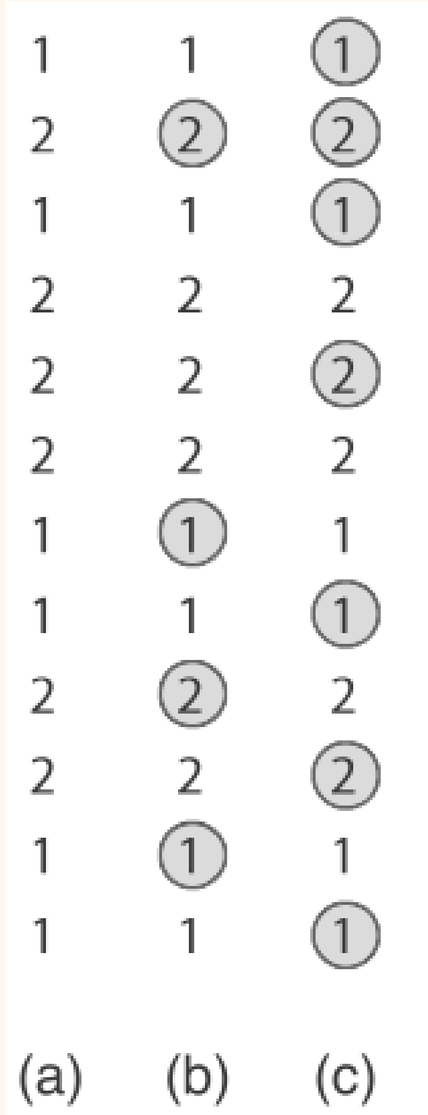
SEQUÊNCIA SECRETA

Olímpiada Brasileira de Informática de 2019 nível Júnior

Na calçada em frente ao Palácio Imperial, não se sabe a razão, existe uma sequência de N números desenhados no chão. A sequência tem a seguinte forma: ela começa e termina com o número 1; apenas os números 1 e 2 aparecem nela; e o número 2 aparece pelo menos uma vez. Veja um exemplo na coluna (a) da figura ao lado.

Ninguém sabe o significado da sequência e, justamente por isso, várias teorias malucas surgiram. Uma delas diz que a sequência representa, na verdade, apenas um valor que estaria relacionado a um segredo dos imperadores. Esse valor é a quantidade máxima de números da sequência que poderiam ser marcados com um círculo, de modo que a sequência de números marcados não contenha dois números iguais consecutivos.A coluna (b) da figura acima ilustra uma sequência de 4 números marcados que obedece a restrição acima. Só que é possível marcar 7 números, como mostra a coluna (c) da figura.

Neste problema, dada a sequência original de números desenhados no chão da calçada, seu programa deve computar e imprimir a quantidade máxima de números da sequência que poderiam ser marcados com um círculo sem que haja dois números iguais consecutivos na sequência marcada.



Entrada

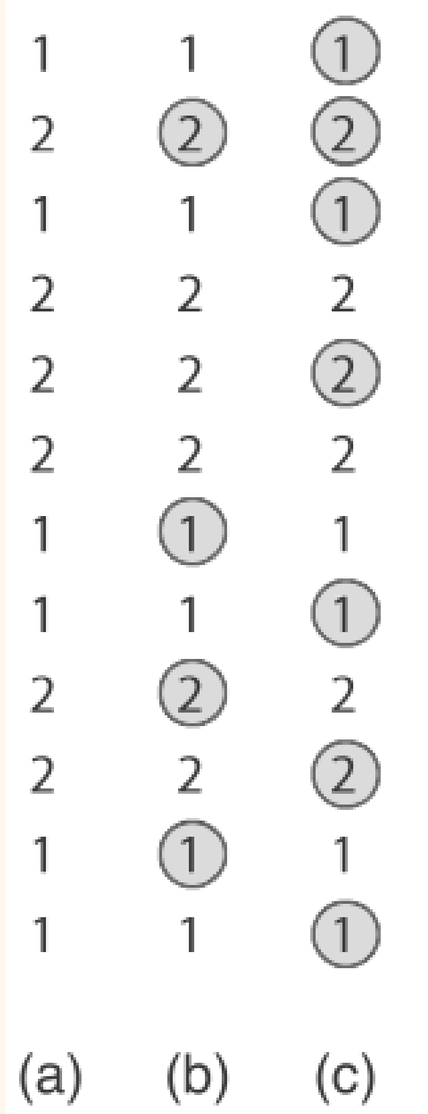
A primeira linha da entrada contém um inteiro N representando o tamanho da sequência. As N linhas seguintes contêm, cada uma, um inteiro V_i, para $1 \leq i \leq N$, definindo a sequência de números desenhados no chão da calçada imperial.

Saída

Seu programa deve imprimir uma linha contendo um número inteiro representando a quantidade máxima de números da sequência que poderiam ser marcados com um círculo sem que haja dois números iguais consecutivos na sequência marcada.

Restrições

- $3 \leq N \leq 500$
- V_i é igual a 1 ou 2, para $1 \leq i \leq N$



Solução

- Criar um array como estrutura de dados para armazenar cada número da sequência

- Criar uma pilha para construir a maior subsequência possível

- Começamos empilhando o primeiro número. Para cada número da sequência, só empilhamos se for diferente do topo da pilha, assim garantimos que nunca há dois números iguais consecutivos

- No final, o tamanho da pilha representa a quantidade máxima de números que podem ser marcados com círculo, obedecendo à regra do problema.

Complexidade de tempo e espaço: O(n)

```c
#define MaxPilha 500

typedef struct{
    int topo;
    int array[MaxPilha];
}Pilha;
```

```c
int numeros_da_sequencia(int array[],int tamanho){
    Pilha stack = criarPilha();
    push(&stack, array[0]);
    int sequencia = 1;

    for(int i = 1; i < tamanho; i++){
        if(peek(stack) != array[i]){
            push(&stack,array[i]);
            sequencia++;
        }
    }
    return sequencia;
}
```

Código Completo

```c
#define MaxPilha 500
#include <stdio.h>

typedef struct{
        int topo;
        int array[MaxPilha];
}Pilha;

Pilha criarPilha();
void push(Pilha*,int);
int peek(Pilha);
int numeros_da_sequencia(int arr[],int);

int main(){
    int tamanho_array,numero, qtd_numeros = 0;
    //input
        FILE *file  =  fopen("input3.txt", "r");

        if(file == NULL){
                printf("Erro ao ler o arquivo");
        }else{
                fscanf(file,"%d", &tamanho_array );

                int array[tamanho_array];

                for(int i = 0; i < tamanho_array; i++){
                        fscanf(file,"%d", &numero);
                        array[i] = numero;
                }

                fclose(file);
                //output
                qtd_numeros = numeros_da_sequencia(array,tamanho_array);
                printf("%d", qtd_numeros);
        }

        return 0;
}
```

```c
Pilha criarPilha(){
        Pilha stack;
        stack.topo = 0;

        return stack;
}

void push(Pilha* stack, int item){

        if(stack->topo < MaxPilha){
                stack->array[stack->topo] = item;
                stack->topo++;
        }
}


int peek(Pilha stack){
        int topo = -1;
        if(stack.topo > 0){
        topo = stack.array[stack.topo - 1];
        }
        return topo;
}

int numeros_da_sequencia(int array[],int tamanho){
        Pilha stack = criarPilha();
        push(&stack, array[0]);
        int sequencia = 1;

        for(int i = 1; i < tamanho; i++){
                if(peek(stack) != array[i]){
                        push(&stack,array[i]);
                        sequencia++;
                }
        }
        return sequencia;
}
```