

AtCoder

AtCoder Beginner Contest 044: *Upsolving*

Prof. Edson Alves - UnB/FGA

2020

1. A – Tak and Hotels (ABC Edit)
2. B – Beautiful Strings
3. C – Tak and Cards
4. D – Unbalance

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.

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

5

3

10000

9000

Saída

48000

2

3

10000

9000

20000

Solução

- Há dois casos a serem tratados
- 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 $O(1)$

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 int main()
6 {
7     int N, K, X, Y;
8     cin >> N >> K >> X >> Y;
9
10    auto ans = X*min(N, K) + Y*max(0, N - K);
11
12    cout << ans << '\n';
13
14    return 0;
15 }
```


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.

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

abaccaba

hthth

Saída

Yes

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 $n = n \text{ XOR } 1$ troca a paridade de n (de par para ímpar e de ímpar para par)
- Esta solução tem complexidade $O(|w|)$

Solução $O(|w|)$

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 int main()
6 {
7     string w;
8     cin >> w;
9
10    vector<int> hw(26, 0);
11
12    for (auto c : w)
13        hw[c - 'a'] ^= 1;
14
15    auto ans = accumulate(hw.begin(), hw.end(), 0)
16        ? "No" : "Yes";
17
18    cout << ans << '\n';
19
20    return 0;
21 }
```

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?

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 $(1 \leq N \leq 16)$.

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

4 8

7 9 8 9

3 8

6 6 9

8 5

3 6 2 8 7 6 5 9

Saída

5

0

19

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

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

- O problema consiste em identificar, dentre todos os subconjuntos de $X = \{x_1, x_2, \dots, x_N\}$, 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

```
1 #include <bits/stdc++.h>
2
3 int solve(int N, int A, const std::vector<int>& xs)
4 {
5     int ans = 0;
6
7     for (long long s = 1; s < (1LL << N); ++s)
8     {
9         int sum = 0, m = 0;
10
11         for (long long i = 0; i < N; ++i)
12             if (s & (1LL << i))
13                 sum += xs[i], ++m;
14
15         if (sum % m == 0 and sum / m == A)
16             ++ans;
17     }
18
19     return ans;
20 }
21
```

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 $x_i, x_{i+1}, \dots, x_{N-1}$, escolher k deles tal que sua soma seja igual a s
- A operação de divisão pode ser evitada:

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

equivale a

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

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

Solução $O(N^3X)$

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 using ll = long long;
5
6 const int MAX { 55 };
7
8 ll st[MAX][MAX][MAX*MAX];
9
10 ll dp(int i, int m, int sum, int N, const vector<int>& xs)
11 {
12     if (i == N)
13         return (sum == 0 and m == 0 ? 1 : 0);
14
15     if (st[i][m][sum] != -1)
16         return st[i][m][sum];
17
18     auto res = dp(i + 1, m, sum, N, xs);
19
20     if (m and xs[i] <= sum)
21         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 }
```

- 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 $y = \{y_1, y_2, \dots, y_N\}$ tal que $y_i = A$, para todo $i = 1, 2, \dots, N$
- A média aritmética de todos os elementos da sequência y é igual a A , e a soma de todos os elementos é igual a $S = NA$

- A ideia central é que, para cada elemento x_i da sequência $\{x_1, x_2, \dots, x_N\}$, 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 $x_i, x_{i+1}, \dots, x_{N-1}$
- 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 $X = \max\{x_1, x_2, \dots, x_N\}$
- Há $O(N^2X)$ estados distintos, com transições em $O(1)$, de modo que a solução tem complexidade $O(N^2X)$

Solução $O(N^2X)$

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 using ll = long long;
5
6 const int MAX { 55 };
7 ll st[MAX][2*MAX*MAX];
8
9 ll dp(int i, int sum, int N, int A, const vector<int>& xs)
10 {
11     if (i == N)
12         return (sum == N*A ? 1 : 0);
13
14     if (st[i][sum] != -1)
15         return st[i][sum];
16
17     st[i][sum] = dp(i + 1, sum, N, A, xs) +
18                 dp(i + 1, sum - A + xs[i], N, A, xs);
19
20     return st[i][sum];
21 }
```

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

Given a string t , we will call it unbalanced if and only if the length of t is at least 2, and more than half of the letters in t are the same. For example, both 'voodoo' and 'melee' are unbalanced, while neither 'noon' nor 'a' is.

You are given a string s consisting of lowercase letters. Determine if there exists a (contiguous) substring of s that is unbalanced. If the answer is positive, show a position where such a substring occurs in s .

Constraints

- $2 \leq |s| \leq 10^5$
- s consists of lowercase letters.

Partial Score

- 200 points will be awarded for passing the test set satisfying $2 \leq N \leq 100$.

Input

Input is given from Standard Input in the following format:

s

Output

If there exists no unbalanced substring of s , print **'-1 -1'**.

If there exists an unbalanced substring of s , let one such substring be $s_a s_{a+1} \dots s_b$ ($1 \leq a < b \leq |s|$), and print **'a b'**. If there exists more than one such substring, any of them will be accepted.

Exemplo de entradas e saídas

Entrada

needed

atcoder

Saída

2 5

-1 -1

- Seja N o tamanho da string s
- Há $O(N^2)$ substrings de s , e a verificação de cada substring pode ser feita em $O(N)$, o que levaria a uma solução $O(N^3)$, que receberia veredito TLE, pois $N \leq 10^5$
- É preciso, portanto, encontrar uma forma eficiente de se processar as substrings e também de reduzir o número de substrings a serem verificadas

- Como qualquer string desbalanceada pode ser dada como resposta do problema, o princípio da casa dos pombos pode reduzir drasticamente o número de substrings a serem verificadas
- Uma string desbalanceada tem mais da metade de seus caracteres repetidos
- Se N é par e s é desbalanceada, ela possui ao menos um par de caracteres consecutivos iguais
- Como uma string com $N = 2$ com dois caracteres iguais é desbalanceada, é suficiente indicar estes caracteres ao invés da string inteira

- Se N é ímpar, é possível que s seja desbalanceada e que não tenha dois caracteres consecutivos iguais
- Isto só ocorre se o número de repetições é igual a $(N + 1)/2$ e o caractere repetido aparece em todos os índices ímpares de s :

$$s = cs_2cs_4 \dots s_{N-1}c$$

- Porém, se $N = 3$ e s é da forma cs_2c , ela também será desbalanceada e poderá ser utilizada como resposta ao invés de s
- Portanto, basta checar apenas estes dois casos, gerando uma solução $O(N)$

Solução $O(|s|)$

```
1 // AtCoder Beginner Contest 044 - Problem D: Digit Sum (https://atco)
2 #include <bits/stdc++.h>
3
4 using namespace std;
5 using ll = long long;
6
7 const ll oo { 1LL << 62 };
8
9 ll sum_digits(ll x, ll base)
10 {
11     ll res = 0;
12
13     while (x)
14     {
15         res += (x % base);
16         x /= base;
17     }
18
19     return res;
20 }
21
```

Solução $O(|s|)$

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

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](#)