Quarentreta

Essa é uma variação do Problema de Josephus, que tem implementação recursiva de custo $O(n)$:

$$josephus(n,k) = ((josephus(n-1,k) + k - 1)\%n) + 1$$

$$josephus(1,k) = 1$$

Como Gelaldo sempre é o primeiro a contar, para cada um dos k valores das amigos, basta verificar para qual valor de k o número de josephus é 1.

Esta solução é $O(nk)$

Reabrindo Cinemas

Para solucionar o problema deve-se verificar a cada teste se todos os vizinhos a uma unidade de distância da cadeira a ser comprada estão livres.

Similaridade entre RNAs

O problema pode ser resolvido em tempo $O(N)$ ao calcular a distância de Hamming entre as duas strings (sequências) X e Y. Considerando que $X = x_1x_2...x_N$ e $Y = y_1y_2...y_N$, a ideia da distância de Hamming é percorrer ambas strings, caractere por caractere, e contabilizar as posições simultâneas x_i e y_i cujos caracteres não são iguais.

Entrando em Forma

Para resolver o problema é preciso, inicialmente, determinar a massa m de John. Isto pode ser feito por meio da expressão

$$m = I \times h^2$$

A massa alvo $\hat{m}$ do IMC desejado é computada da mesma forma:

$$\hat{m} = S \times h^2$$

Assim, a resposta será dada por $m - \hat{m}$, e a solução tem complexidade $O(1)$.

Maratonando Cursos

Este problema pode ser resolvido através de uma abordagem gulosa em tempo $O(N \lg N + M^2)$.

Primeiramente os cursos são ordenados em ordem decrescente pelo conhecimento e, em caso de empate, pela quantidade de semanas de gratuidade do curso, dando preferência pelo curso com mais semanas de gratuidade.

Após esta ordenação, podemos criar um vetor **booleano** de tamanho $M + 1$ indicando representando a disponibilidade de Patrícia em cada semana. Inicialmente, todos os elementos são verdadeiros.

De acordo com a ordem criada, sejam g_i e v_i a quantidade de semanas de gratuidade e o conhecimento do i-ésimo curso. Caso seja possível enquadrar o i-ésimo curso na semana g_i, isto é feito, caso contrário, tentamos enquadrá-lo na semana $g_i - 1$ e assim sucessivamente. Caso tenha sido possível enquadrar o curso em alguma semana, adicionamos v_i à soma total de conhecimento e atualizamos o vetor de disponibilidade com falso na semana em que foi possível enquadrar o curso. Sempre que tentamos enquadrar um curso em uma semana é preciso verificar se aquela semana já não está ocupada.

Seguindo esta estratégia gulosa, estamos dando prioridade para os cursos que propiciam maior conhecimento e estamos postergando ao máximo estes cursos para que, caso haja um curso com menor tempo de gratuidade mas com valor significativo de conhecimento, ele consiga ser alocado nas semanas de Patrícia.

Atendimento

A solução do problema consiste em simular a situação descrita no problema. Dois cuidados, porém, são necessários, para uma solução eficiente.

O primeiro ponto é o cálculo da distância entre dois retângulos. Este problema pode ser dividido em dois sub-problemas: a interseção entre as projeções dos retângulos nos eixos coordenados. Se as projeções em um eixo tem interseção com área não-nula, significa que existe ao menos um ponto em cada retângulo cuja distância é a diferença, em valor absolutos, de suas coordenadas no outro eixo.

Assim, faça $d_x = 0$, se há interseção não nulo entre as projeções no eixo-x, ou $d_x = d$, onde d é a distância entre estas projeções. Faça o mesmo com d_y e a distância entre os retângulos será $d_r = \sqrt{d_x^2 + d_y^2}$.

O segundo ponto é manter um ponteiro para o próximo hospital a ser avaliado caso o hospital em questão já esteja com todas as vagas preenchidas. Caso o próximo hospital também esteja cheio, este ponteiro deve ser movido para o próximo, de modo a não tornar a considerar os hospitais da sequência já cheios. Isto pode ser feito mantendo as distâncias para o hospital i ordenadas, e movendo um ponteiro j quando necessário.

Organizando a gaveta

O número de sequências distintas b é obtido por meio de um permutação com repetições

$$b = P(n, n_1, n_2, \ldots, n_k) = \frac{n!}{n_1! n_2! \ldots n_k!},$$

onde n_i é o número de camisas com a cor i. O número de sequências desejadas a é a permutação das k cores distintas, que indicam a ordem dos blocos, isto é, $a = k!$.

Sendo $d = (a, b)$ o maior divisor comum entre a e b, a resposta esperada é $p = a/d$ e $q = b/d$.

A complexidade da solução é $O(N)$, devido ao cálculo dos fatoriais.

Linhas de Ônibus

O problema pode ser resolvido fazendo-se uma simulação dos ônibus que passam pelo ponto de ônibus utilizando uma fila de prioridade mínima.

Pode-se declarar uma fila de prioridade mínima Q, em que cada nodo armazena uma tupla (tempo em minutos que um ônibus da linha k passa pelo ponto, a incidência de Covid-19 no bairro k, a linha do ônibus k). Primeiramente, deve-se ler a entrada e inserir em Q os primeiros ônibus de cada linha conforme a estrutura do nodo. A simulação das passagens dos ônibus pelo ponto até a chegada de Luis Paulo ocorre como:

- desenfileira o ônibus de Q, denotado pela tupla $< t_k, c_k, k >$
- verifica-se se Luis Paulo está no ponto de ônibus no momento t_k. Se sim, parar a simulação. Se não estiver, deve-se enfileirar em Q a tupla $< t_k + d_k, c_k, k >$, que é o registro do próximo ônibus da mesma linha que passará pelo ponto de ônibus.

Sabemos que o custo para inserir cada um dos N ônibus em Q é $O(Nlog(N))$. Como podemos fazer, na pior situação, até T inserções devido à duração máxima da simulação do processo, o custo computacional total dessa solução é $O(N.T.log(N))$.

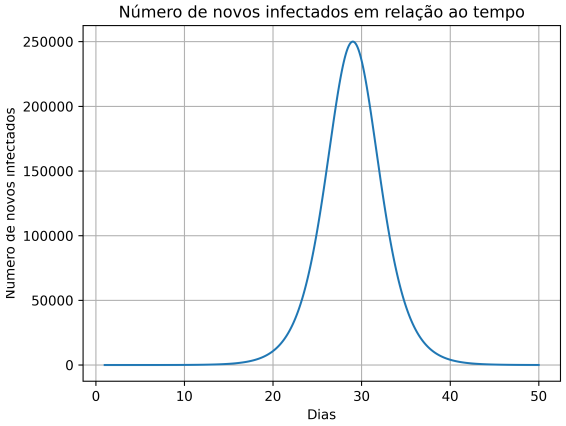
Encontrando o PIco

O que queremos saber é o momento em que a taxa de variação do número total de infectados em relação ao tempo, isto é, o ponto em que a derivada da função logística em relação ao tempo, é a maior possível.

Podemos calcular este ponto de várias formas. Duas delas serão apresentadas a seguir. A primeira é uma solução numérica e a segunda uma solução analítica.

Primeira solução

A função derivada é uma função unimodal, que cresce, chega a um pico, e logo diminui. O gráfico a seguir corresponde à derivada do primeiro exemplo de teste.



Desta forma, para achar o momento de pico e este número propriamente dito, basta aplicar a técnica de **busca ternária** sobre a derivada.

Para calcular o valor desta derivada em um ponto x específico durante a busca ternária, podemos usar a definição de derivada através de limite:

$$\lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

Com ϵ na prática sendo um valor minúsculo.

É importante também escolher um intervalo de busca grande o suficiente na busca ternária para garantir a otimalidade da solução.

Segunda solução

A derivada da função logística é:

$$f'(x) = \frac{abce^{bx}}{(e^{bx}+a)^2}$$

O objetivo é achar o valor máximo de $f'(x)$. O ponto de valor máximo é justamente o ponto em que a segunda derivada da função logística é 0, em outras palavras, estamos buscando x tal que $f''(x) = 0$.

Calculando a segunda derivada, temos:

$$f''(x) = -\frac{ab^2ce^{bx}(e^{bx}-a)}{(e^{bx}+a)^3}$$

Igualando $f''(x)$ a zero, obtemos que a raiz é justamente $x = \ln(a)/b$, este é o ponto em que $f'(x)$ é o maior possível.

Ao calcular $f'(\frac{\ln(a)}{b})$, obteremos $\frac{bc}{4}$, que é o número máximo de novos infectados.

Taxa de contaminaçao

Para resolver esse problema, basta dividir todos os termos da sequência por seu antecessor. Como a ordem máxima é 4, esta será a quantidade máxima de vezes que a sequência deve ser verificada.

Como os elementos são menores ou iguais a 3×10^4, é possível armazenar os elementos como frações cujo numerador e denominador são, no máximo, iguais a 10^{16}, o que pode evitar problemas com aritmética em ponto flutuante.

Máscaras

Este problema equivale a se determinar a cobertura de vértices mínima, que é um problema NP, de modo que a solução deve utilizar a busca completa. Assim, todas as possíveis distribuições devem ser avaliadas, de modo que a complexidade é $O(N^2 2^N)$.

Cara ou Corona

Os percentuais são simples de serem calculados:

$$P_{CARA} = \lfloor \frac{100 \cdot L}{L+G} \rfloor, P_{CORONA} = \lfloor \frac{100 \cdot G}{L+G} \rfloor$$

Esta solução é $O(1)$.

Bolhas Sociais

Uma maneira de resolver este problema em tempo $O(N + Q \cdot \alpha(N))$ é utilizar a estrutura de dados clássica com compressão de caminhos para união e pesquisa sobre conjuntos-disjuntos (union-find) com uma pequena modificação. Nesta notação, $\alpha(N)$ corresponde à função inversa de Ackermann sobre o parâmetro N, que é uma função que cresce absurdamente devagar.

Além de armazenar o membro representativo de cada conjunto, também podemos armazenar o tamanho do conjunto ligado àquele membro representativo. Para cada operação de pesquisa, utilizamos o tamanho calculado. Para cada operação de união de dois conjuntos disjuntos, somam-se os tamanhos dos dois conjuntos.

Esta ideia pode ser ilustrada pelo Algoritmo a seguir:

```
void union_set(int i, int j) {
    // Procura os elementos representativos de i e j
    auto x = find(i);
    auto y = find(j);
    // Se i e j não fazem parte do mesmo conjunto, a união é feita
    if (x != y) {
        if (rank[x] > rank[y]) {
            parent[y] = x;
            component_sz[x] += component_sz[y];
    } else {
        parent[x] = y;
        component_sz[y] += component_sz[x];
        if (rank[x] == rank[y]) {
            rank[y]++;
        }
    }
}
```

Problem Tutorial: "Vacinação"

A solução simples $O(N^2)$ resulta em veredicto *Time Limit Exceeded*, pois o valor máximo de $N = 10^5$, ultrapassando o tempo limite do problema, que é 1 segundo.

Por isso, o problema pode ser efetivamente resolvido utilizando a estrutura de dados *Delta Encoding*. Utilizaremos dois vetores *op* e *vac* para descrever a quantidade de operações de vacinação em cada um dos setores censitários e para quantificar a população que ainda não foi vacinada.

Primeiramente, deve-se contabilizar as operações de vacinação nos setores censitários no vetor *op* e quantificar os habitantes que são vacinados em cada uma delas. O custo computacional fica em O(M). Em seguida, vamos definir uma variável *ans* que armazenará a quantidade de operações de vacinação em setores com habitantes gripados que não foram vacinados. Deve-se realizar a soma de prefixos dos vetores *op* e *vac*, obtendo-se os vetores *psumo* e *psumv*, respectivamente. Agora percorre-se o vetor *psumv* e se $psumv_i$ for igual a zero, deve-se contabilizar as operações como $ans = ans + psumo_i$, que pode ser feito em O(N).

Por isso, o problema pode ser resolvido em um custo computacional $O(N + M)$.