Maratona de Programação
Semana de Informática 2018

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

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

Semana de Informática, 2018

1. Yes  (20)[1]

Resumo

- Foi dada uma lista fixa de 5 categorias, com 2 unidades cada
- Dada uma lista de N unidades, contar o total de cada categoria

Solução

- Basta ler, verificar de qual categoria e contar

[1] este valor indica quantos times passaram a questão, estão ordenadas por dificuldade

5. Gold, please  (18)

Resumo

- Dada a quantidade de balões de cada uma de 3 cores, descobrir quantos transformar, no mínimo, para ter todos da mesma cor

Solução

- Somar os dois menores valores
- Solução curta: somar tudo e subtrair o maior

```
cout << a + b + c - max(a,max(b,c))
```

7. Capiflix (17)

Resumo

- Entrada: número de temporadas T da série, número de episódios E de cada uma, duração D em minutos de cada um, tempo livre H em horas e mínimo de episódios M para ser interessante
- Decidir se deve assistir (se dá tempo e é interessante)

Solução

- Assistir se $(T \cdot E \cdot D \leq H \cdot 60)$ and $(T \cdot E \geq M)$

6. Capivarić (17)

Resumo

- Ler o nome de um jogador e informar se é croata ou islandês

Solução

- Verificar se as 2 últimas letras são ‘i’ ‘c’ ou as 3 últimas são ‘s’ ‘o’ ‘n’
- Obs.: ler com `getline`, pois `cin` e `scanf` lêem uma palavra por vez

8. Capivara viúva UFV-Florestal  (16)

Resumo

- Dada uma lista com vários nomes (de mesmo tamanho – mas a solução é a mesma se não fossem) onde 1 não tem par, qual o nome sem par?
- Ex.: A, A, B, C, C $\Longrightarrow$ B

Solução: ($\approx N^2$ *operações*)

- Ler e armazenar todos os nomes
- Para cada nome, procurar uma ocorrência sua entre os nomes seguintes
- Se encontrado, apagar esta ocorrência (ou flag para desconsiderá-la)
- Se não encontrado, ele é a resposta
- Obs: esta solução é muito lenta para o problema 9, com $N = 50.000.000$; soluções lineares, $O(N)$, são apresentadas mais à frente.

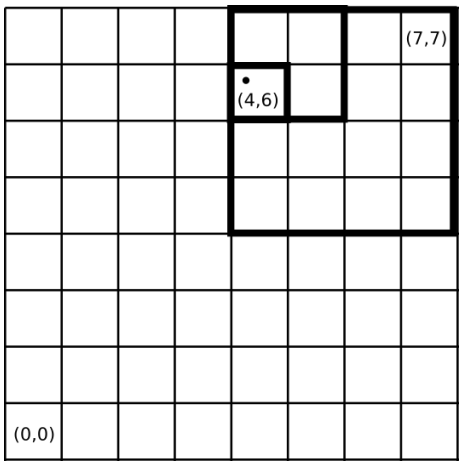
10. Busca capivárica (12)

Resumo

- Dada uma grade quadrada $N \times N$ (sendo N potência de 2) e uma sequência de dicas "quente" / "frio" que informam quadrantes na grade, identificar a célula buscada (ou dizer que as dicas não foram suficientes)

Solução

- Similar à busca binária: cada quadrante informado reduz pela metade os limites X e Y
- Considere como limites (a, b) e (c, d)
- Início: $a = b = 0$ e $c = d = n - 1$
- Para dica 1° quad.: $a = \lceil \frac{a+c}{2} \rceil$, $b = \lceil \frac{b+d}{2} \rceil$
  (os outros seguem regras semelhantes)
- Célula encontrada quando $a = c$ (e/ou $b = d$)
- Obs.: são necessárias $\log N$ dicas

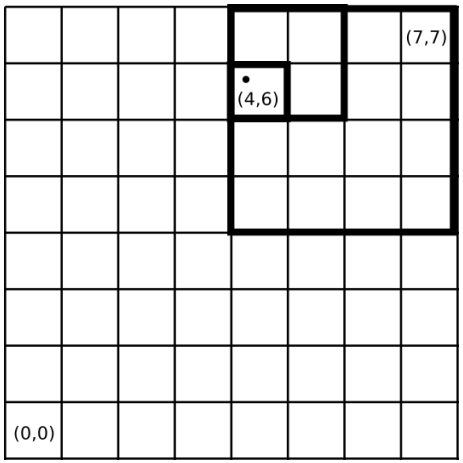


Ex.: grade 8×8, seq. de quadrantes $1°, 2°, 3° \Rightarrow (4, 6)$

10. Busca capivárica (12)

Solução alternativa

- A dimensão d da grade é potência de 2, então dar passos de tamanhos $d/2$, $d/4$, ...
- Início: $a = b = 0$ e $d = N$
- Para cada dica
  - $d = d/2$
  - Se 1° quad.: $a = a + d$, $b = b + d$
    (os outros aumentam só a, só b ou nenhum)
- Célula encontrada quando $d = 1$



Ex.: grade 8×8, seq. de quadrantes $1°, 2°, 3° \Rightarrow (4, 6)$

11. Falta d'água (6)

Resumo

- Há uma grade quadrada onde em cada célula chove 1 un. de água por seg.
- A água de uma célula escoa para uma célula vizinha (ou para fora da grade)
- Algumas células sao "lagos" (a água fica presa neles)
- Quanta água chega aos lagos por segundo? (após o fluxo se estabilizar)

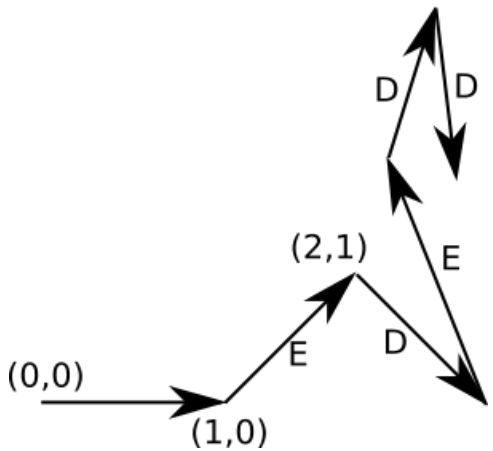
Solução

- Fluxo em um lago: chuva direta + quantidade de células que jogam água nele (direta e indiretamente)
- Ordenação topológica: o fluxo de uma célula só pode ser calculado após o fluxo em todas que fluem para ela já terem sido calculados
- Alternativamente: algoritmo (recursivo ou não) para calcular o tamanho de uma árvore (a direção contrária do fluxo forma as arestas)
- Obs.: existe solução (mais complexa) com basicamente 2 `for` e 1 `while`!

12. PCdaU (6)

Resumo

- Dado um caminho representado por uma lista de N pontos no plano, decidir se virou mais vezes para direita ou para esquerda para saber se é de "direita", de "esquerda" ou de "centro"



12. PCdaU (6)

Solução:

- Dados 3 pontos consecutivos P_i, P_{i+1}, P_{i+2}, o sinal do produto vetorial $(\overrightarrow{P_iP_{i+1}} \times \overrightarrow{P_iP_{i+2}})$ indica a direção: + direita, - esquerda
- $\Rightarrow$ Calcular $(\overrightarrow{P_iP_{i+1}} \times \overrightarrow{P_iP_{i+2}})$ para $i = 0 \ldots n-3$, e contar quantos foram positivos e quantos negativos

- Cálculo de um vetor $\overrightarrow{AB}$ pelas coordenadas: $(X_b - X_a, Y_b - Y_a)$
- Cálculo de produto vetorial $\overrightarrow{u} \times \overrightarrow{v}$ por determinante:
$$\overrightarrow{u} \times \overrightarrow{v} = \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} = u_x v_y - v_x u_y$$

Obs.: produto vetorial é definido em $\mathbb{R}^3$, mas podemos usar este cálculo simplificado em matriz 2×2 porque queremos apenas seu sinal

9. Capivara viúva UFV-Viçosa (4)

Resumo

- Dada uma lista com vários nomes (de mesmo tamanho – mas a solução é a mesma se não fossem) onde 1 não tem par, qual o nome sem par?
- Ex.: A, A, B, C, C $\implies$ B

Solução 1 (*10N operações*)

- Suponha que em vez de nome tivéssemos bits: 1, 0, 1, 0, 1 $\implies$ 1
- Como resolver de forma simples e eficiente se fosse apenas 1 bit? XOR (pares de 1s e 0s se cancelam)
- E se fossem dois bits? 10 10 11 01 01 $\implies$ o XOR também funcionaria.
- E se fossem caracteres? XOR dos bits deles!
- E strings? XOR dos caracteres correspondentes.

9. Capivara viúva UFV-Viçosa (4)

Exemplo:

String	Bits dos caracteres	
ABC	01000001|01000010|01000011	
ADC	01000001|01000100|01000010	
CBD	01000011|01000010|01000100	
CBD	01000011|01000010|01000100	
ABC	01000001|01000010|01000011	
XOR	01000001|01000100|01000010	$\Rightarrow$ ADC

Obs.: em C/C++ temos o operador ^ (xor bitwise): `a ^ b`

9. Capivara viúva UFV-Viçosa  (4)

Solução 2 (*10N incrementos + até 10×26 mod*)

- Contar ocorrências de A, B, ... Z em cada posição
- Imprimir a que aparecer um número ímpar de vezes

Solução 3 (*10N operações + até N/2 mod*)

- Criar um hash perfeito para mapear string (C++: map, unordered_map)
- Imprimir a que aparecer um número ímpar de vezes

2. No  (4)

Resumo

- Dada um conjunto de unidades inimigas, e um lista de unidades, cada uma sendo boa contra um subconjunto de unidades, decidir o mínimo de unidades que é, coletivamente, boa contra todas as inimigas.

Solução $O(2^N)$

- Força bruta (busca exaustiva)
- Temos no máximo $N = 15$ unidades, são $2^{15} = 32768$ subconjuntos
- Pouco o suficiente para gerar todos, avaliar quais cobrem todos os inimigos e reportar o tamanho do menor subconjunto que cobre todos
- Obs.: trata-se do problema de dominância em grafos, um problema NP-Difícil, não há método guloso conhecido que o resolva

3. Food, please (1)

Resumo

- Existem N capivaras. A i-ésima delas só pode ser abatida no momento i na posição x_i (coordenadas 1D informadas na entrada).
- Com uma arma inicialmente na posição 0, que pode se mover 1 un. de distância por un. de tempo, qual o máximo de capivaras se consegue abater?
- Restrição: deve-se obrigatoriamente abater a última delas.

Solução $O(N^2)$

- Para abater a última, a arma deve apontar para x_N no momento N (isto será possível se e somente se $|x_N| \leq N$)
- Que capivara abater antes? Alguma capivara i com $|x_N - x_i| \leq N - i$ (após abater i em x_i no momento i, terá $N - i$ un. de tempo para se mover até x_N)
- Das opções disponíveis, deve-se escolher a que pode abater mais antes dela
- Seja T_i o máximo de capivaras abatidas de 1 a i, abatendo-se a i-ésima
- Recorrência:

$$T_i = \begin{cases} 0 & \text{se } |x_i| > i \\ 1 + \max_{j|j<i \wedge |x_i - x_j| \leq j-i}\{T_j\} & \text{caso contrário} \end{cases}$$

4. Wood, please (1)

Resumo

- N tipos de unidades: i-ésima tem poder a_i e consome c_i, m_i, o_i dos recursos
- Qual o poder máximo pode ser obtido com C, M, O dos recursos?

Solução *(problema da mochila)*

- Quantos i adquirir com c, m, o de recursos? De 0 a $\min\{\lfloor \frac{c}{c_i} \rfloor, \lfloor \frac{m}{m_i} \rfloor, \lfloor \frac{o}{o_i} \rfloor\}$
- Se adquirir t, ganha $t\,a_i$ de poder e gasta $t\,c_i, t\,m_i, t\,o_i$ de recursos, sobrando $c - tc_i, m - tm_i, o - to_i$ para adquirir os demais
- Escolher t que maximiza, i.e., ta_i + máximo poder possível com o restante
- Seja T^i_{cmo} o máximo poder possível com unidades $1 \ldots i$ e recursos c, m, o
- Se $i = 0$, então $T^i_{cmo} = 0$ (caso base, sem unidades). Senão
- $T^i_{cmo} = \max\{ta_i + T^{i-1}_{c-tc_i, m-tm_i, o-to_i}\}$, para $0 \leq t \leq \min\{\lfloor \frac{c}{c_i} \rfloor, \lfloor \frac{m}{m_i} \rfloor, \lfloor \frac{o}{o_i} \rfloor\}$
- Obs.: cuidado com unidades 0 0 0 0