

Maratona de Programação

Semana de Informática 2020

André G. Santos¹ Salles V.G. Magalhães¹

¹Departamento de Informática
Universidade Federal de Viçosa (UFV), Brazil

Semana de Informática, 2020

B. VOIP (19/25)¹

Resumo


- Ler duas letras, A e B, de “VOIP”
- Calcular a distância (circular) de A até B

Solução

- Seja valor de V, O, I, P igual a 0, 1, 2, 3 respectivamente
- `demora = valor(B) - valor(A);`
`if (demora<=0) demora +=4;`

¹ quantos times passaram a questão e número de submissões

F. Por que a capivara atravessou a lagoa?

 (19/32)

Resumo

- Dada uma matriz quadrada e a posição de N jacarés, encontrar uma linha segura (que não tem jacarés)

Solução 1 $O(L^2)$

- Marcar os jacarés numa matriz $L \times L$
- Percorrer a matriz linha por linha até achar uma linha segura

Solução 2 $O(N + L)$

- Vetor com L booleanos, inicialmente seguros
- Para cada jacaré, marcar no vetor o índice da linha como inseguro
- Percorrer o vetor até achar um índice seguro

A. Senha Super Secreta (19/47)

Resumo

- Ler dois valores A e B
- Escrever qualquer valor entre 0 e 100 que não seja A , B , $A + B$, $A - B$, $B - A$

Soluções Força Bruta

- Iterar i de 1 a 100 até achar i diferente dos valores citados
- Sortear i aleatoriamente até achar i diferente dos valores citados

Solução com dois condicionais

- Se $A \neq B$ escrever 0
- Se $A = B$ e $A \neq 1$ escrever 1
- Senão escrever 3
- `cout << (A!=B?0:A!=1?1:3) << endl;`

N. Capivara aravipac está presente (18/37)

Resumo

- Verificar se um string é quase palíndromo
- No caso, se o reverso difere no máximo em uma letra

Solução

- Ler o string S
- Criar T apenas com as letras de S , todas em minúsculo
- Contar quantos $T[i] \neq T[N - i - 1]$, sendo N o tamanho de T

Resumo

- Ler um sequência de linhas e verificar se alguma contém capivara, capivaro, capivarista ou capivaristo

Solução (em C++)

```
while(getline(cin,linha)) {
    string t = ... //mudar linha para minusculo
    n = t.size();
    if (t.find("capivara")<n || t.find("capivaro")<n ...) \{
        cout << "YES\n";
        return 0;
    }
}
cout << "NO\n";
```

P. 水豚语 (10/19)

Resumo

- Ler número de consoantes C , vogais A , e caracteres P permitidos
- Calcular o número de palavras diferentes

Solução

- $\sum_{i=1}^P (4AC)^i$
- **Overflow!** fazer *mod* 1000007 em **todo** soma e multiplicação
- Obs.: em Python pode fazer *mod* apenas no fim

Curiosidade: 水豚(Shuǐ tún) = capivara, 语(yǔ) = idioma

E. Eu sabia essa com maçãs!

📌 (7/75)

Resumo

- Dados N, A, B verificar se $N! = A! \times B!$
- Dica: o enunciado fala que para $N, A, B > 1$ só há solução se $N = 10$

Solução direta: **WA, TLE**

- Calcular $N!, A!, B!$ e verificar se $N! = A!B!$
- **overflow**: $13!$ não cabe em `int`, $21!$ não cabe em `long long int`

Solução

- **Cuidado!** Se $N = 0$ ou 1 , YES se A, B forem 0 ou 1
- Se A for 0 ou 1 , YES se $N = B$ (e vice-versa)
- Se $N = 10$, YES se $A = 6, B = 7$ ou $B = 7, A = 6$
 - alternativamente, se $N = 10$ calcular $N!, A!, B!$ e verificar se $N! = A!B!$
- Senão, NO

D. Distanciamento capivaral (6/63)

Resumo

- N capivaras, lagoa de dimensões $L \times C$ (em m)
- Máximo de D por m^2 dentro da lagoa e F por m na margem
- Verificar se é possível respeitar o distanciamento
- Obs.: 1 dentro não interessa D e 1 fora não interessa F

Solução

- Máximo dentro: $M_D = \max(1; L \times C \times D)$
- Máximo fora: $M_F = \max(1; 2 \times (L + C) \times F)$
- YES se $N \leq M_D + M_F$, e NO caso contrário

H. Capivaras querem doce (de novo)!

📌 (6/48)

Resumo

- 2 latas de doce de leite com pesos A e B
- Jacaré pode comer A ou B e deixar $1/2$, $1/3$, $1/5$ (se divisível)
- Quantas vezes no mínimo para chegar em $A = B$?

Solução 1: $O(\log(N))$

- Fatorar os dois números em fatores primos
- Tem solução sse os fatores (e potências) em comum forem iguais
- Resposta = soma das potências “a mais” de cada número fatorado
- Exemplo: $168 = 2^3 \times 3 \times 7$ e $14 = 2^1 \times 7$. Resposta será $(3 - 1) + (1 - 0) = 3$ (diferenças entre potências do 2 e do 3).
- Na verdade, nem precisamos fatorar o número completamente. Só precisamos dos fatores primos 2, 3 e 5. O problema terá solução sse o que “sobrar” nos dois números (após extrair tais fatores) for igual.

H. Capivaras querem doce (de novo)!

📌 (6/48)

Solução 2: PD + memoization (TLE)

$$P(A, B) = \begin{cases} 0 & A = B \\ P(B, A) & A < B \\ 1 + \min\{P(A/5, B), P(A/3, B), P(A/2, B), \infty\} & * \end{cases}$$

* em cada caso, chamar somente se A for divisível

Solução 3: PD + memoization + corte

$$P(A, B, p) : \begin{cases} A = B & \text{atualizar } best \text{ se } p < best \\ A < B & P(B, A, p) \\ A > B \text{ and } p < best & \min\{P(A/5, B, p + 1), \dots, \infty\} * \end{cases}$$

G. Pq a capivara atravessou a lagoa de novo?

4/14

Resumo

- N capivaras, cada uma demora T_i para atravessar a lagoa
- No máximo 2 simultaneamente na lagoa
- É possível todas atravessarem no tempo limite K ?
- \Rightarrow é possível dividi-las em 2 grupos com $\sum T_i \leq K$ em cada?

Solução 1 $O(NK)$

- Problema da mochila com capacidade K e itens de peso = valor = T_i
- Verificar se os que ficam de fora tem soma dos tempos $\leq K$

Solução 2 $O(NK)$

- Max Subset Sum do conjunto $\{T_1, T_2, \dots, T_n\}$ de soma $\leq K$
- Verificar se os que ficam de fora tem soma dos tempos $\leq K$

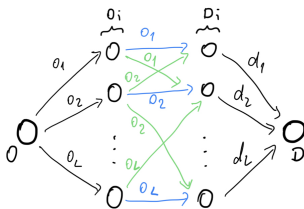
J. Balanceamento de capivaras nas lagoas (1/3)

Resumo

- L lagoas, cada uma com o_i capivaras e deve ter d_i capivaras
- T transportes (a, b) : capivaras da lagoa a podem ir para lagoa b

Solução

- Nó O (origem) e nó D (destino)
- Para cada lagoa i , nó O_i e nó D_i
- arco azul (O_i, D_i) : permanece em i
- arco verde (O_a, D_b) : transporte (a, b)
- Fluxo máximo de O até D



C. Capivid-20 (1/9)

Resumo

- Capivara infectada pelo jacaré com capivid-20 na hora 0 posição $(0, 0)$
- Se na hora t esteve em (i, j) , na hora $t + 1$ esteve em uma adjacente: $(i - 1, j), (i, j - 1), (i + 1, j), (i, j + 1)$ – com igual probabilidade cada
- Qual a probabilidade de cada capivara ter se encontrado com a infectada, dadas até 3 posições e horários de cada uma?

Solução

- Pré-calcular a probabilidade de cada posição e horário
- $$P(i, j, t + 1) = \frac{P(i-1, j, t) + P(i, j-1, t) + P(i+1, j, t) + P(i, j+1, t)}{4}$$
- **Obs:** não necessário, mas possível acelerar pois há posições visitadas apenas em horas pares e as demais somente em ímpares; ou fazer *floodfill* do $P(0, 0, 0) = 1$
- Para 1 posição (X, Y) e hora H , basta escrever $P(X, Y, H)$
- Para 2 e 3 usar inclusão/exclusão (para 2: $P_{a \cup b} = P_a + P_b - P_a P_b$)
- Ou separar os casos ($P_{a \cup b} = P_a(1 - P_b) + (1 - P_a)P_b + P_a P_b$)

I. The OC: uma capivara na mata do Paraíso (-/17)

Resumo

- Capivara deve ir da mata do Paraíso à UFV fazer prova de OC
- Possui pontos de ataque K e vida V que perde nos trechos
- Encontrar o caminho mais curto que não deixa nenhum deles negativo

Solução Bellman-Ford $O(\dots)$

- Não basta manter o caminho mínimo pois ele pode ficar sem pontos
- Cada nó tem uma matriz $K \times V$ de distâncias: $[k, v]$ guarda a distância mínima da origem ao nó deixando k e v de ataque e vida

Solução Dijkstra $O(\dots)$ TLE bem provável

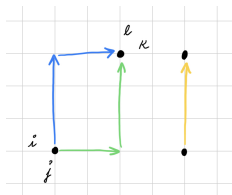
- Cada nó tem um conjunto de labels $[d, v, k]$: distância da origem ao nó e pontos de ataque e vida restantes de cada caminho conhecido
- O nó (e caminho) escolhido para expandir é o de menor d , desempate por maior $k + v$
- um label $[d, k, v]$ pode ser removido se há outro com $< d$ e $> k, > v$

Resumo

- Mapa retangular (grid com L linhas e C colunas)
- M compromissos do tipo capivara deve ir de (i, j) para (k, l)
 - Ela vai ao compromisso no caminho mínimo por no máximo 2 ruas
- É possível orientar as linhas e colunas atendendo todos compromissos?

Solução *(termina no próximo slide)*

- Pela restrição apresentada, para ir de (i, j) a (k, l) há duas opções: pela linha i coluna l (verde) ou pela coluna j linha k (azul)
- Obs.: se casa e compromisso estão na mesma linha ou coluna, só há uma opção (amarelo)



Solução 2-SAT $O(\dots)$

- Seja $L_i = true$ se a linha i deve ser orientada \rightarrow , e $false$ se \leftarrow
- Seja $C_i = true$ se coluna i deve ser orientada \uparrow , e $false$ se \downarrow
- Para uma capivara ir de (i, j) para (k, l) :
 - Se $i = k$ e $j < l \implies L_i = true$
 - Se $i = k$ e $j > l \implies L_i = false$
 - Se $i < k$ e $j < l \implies (L_i = true \wedge C_l = true) \vee (C_j = true \wedge L_k = true)$
 - ...
- $(L_i \wedge C_l) \vee (C_j \wedge L_k) \iff (L_i \vee C_j) \wedge (L_i \vee L_k) \wedge (C_l \vee C_j) \wedge (C_l \vee L_k)$
- Capivaras com $i = k$ ou $j = l$ atribuem $true/false$ a uma variável
- Capivaras com $i \neq k$ e $j \neq l$ criam 4 cláusulas de 2 variáveis
- Montar a expressão completa e resolver o 2-SAT (obs.: 2-SAT $\in \mathcal{P}$)

L. Aterrando a lagoa (-/-)

Resumo

- Uma lagoa, em forma de matriz, com jacarés em algumas células
- A lagoa pode ter colunas aterradas pela direita e/ou esquerda
- Os aterramentos podem deixar linhas livres de jacarés
- Quantas combinações diferentes de linhas livres são possíveis?

Observações

- A entrada codifica colunas em números binários: jacaré = 1 e livre = 0
- **OU** bit-a-bit de colunas resulta em um número com 0 em linhas livres
- Qtos núm. diferentes podem ser criados por OU de colunas consecutivas?
- A lagoa tem no máximo 20 linhas, já que os valores vão até 1 milhão

Solução 1 $O(L^2)$ ou $O(L^3)$ TLE

- Testar todas possibilidades de aterramento e anotar os números possíveis
- Cada possibilidade é gerada fazendo-se OU das colunas não aterradas
- Vetor de booleanos (com 2^{20} posições) para marcar os números gerados

L. Aterrando a lagoa (-/-)

Solução 2 – PD $O(cL) + O(\log C) = O(L)$ $c \leq 20, C \leq 2^{20}$

- Como na anterior, contar os números distintos da parte não aterrada
- Seja: $C[l..J] = C[l] \text{ OR } C[l+1] \text{ OR } \dots \text{ OR } C[J]$, onde $C[i]$ é o número binário representando a i -ésima coluna da lagoa
- Então, queremos calcular todos $C[l..J]$ distintos possíveis de serem gerados
- Observação: cada número termina em algum J
- For J de 1 a L calcular quais números binários terminam em cada posição J
- Os números que terminam na posição J são: – $O(cL)$
 $\{C[J]\} + \{K \text{ OR } C[J], \text{ para cada número } K \text{ que termina na posição } J - 1\}$.
- Em cada passo, guardar em um set os números obtidos no passo – $O(\log C)$

L. Aterrando a lagoa (-/-)

Solução 3 $O(L) + O(cL) + O(\log C) = O(L)$ $c \approx 20, C \leq 2^{20}$

- Aterros na direita $O(L)$
 - Seja $T[i]$ o **OU** das colunas $C[0]$ a $C[i]$: $T[0]=C[0]$ e $T[i]=T[i-1] \mid C[i] \forall i$
 - Todos valores $T[i]$ são combinações possíveis (aterro de $C[i] \dots C[n]$)
- Aterros na esquerda $O(cL)$
 - Mesmo processo, iniciando de cada $C[j]$: $\forall j > 0, A=C[j]$ e $A=A \mid C[i] \forall i$
 - Todos A são comb. possíveis (aterros $C[0] \dots C[j-1]$ e $C[i] \dots C[n]$)
 - *eficiência: parar quando $T[i]$ cobre todas linhas de A ($T[i] \& A = T[i]$)
- Os valores são até 1 milhão, então a lagoa tem no máximo 20 linhas
 - Os aterros na direita (com $|C[i]$) geram no máximo 20 combinações
 - Também geram no máximo 20 para cada aterro da esquerda ($A = C[j]$)
 - Então a maioria (*efic.) para em poucas iterações ($c \pm$ amortizado)
 - Guardar todos gerados nas duas etapas em um set

O. Transportando Doce de Leite na Nlogônia



(-/1)

Resumo

- Problema de caminho mínimo, com algumas arestas de distância 0
- Se dois túneis (segmentos) se cruzam em uma coordenada inteira, pode-se “pular” de um para outro em tal coordenada (como em cruzamento de estradas, onde um carro pode trocar de estrada).

Solução 1 $O(N^2 + N \log N) = O(N^2)$

- Encontrar as interseções testando todos pares de túneis e vendo quais delas são inteiras – $O(N^2)$
- Criar o grafo e encontrar o caminho pelo método de Dijkstra
- Mesmo algoritmos mais eficientes de detecção de interseções (como *sweep-line*) seriam $O(N^2)$, pois pode haver N^2 interseções (ex.: grade com metade dos túneis na vertical e metade na horizontal)
- **TLE** para qualquer método que tenta encontrar todas interseções

O. Transportando Doce de Leite na Nlogônia (-/1)

Solução 2 $O(N \log N)$

- Note que estamos interessados apenas nas interseções inteiras
- As coord. vão apenas até 1000, então no pior caso cada segmento terá 1001 pontos de coord. inteiras (ex: segmento 0,0 - 1000,1000)
- Solução: dividir cada segmento em vários segmentos com coordenadas inteiras. Ex: o segmento 0,0 - 3.2, 3.2 será dividido em: 0,0 - 1,1; 1,1 - 2,2; 2,2 - 3,3; 3,3 - 3.2, 3.2
- A seguir, caminho mínimo em um grafo considerando vértices com mesmas coordenadas sendo os mesmos vértices (pode-se utilizar um map para mapear cada coordenada no id de um vértice)
- Ou seja, não é preciso “procurar” pelas interseções
- No pior caso, são 1000 vezes mais segmentos do que originalmente há na entrada; além disso, o número de coordenadas inteiras é no máximo $1001 \times 1001 \Rightarrow$ o grafo tem tamanho cN , c constante