N - Temple Queues

Ideia



- Agora podemos somar 1 no índice x da Seg, que é onde a fila original i está, e a Seg continuará com os valores ordenados

O - Frogs and mosquitoes

Ideia

- Ordene todos os Frogs pela sua coordenada x (cuidado para imprimir a resposta)
- Iremos manter um Seg nessa ordem, onde iremos guardar pra cada Frog o seu final yi (yi = xi + ti inicialmente), e em cada nó da Seg manteremos o valor máximo.
- Também iremos manter um Set S com as posições dos mosquitos que não foram comidos (multiset pois acredito que os mosquitos podem aparecer em posições iguais)
- Para processar um mosquito, j:
  - Encontrar o menor índice i tal que yi >= pj, usando Busca Binária na Seg
  - Se xi > pj então esse mosquito não será comido agora, então adicione ele à S
  - Senão esse frog i irá comer o mosquito j, então some bj em yi
- Toda vez que um frog comer um mosquito, devemos checar se ele não irá comer mais mosquitos que não foram comidos, e que estão em S. Para isso basta determinar o primeiro mosquito após a posição inicial do frog (xi) por exemplo com lower_bound. E verificar se esse frog consegue comer esse mosquito

O - Frogs and mosquitoes

Ideia

- Complexidade:
  - NlogN para ordenar os frogs
  - MlogN para processar cada mosquito (BB na Seg + update se necessário)
  - MlogM operação no Set, pois cada mosquito entra no máximo uma vez e sai no máximo uma vez