Maratona de Programação
Semana de Informática 2021

André G. Santos[1] Salles V.G. Magalhães[1]

[1]Departmento de Informática
Universidade Federal de Viçosa (UFV), Brazil

Semana de Informática, 2021

G. Investindo em ações  (17/27)[1]

Resumo

- Ler uma lista de valores D_i e P_i, dividendos e preços das ações
- Escrever o que dá maior rendimento, i.e., o de maior D_i/P_i

Solução

- Simplesmente iterar pela lista e reportar o de maior D_i/P_i
- Runtime error: buscar o menor P_i/D_i; note que D_i podia ser zero...
- Sugestão: evitar floats (mas neste problema provavelmente passa)

[1] quantos times passaram a questão e número de submissões

A. Promoções  (16/60)

Resumo

- Temos o preço de um produto e vários cupons de desconto (percentuais e em valor absoluto)
- Objetivo: encontrar cupom que deixa o preço o menor possível (apenas um cupom por compra)
- Cada cupom só pode ser utilizado se valor da compra estiver maior ou igual a um determinado mínimo.

Solução

- Solução simples. Basta testar todos cupons e, para cada um, ver qual o valor do produto comprado com ele.
- Utilizar valores em centavos para evitar erros de arredondamento (R$123.40 = 12340 centavos).
- Atenção com cálculos e arredondamentos.

H. Coffe break (14/16)

Resumo (adaptado de https://codeforces.com/problemset/problem/1433/B)

- Dada uma sequência de bandejas cheias ou vazias, quantos movimentos são necessários para que todas as cheias fiquem juntas?
- Bandejas cheias adjacentes podem ser movidas como um bloco
- Mover uma bandeja cheia (ou um bloco) para esquerda ou direita conta um movimento
- Exemplo: VVCVVCCV $\rightarrow$ VVCVCCVV $\rightarrow$ VVCCCVVV

Solução

- Cada movimento retira uma bandeja vazia entre bandejas cheias
- $\Rightarrow$ Contar número de bandejas vazias entre a primeira e a última cheia

J. Já se foi o disco voador  (10/36)

Resumo (adaptado de https://www.interviewbit.com/problems/colorful-number/)

- Uma placa é válida se os produtos dos dígitos de toda substring são distintos; ou seja, se não há substrings que geram mesmo produto
- Ex: 234 é válida pois $2 \times 3 \times 4$, 2×3, 3×4, 2, 3 e 4 são distintos
  123 não é válida pois $1 \times 2 \times 3$ e 2×3 geram o mesmo valor

Solução

- Placas de 1 dígito são válidas
- Placas com > 8 dígitos são inválidas *(não escapam dos 2 casos abaixo)*
- Placas com dígitos 0 e/ou 1 são inválidas
- Placas com dígitos repetidos são inválidas
- Placas com 2 a 8 são pequenas o suficiente para força bruta
  - Verificar se substring $a[i \ldots j], \forall i = 0 \ldots 7, \forall j = i \ldots 7$ repete produto[2]

[2] ou testar os casos: 24 ou 42 e 8; 23 ou 32 e 6; com risco de esquecer 83 e 46, etc

F. Atrapalhando a foto (7/26)

Resumo

- Detectar e apagar as "ilhas" de 1's numa matriz binária
- Ex.:

$$
\begin{array}{cc}
000100 & \\
110001 & \\
111100 & \\
100000 & \Rightarrow \\
000010 & \\
011010 & \\
000000 &
\end{array}
\quad
\begin{array}{c}
000100 \\
110001 \\
111100 \\
100000 \\
000000 \\
000000 \\
000000
\end{array}
$$

Solução

- Fazer *flood fill* a partir dos 1's das bordas
- Apagar os 1's não alcançados

E. Maratona de Programação Paralela  (5/20)

Resumo

- Várias instruções de um programa, algumas dependem de outras
- Sempre podemos tentar processar o máximo de instruções possíveis paralelamente (desde que todas as dependências delas tenham sido satisfeitas), pois há infinitos processadores

Solução

- Criar um grafo de dependências (instrução aponta para as que dependem dela)
- Instruções formam um DAG (caso contrário, resultado é -1)
- Encontrar caminho máximo no grafo (possivelmente desconectado)

I. Já chegou o disco voador (4/12)

Resumo

- Dadas as coordenads de vários pontos, contar quantos retângulos diferentes podemos formar (as tendo como os 4 vértices)
- Testar todas possibilidades de quadruplas de pontos é muito lento.

Solução:

- Ordenar pontos por coordenada X e agrupá-los.
- Processar os grupos de pontos em ordem crescente de X
- Criar map de map M, onde $M[y_1][y_2]$ indica quantos pares dos pontos processados até o momento possuem mesma coordenada x e coordenadas y sendo y_1 e y_2.
- Para cada par de pontos a, b com mesma coordenada X, os retangulos com vértices da direita iguais a a e b são os que possuem vértices da esquerda com x menor e y igual ao de a e b (ou seja, a quantidade que teremos desses retângulos é $M[a_y][b_y]$). Após acumular esses valores, conte a e b em M ($M[a_y][b_y]++$)

C. Capifrutas (3/30)

Resumo

- N ingredientes (N pequeno), cada um contém alguns nutrientes
- Encontrar o menor número de ingredientes que, juntos, possuem no mínimo K nutrientes distintos.
- Exemplo:
  - 0: 0001001 (ingrediente 0 possui nutrientes 3 e 6)
  - 1: 1001010 (ingrediente 1 possui nutrientes 0, 3 e 5)
  - 2: 1001001 (ingrediente 2 possui nutrientes 0, 3 e 6)
  - Para K=4, se usarmos os ingredientes 0 e 1 teremos 4 nutrientes distintos

Solução

- Problema NP-Completo (Set Covering Problem)
- Codificar nutrientes em bits de long long.
- Testar todas combinações possíveis de ingredientes usando um for de 0 a 2^N
- Usar OR bit-a-bit e popcount para fazer união e contagem rápida.

B. Buscadecarro  (3/9)

Resumo

- Similar ao problema A . Porém, cupons podem ser aplicados múltiplas vezes (desde que valor atual da compra esteja pelo menos igual ao mínimo)

Solução

- Similar ao problema da mochila. Pode-se usar PD ou memoização.
- Exemplo de função recursiva: int custoMinimo(int valorAtual)
- Em cada chamada, teste cada um dos cupons e chame recursivamente (caso ele possa ser aplicado no preço atual e gere algum desconto).
- Note que um cupom pode ser utilizado novamente. Em cada chamada recursiva teste todos cupons possíveis.
- Cuidado que um desconto pode ser de 0 (exemplo: 1% de desconto em 0.50). Isso pode gerar recursão infinita.

D. Distranciamento capivaral 2.0  (0/0)

Resumo

- Dados vários segmentos (horizontais ou verticais), contar quantos polígonos eles formam.
- Interseção de segmentos é no máximo um ponto.
- Há muitos: não é possível testar todos pares para interseção

Solução

- Usar fórmula de Euler: $V + F = E + 2$. Só funciona para 1 componente conexo, mas pode ser facilmente adaptada para múltiplos componentes.
- Para cada aresta, calcular as interseções delas com outras (fácil, já que apenas verticai e horizontais)
- Assim, para cada aresta contamos quantas subarestas serão formadas devido a interseções.
- Guardar interseções em um set para contar vértices únicos.
- Usar índice para acelerar testes de interseção (ex: uniform grid: fácil e rápido – talvez também dê para resolver verificando intervalos)