AtCoder

AtCoder Beginner Contest 044: *Upsolving*

Prof. Edson Alves - UnB/FGA

2020

Sumário

A - Tak and Hotels (ABC Edit)

Problema

There is a hotel with the following accommodation fee:

- X yen (the currency of Japan) per night, for the first K nights
- Y yen per night, for the $(K+1)$-th and subsequent nights

Tak is staying at this hotel for N consecutive nights. Find his total accommodation fee.

Entrada e saída

Constraints

- $1 \leq N, K \leq 10000$
- $1 \leq Y < X \leq 10000$
- N, K, X, Y are integers.

Input

Input is given from Standard Input in the following format:

N
K
X
Y

Output

Print Tak's total accommodation fee.

Exemplo de entradas e saídas

Entrada	Saída
5 3 10000 9000	48000
2 3 10000 9000	20000

Solução

- Há dois casos a serem tratatos
- Se $N \leq K$, Tak pagará X por dia, de modo que sua estadia custará, no total, NX ienes
- Se $N > K$, os primeiros K dias custarão X ienes cada, e os $N - K$ dias restantes custarão Y ienes
- Assim o total da estadia será igual a $KX + (N - K)Y$
- Usando as funções `min()` e `max()` do C++ é possível unir ambos casos em uma única expressão:

$$T = X \times \min\{N, K\} + Y \times \max\{0, N - K\}$$

- Esta solução tem complexidade $O(1)$

Solução $O(1)$

```cpp
#include <bits/stdc++.h>

using namespace std;

int main()
{
    int N, K, X, Y;
    cin >> N >> K >> X >> Y;

    auto ans = X*min(N, K) + Y*max(0, N - K);

    cout << ans << '\n';

    return 0;
}
```

B – Beautiful Strings

Problema

Let w be a string consisting of lowercase letters. We will call w beautiful if the following condition is satisfied:

- Each lowercase letter of the English alphabet occurs even number of times in w.

You are given the string w. Determine if w is beautiful.

Entrada e saída

Constraints

- $1 \leq |w| \leq 100$
- w consists of lowercase letters (`'a'`-`'z'`)

Input

Input is given from Standard Input in the following format:

w

Output

Print `'Yes'` if w is beautiful. Print `'No'` otherwise.

Exemplo de entradas e saídas

Entrada	Saída
abaccaba	Yes
hthth	No

Solução

- A solução do problema pode ser obtida por meio da avaliação do histograma h_w de w
- Para um caractere c, $h_w(c)$ retorna o número de ocorrências de c em w
- Assim, a resposta será `'Yes'` se, para qualquer caractere minúsculo c, $h_w(c)$ é par
- Como apenas a paridade de $h_w(c)$ é relevante para o resultado, este valor pode ser atualizado pela operação binária XOR (ou exclusivo)
- A operação $n = n$ XOR 1 troca a paridade de n (de par para ímpar e de ímpar para par)
- Esta solução tem complexidade $O(|w|)$

Solução $O(|w|)$

```cpp
#include <bits/stdc++.h>

using namespace std;

int main()
{
    string w;
    cin >> w;

    vector<int> hw(26, 0);

    for (auto c : w)
        hw[c - 'a'] ^= 1;

    auto ans = accumulate(hw.begin(), hw.end(), 0)
        ? "No" : "Yes";

    cout << ans << '\n';

    return 0;
}
```

C - Tak and Cards

Problema

Tak has N cards. On the i-th ($1 \leq i \leq N$) card is written an integer x_i. He is selecting one or more cards from these N cards, so that the average of the integers written on the selected cards is exactly A. In how many ways can he make his selection?

Entrada e saída

Constraints

- $1 \leq N \leq 50$
- $1 \leq A \leq 50$
- $1 \leq x_i \leq 50$
- N, A, x_i are integers.

Partial Score

- 200 points will be awarded for passing the test set satisfying ($1 \leq N \leq 16$).

Entrada e saída

Input

Input is given from Standard Input in the following format:

N A
x_1 x_2 ... x_N

Output

Print the number of ways to select cards such that the average of the written integers is exactly A.

Exemplo de entradas e saídas

Entrada	Saída
4 8 7 9 8 9	5
3 8 6 6 9	0
8 5 3 6 2 8 7 6 5 9	19

Solução

- A média aritmética de k elementos $x_1, x_2, \ldots, x_k$ é dada por

$$M = \frac{x_1 + x_2 + \ldots + x_k}{k}$$

- O problema consiste em identificar, dentre todos os subconjuntos de $X = \{x_1, x_2, \ldots, x_N\}$, todos cujo média dos elementos é igual a A
- Um conjunto com N elementos tem um total de 2^N subconjuntos distintos
- Como $N \leq 50$, a verificação de todos estes subconjuntos resulta em um veredito TLE
- Contudo, há uma pontuação parcial para $N \leq 16$, limite para qual esta abordagem é válida

Solução de força bruta com pontuação parcial

```cpp
#include <bits/stdc++.h>

int solve(int N, int A, const std::vector<int>& xs)
{
    int ans = 0;

    for (long long s = 1; s < (1LL << N); ++s)
    {
        int sum = 0, m = 0;

        for (long long i = 0; i < N; ++i)
            if (s & (1LL << i))
                sum += xs[i], ++m;

        if (sum % m == 0 and sum / m == A)
            ++ans;
    }

    return ans;
}

```

Solução de força bruta com pontuação parcial

```
22 int main()
23 {
24     int N, A;
25     std::cin >> N >> A;
26
27     std::vector<int> xs(N);
28
29     for (int i = 0; i < N; ++i)
30         std::cin >> xs[i];
31
32     auto ans = solve(N, A, xs);
33
34     std::cout << ans << '\n';
35
36     return 0;
37 }
```

Solução

- O problema pode ser resolvido por meio de um algoritmo de programação dinâmica
- Considere que os elementos x_i sejam indexados de 0 a $N-1$
- Seja $dp(i, k, s)$ o número de maneiras distintas de, dentre os elementos $x_i, x_{i+1}, \ldots, x_{N-1}$, escolher k deles tal que sua soma seja igual a s
- A operação de divisão pode ser evitada:

$$\frac{x_1 + x_2 + \ldots + x_k}{k} = A$$

equivale a

$$x_1 + x_2 + \ldots + x_k = kA$$

- Assim, a solução do problema será a soma

$$S = \sum_{k=1}^{N} dp(0, k, kA)$$

Solução

- O caso base acontece quando $i = N$: como não há mais elementos a serem escolhidos, $dp(N, 0, 0) = 1$ e $dp(N, k, s) = 0$, se $k > 0$ ou $s > 0$
- As transições possíveis consistem em escolher, ou não, o elemento x_i
- Assim, se $m > 0$ e $x_i \leq s$,

$$dp(i, k, s) = dp(i + 1, k - 1, s - x_i) + dp(i + 1, k, s)$$

- Se $x_i > s$, então

$$dp(i, k, s) = dp(i + 1, k, s)$$

- Há $O(N^3 X)$ estados distintos, onde $X = \max\{x_1, x_2, \ldots, x_N\}$, e as transições são feitas em $O(1)$
- Portanto esta solução tem complexidade $O(N^3 X)$

Solução $O(N^3X)$

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

const int MAX { 55 };

ll st[MAX][MAX][MAX*MAX];

ll dp(int i, int m, int sum, int N, const vector<int>& xs)
{
    if (i == N)
        return (sum == 0 and m == 0 ? 1 : 0);

    if (st[i][m][sum] != -1)
        return st[i][m][sum];

    auto res = dp(i + 1, m, sum, N, xs);

    if (m and xs[i] <= sum)
        res += dp(i + 1, m - 1, sum - xs[i], N, xs);
```

Solução $O(N^3X)$

```
22
23      st[i][m][sum] = res;
24
25      return res;
26  }
27
28  long long solve(int N, int A, const vector<int>& xs)
29  {
30      memset(st, -1, sizeof st);
31
32      long long ans = 0;
33
34      for (int i = 1; i <= N; ++i)
35          ans += dp(0, i, i*A, N, xs);
36
37      return ans;
38  }
39
40  int main()
41  {
42      ios::sync_with_stdio(false);
```

Solução $O(N^3X)$

```
43
44      int N, A;
45      cin >> N >> A;
46
47      vector<int> xs(N);
48
49      for (int i = 0; i < N; ++i)
50          cin >> xs[i];
51
52      auto ans = solve(N, A, xs);
53
54      cout << ans << '\n';
55
56      return 0;
57  }
```

Solução

- É possível reduzir a complexidade para $O(N^2X)$ se for utilizado um estado diferente para caracterizar o problema
- Esta caracterização, contudo, envolve um certo engenho, o qual não é óbvio à primeira vista
- Inicialmente, considere uma sequência $y = \{y_1, y_2, \ldots, y_N\}$ tal que $y_i = A$, para todo $i = 1, 2, \ldots, N$
- A média aritmética de todos os elementos da sequência y é igual a A, e a soma de todos os elementos é igual a $S = NA$

Solução

- A ideia central é que, para cada elemento x_i da sequência $\{x_1, x_2, \ldots, x_N\}$, há duas opções: não escolher x_i ou substituir y_i na soma S por x_i
- Após considerados todos os elementos x_i, o número de maneiras de obter a soma NA, consideradas as duas opções, será a resposta do problema, mais uma unidade
- Esta unidade extra vem do fato de que, se nenhum x_i for escolhido, a soma resultante será NA
- Assim, o novo estado dispensa a dimensão que rastreava o número de elementos já somados

Solução

- Seja $st(i, s)$ o número de maneiras distintas de se obter a soma s considerando-se os elementos $x_i, x_{i+1}, \ldots, x_{N-1}$
- O caso base acontece quando $i = N$: $st(i, s) = 1$, se $s = NA$, e $st(i, s) = 0$, caso contrário
- As transições correspondem às duas opções já citadas:

$$st(i, s) = st(i + 1, s - A + x_i) + st(i + 1, s)$$

- O valor máximo da segunda dimensão será $2NX$, onde $X = \max\{x_1, x_2, \ldots, x_N\}$
- Há $O(N^2X)$ estados distintos, com transições em $O(1)$, de modo que a solução tem complexidade $O(N^2X)$

Solução $O(N^2X)$

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

const int MAX { 55 };
ll st[MAX][2*MAX*MAX];

ll dp(int i, int sum, int N, int A, const vector<int>& xs)
{
    if (i == N)
        return (sum == N*A ? 1 : 0);

    if (st[i][sum] != -1)
        return st[i][sum];

    st[i][sum] = dp(i + 1, sum, N, A, xs) +
                 dp(i + 1, sum - A + xs[i], N, A, xs);

    return st[i][sum];
}
```

Solução $O(N^2X)$

```
22
23 ll solve(int N, int A, const vector<int>& xs)
24 {
25     memset(st, -1, sizeof st);
26     return dp(0, N*A, N, A, xs) - 1;
27 }
28
29 int main()
30 {
31     int N, A;
32     cin >> N >> A;
33
34     vector<int> xs(N);
35
36     for (int i = 0; i < N; ++i)
37         cin >> xs[i];
38
39     cout << solve(N, A, xs) << '\n';
40
41     return 0;
42 }
```

D – Digit Sum

Problema

For integers b ($b \geq 2$) and n ($n \geq 1$), let the function $f(b, n)$ be defined as follows:

- $f(b, n) = n$, when $n < b$
- $f(b, n) = f(b, \text{floor}(n/b)) + (n \bmod b)$, when $n \geq b$

Here, $\text{floor}(n/b)$ denotes the largest integer not exceeding n/b, and $n \bmod b$ denotes the remainder of n divided by b.

Problema

Less formally, $f(b, n)$ is equal to the sum of the digits of n written in base b. For example, the following hold:

- $f(10, 87654) = 8 + 7 + 6 + 5 + 4 = 30$
- $f(100, 87654) = 8 + 76 + 54 = 138$

You are given integers n and s. Determine if there exists an integer b ($b \geq 2$) such that $f(b, n) = s$. If the answer is positive, also find the smallest such b.

Entrada e saída

Constraints

- $1 \leq n \leq 10^{11}$
- $1 \leq s \leq 10^{11}$
- n, s are integers.

Input

Input is given from Standard Input in the following format:

n
s

Output

If there exists an integer b ($b \geq 2$) such that $f(b, n) = s$, print the smallest such b. If such b does not exist, print `'-1'` instead.

Exemplo de entradas e saídas

Entrada	Saída
87654 30	10
87654 138	100
87654 45678	-1

Solução

- Primeiramente devem ser observados dois casos excepcionais, os quais serão tratados à parte
- Observe que, se $S > N$, não há solução para o problema
- Isto porque a maior soma possível ocorre quando $b > N$: neste caso, a soma será o próprio N
- Assim, se $S = N$, a resposta deve ser $b = N + 1$
- Estes dois casos delimitam a busca das bases b ao intervalo $[2, N]$
- Como $N \leq 10^{11}$, uma solução de busca completa que avalia todo este intervalo leva a um veredito TLE

Solução

- Seja

$$N = c_0 + c_1 b + c_2 b^2 + \ldots + c_k b^k$$

  a representação de N na base b

- O problema consiste em determinar, se existir, o menor b tal que

$$s = c_0 + c_1 + c_2 + \ldots + c_k$$

- A diferença entre estas duas igualdades é

$$N - s = c_1(b-1) + c_2(b^2-1) + \ldots + c_k(b^k-1)$$

- Assim, $(b-1)$ é um divisor da diferença $N - s$

Solução

- Esta importante propriedade reduz a busca das bases aos divisores de $N - s$
- Sejam a, b dois inteiros tais que b divide a
- Logo existe um inteiro c tal que $a = bc$
- Esta expressão mostra que c também é um divisor de a, pois $a = cb$
- É possível demostrar, por contradição, que ou $b \leq \sqrt{a}$ ou $c \leq \sqrt{a}$

Solução

- Assim, o número de divisores de a é $O(\sqrt{a})$
- A rotina $sum\_digits(x, b)$, que computa a soma dos dígitos de x na base b, tem complexidade $O(\log x)$
- Portanto, uma solução que avalie todas as bases $b = d + 1$, onde d é um divisor de $N - s$, tem complexidade $O(\sqrt{N} \log N)$

Solução $O(\sqrt{N} \log N)$

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

const ll oo { 1LL << 62 };

ll sum_digits(ll x, ll base)
{
    ll res = 0;

    while (x)
    {
        res += (x % base);
        x /= base;
    }

    return res;
}

```

Solução $O(\sqrt{N} \log N)$

```
21 ll solve(ll N, ll S)
22 {
23     if (S > N)
24         return -1;
25
26     if (S == N)
27         return N + 1;
28
29     // (b - 1) tem que dividir a diferença N - S
30     auto diff = N - S, ans = oo;
31
32     for (ll d = 1; d * d <= diff; ++d)
33     {
34         if (diff % d == 0)
35         {
36             auto base = d + 1;
37
38             if (sum_digits(N, base) == S)
39                 ans = min(ans, base);
40
41             auto k = diff / d;
```

Solução $O(\sqrt{N} \log N)$

```
43              if (k != d)
44              {
45                  base = k + 1;
46
47                  if (sum_digits(N, base) == S)
48                      ans = min(ans, base);
49              }
50          }
51      }
52
53      return ans == oo ? -1 : ans;
54  }
55
56  int main() {
57      ll N, S;
58      cin >> N >> S;
59
60      cout << solve(N, S) << '\n';
61
62      return 0;
63  }
```

Referências

1. AtCoder Beginner Contest 044 - Problem A: Tak and Hotels (ABC Edit)
2. AtCoder Beginner Contest 044 - Problem B: Beautiful Strings
3. AtCoder Beginner Contest 044 - Problem C: Tak and Cards
4. AtCoder Beginner Contest 044 - Problem D: Digit Sum