A. Alimentação Saudável

Dados os limites apresentados pelo problema para as variáveis N e M, é possível resolver este problema construindo e contando as diferentes maneiras. Se identificarmos as frutas e os legumes e verduras por inteiros nos intervalos $[1, N]$ e $[1, M]$, respectivamente, o número E de escolhas distintas é dado por

$$E = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \sum_{k=j+1}^{N} \sum_{a=1}^{M} \sum_{b=a+1}^{M} \sum_{c=b+1}^{M} 1$$

Esta primeira solução tem complexidade $O(N^3M^3)$. Uma segunda solução é observar que, dados N alimentos, é possível escolher 3 deles de C_N^3 maneiras distintas, onde

$$C_N^3 = \binom{N}{3} = \frac{N!}{3!(N-3)!} = \frac{N(N-1)(N-2)}{6}$$

Assim, a resposta E para o problema será dada por

$$E = C_N^3 \times C_M^3,$$

de modo que esta segunda solução tem complexidade $O(1)$.

B. Embalando brinquedos

O problema pode ser resolvido por uma estratégia gulosa. Inicialmente, ordene tanto a sequência e_n da capacidades de carga das embalagens quanto a sequência b_m das massas dos brinquedos em ordem não-crescente. Após a ordenação, basta avaliar cada brinquedo b_j, em ordem, em relação a embalagem e_i de maior carga ainda não utilizada:

- se $b_j \leq e_i$, a embalagem pode ser usada. Neste caso, remova e_i da sequência e incremente em uma unidade o total de brinquedos embalados;
- caso contrário, como e_n está em ordem não-crescente, não há embalagem que comporte b_j: ignore este brinquedo.

Esta solução tem complexidade $O(N \log N)$, devido ao custo da ordenação.

C. Ações

Um algoritmo *naive*, que a cada consulta q_j avalia todos os elementos da sequência r_n, tem complexidade $O(NQ)$, obtendo veredito TLE.

É preciso, portanto, identificar o índice i que minimiza a função $d(x, q_j)$ de forma eficiente. Isto pode ser feito por meio da ordenação da sequência dos pares p_n, com $p_i = (r_i, i)$. Uma vez ordenada esta sequência, uma busca binária com o valor $(q_j, 0)$ vai retornar o primeiro par p_k tal que $r_k \geq q_j$, ou inexistente, caso nenhum valor da sequência satisfaça tal desigualdade.

Caso nenhum par seja identificado, a resposta para a pergunta será o segundo elemento do par que encerra a sequência ordenada p_n. Nos demais casos, se o par identificado foi $p_k = (r_k, i_k)$, deve-se avaliar também o par p_{k-1}, caso exista. A resposta da consulta será o segundo elemento do par cuja distância do primeiro elemento é a menor em relação a q_j.

Esta solução tem complexidade $O((Q + N) \log N)$.

D. Investimento de alto risco

Seja $m(p, \vec{j}, t)$ o montante obtido após t meses com um depósito inicial e mensal de p reais, com juros mensais dados pelo vetor $\vec{j}$. Como o vetor $\vec{j}$ é constante, para um período de tempo T fixo, a função $m(p, \vec{j}, T)$ é crescente em p. Em outras palavras, quanto maior o valor dos depósitos recorrentes, maior será o montante ao final do período.

Neste cenário, o problema pode ser resolvido por meio de uma busca binária na resposta, sendo o menor valor possível igual a 0 e o maior possível igual a $\min(M, 100.000, 00)$. Atente ao tratamento das casas decimais, as quais devem ser descartadas após o cálculo do juros percentual.

Esta solução tem complexidade $O(T \log M)$.

E. Duplas para a prova final

Há $16! = 20922789888000$ permutações possíveis dos alunos, o que inviabiliza uma verificação de cada uma delas. Contudo, se observarmos que muitas destas permutações levam a pareamentos equivalentes (por exemplo, as permutações 1234, 2143, 3421, etc levam ao pareamento {1, 2}, {3, 4}), podemos reduzir o número de pareamentos possíveis através destas simetrias.

Para explorarmos estas simetrias, devemos gerar os pareamentos com os elementos "à esquerda" do par em ordem crescente. Assim, fixando o número 1 na primeira posição, teremos $N - 1$ alunos possíveis para completar o par. Escolhido o primeiro par, o elemento à esquerda do segundo par deve ser o de menor identificador ainda não pareado. Daí teremos $N - 3$ escolhas possíveis para a segunda dupla.

Seguindo este raciocínio para $n = 16$, teremos $15!! = 15 \times 13 \times 11 \times \ldots \times 3 \times 1 = 2027025$ pares possíveis, quantia que permite a busca completa.

A busca pode ser acelerada usando-se a técnica da poda: armazenando em uma variável global o melhor resultado encontrado até o momento, se na escolha de uma dupla o δ correspondente for maior ou igual ao melhor resultado, a busca pode ser interrompida, melhorando o tempo de execução.

Existe uma solução de programação dinâmica com máscara de *bits* para este problema.