

**Bit Mask Search**

ou

**Busca de Máscara de Bits**

Usa a representação binária para  
representar dados...

Paula Rehbein

## Bit Mask Search

ou

## Busca de Máscara de Bits

- Usa a representação binária de um número.

Por exemplo:  $5 = 0101$  representa que dois fatos/acontecimentos são verdadeiros.

- Os bits são indexados iniciando em 0 e da direita para a esquerda.

Por exemplo: tendo  $5 = 0 \ 1 \ 0 \ 1$

Indices = [3] [2] [1] [0]

- Podemos usar essa representação para se referir a acontecimentos, sendo cada 0 e 1 um acontecimento, com 0 como não ocorrido e 1 para ocorrido.

## Bit Mask Search

ou

## Busca de Máscara de Bits

Por exemplo, temos uma lista de compras com 5 produtos e queremos verificar se todos os produtos foram comprados:

- Se o produto do índice 1 e o produto do índice 4 não foram comprados a representação seria  $01101 = 18$  (em decimal).
- Agora se apenas os produtos do índice 2 e do índice 0 (zero) foram comprados a representação seria  $00101 = 5$  (decimal).
- Ou se todos tivesse sido adquiridos a representação seria  $11111 = 31$ .

## Bit Mask Search

ou

## Busca de Máscara de Bits

Como é feita as mudanças nas representação dos bits?

- Feita com a operação shift left, ou seja, ele move o 1 para a esquerda j vezes no número binário, o que é equivalente a  $2^j$ .     ( $1 << j$ )

Ou seja, se  $j = 2$ , então  $2^2$ :

$$4 = 0 \ 1 \ 0 \ 0$$

Indices = [3] [2] [1] [0]

Verificação: verificar se algo aconteceu olhando o seu bit correspondente.

- Seguindo o exemplo da lista de compras temos os itens: sabonete (índice 0), farofa (índice 1), café (índice 2), erva-mate (índice 3) e pão (índice 4), sendo cada item um número 0 ou 1.
- Digamos que a lista de itens comprados esteja dessa forma  $10110 = 22$  e queremos verificar se foram comprados café e sabonete, para isso verificamos o valor de seu índice (representado por  $j$ ), na lista (representado por  $x$ )  $x \& (1 << j)$  ou seja, verificamos se o elemento da posição  $j$  está ativo em  $x$ .

10110 (lista atual)

— and 00100 (café, valor = 4)

— 00100 (não retornou 0, assim é true, o café está na lista)

```
def verifica(x:int, j:int ):  
    if x&(1<<j):  
        return True  
    else:  
        return False
```

## Ligar Bit:

- Agora digamos que em nossa lista de compras (sabonete, farofa, café, erva-mate e pão) representada por 10110, queremos ligar o bit do sabonete (índice 0), pois ele foi comprado. Para isso aplicamos a operação  $x |= (1 << j)$ , ou seja, aplicamos a operação lógica or com o número 1 ao índice que queremos ligar, o que resulta em 1 ( 1 or 0 = 1):

101 10 (lista inicial, valor = 22)

or 00001 (sabonete, valor = 1)

— 101 11 (lista final, retornou 23 pois é o resultado de or entre 1 e 22).

```
def liga(x:int, j:int):  
    x |= (1<<j)  
    return x
```

## Desligar Bit:

- Agora na nossa lista (sabonete, farofa, café, erva-mate e pão), atualmente  $10111 = 23$ , percebemos um erro, não foi comprado pão (índice 4), ou seja, temos que desligar seu bit correspondente.
- Isso pode ser feito com  $x = x \& \sim(1 << j)$ ; o que é nada mais que  $x$  and negação de 1 na posição  $j$ , o que retorna false ( $\sim 1 = 0$ , assim,  $\sim 1$  and  $1 = 0$ ), vejamos:

$$\sim(10000) = 01111$$

10111 (lista inicial, valor = 23)

and 01111 (lista sem pão = 15)

— — 00111 (lista final, retornou 7).

```
def desliga(x:int, j:int):  
    x = x&~(1<<j)  
    return x
```

## Operação Flip:

- Essa operação é usada para ligar um bit desligado e desligar um bit ligado, ela faz isso aplicando a operação lógica xor ao índice que escolhemos:  $x = x \wedge (1 << j)$ ; Por exemplo, temos nossa lista atual (sabonete, farofa, café, erva-mate e pão) representada por  $00111 = 7$ , e acabamos de ir no mercado novamente e compramos a erva-mate (índice 3), veja:

00111 (lista inicial, valor = 7)

- xor 01000 (valor da erva-mate = 8) -

01111 (lista final, retornou 15).

```
def flip(x:int, j:int):  
    x^=(1<<j)  
    return x
```

```
7  v = []
8  resp = 0
9  maior = []
10
11 quantidade_itens = int(input("Quantidade de itens: "))
12 peso_maximo = float(input("Peso máximo em Kg que a mochila aguenta: "))
13
14 for i in range(quantidade_itens):
15     v.append(float(input(f"Peso do item {i+1} em Kg: ")))
16
17
18 for mask in range((1 << quantidade_itens)):
19     soma = 0
20     itens = []
21     for i in range(quantidade_itens):
22         if(mask &(1<<i)):
23             soma+=v[i]
24             itens.append(i+1)
25
26         if soma <= peso_maximo:
27             if len(maior)<len(itens):
28                 maior = itens
29
30
31 print("Quantidade de itens máximos que cabem na mochila: ",len(maior))
32 print("Itens", maior)
33
```