

SubSets e BitMasking

SCC 218 - Alg. Avançados e Aplicações.

O problema Subset Sum !!!!

- O problema do SSS (Subset Sum) é a generalização do problema da Mochila que diz o seguinte: dado um conjunto de números inteiros, verificar se existe um subconjunto (subset) cuja soma seja um valor V .
- Ex: $S = \{1,2,3,4\}$ e $V = 7$
 - há duas soluções possíveis: $\{1,2,4\}$ e $\{3,4\}$
- Se vc resolveu o problema da mochila, poderá resolver este aqui:
- Primeiro passo é ver que o problema pode ser recursivo e que passa por assumir que: a) o elemento pode **não** pertencer ou pode pertencer ao subconjunto.
- Suponha que os elementos estão em: $\text{vet}[] = \{1,2,3,4\}$

O Problema Subset Sum!!!!

- Para cada elemento de vet:
 - $\text{vet}[i] > V$ (não podemos incluir $\text{vet}[i]$ na solução) $> \text{SSS}(i, V) = \text{SSS}(i-1, V)$
 - $\text{vet}[i] \leq V$ (podemos incluir OU não incluir) $> \text{SSS}(i, V) = \text{SSS}(i-1, V)$ **OU** $\text{SSS}(i-1, V - \text{vet}[i])$
 - Assim sendo, teremos
- Quais são os casos base?
 - $\text{SSS}(0, V) = \text{false}$ (**para $S > 0$**): se não há mais elementos não encontramos um subconjunto
 - $\text{SSS}(i, 0) = \text{true}$ (para qq i): chegamos a uma solução, incluindo ou não o elemento
- Com isso em mente, faça:
 - a versão recursiva
 - a versão PD recursiva (top-down)
 - a versão PD iterativa (bottom-up)

Conjuntos e SubConjuntos

- Seja um conjunto $S = \{A, B, C\}$
- Sabemos que a quantidade de todos os possíveis subconjuntos de S é 2^n , onde n é a quantidade de elementos em S
- Subconjuntos de $S = \{ \{\}, \{A\}, \{B\}, \{C\}, \{AB\}, \{AC\}, \{BC\}, \{ABC\} \}$

SubConjuntos e cadeias binárias

- Podemos representar os elementos de um conjunto, $S = \{A, B, C\}$, como sendo bits de uma string(cadeia) binária.
- Vamos assumir que o 1o elemento é o bit menos significativo e assim sucessivamente.
- Subconjuntos de $S = \{ \{\}, \{A\}, \{B\}, \{C\}, \{AB\}, \{AC\}, \{BC\}, \{ABC\} \}$

SubConjuntos e cadeias binárias

$\{\}$	000	0
$\{A\},$	001	1
$\{B\},$	010	2
$\{C\},$	100	4
$\{AB\}$	011	3
$\{AC\}$	101	5
$\{BC\}$	110	6
$\{ABC\}$	111	7

Verificar se o i^{th} bit está se

- Seja $N = 107$ (01101011)
- Vamos verificar se o 4th está setado:

```
mask = 1 << 4; // criamos uma máscara >>> 00001000
```

```
if ( (N & mask) != 0) >>> bit setado
```

```
caso contrário, >>>> bit não setado
```

Gerando todos os possíveis subconjuntos...

Seja `string = str = "ABC"; int n = 3; int nSubConj = 1 << n;`

```
// para todos os possíveis subconjuntos
for (int scon = 0; scon < nSubConj; ++scon){
    // para cada elemento (A, B, C, ....)
    for (int elem = 0; elem < n; ++elem){
        int mask = 1 << elem;
        if ((scon & mask) != 0)
            printf("%c,", str[elem]);
    }
    printf("\n");
}
```

SubSet Sum e bitmasking

- Retome o problema de SubSet Sum
- É possível implementá-lo usando os conceitos de bitmask que acabamos de ver e de forma iterativa?

```
bool SSS(int *vet, int n, int valor){
    int nSubConj = 1 << n; // nro de subconjuntos !!!!!
    bool achou = false;

    printf("%d\n", nSubConj);

    // para todos os possiveis subconjuntos
    for (int scon = 0; scon < nSubConj; ++scon){
        int soma = 0; // acumula a soma
        // para cada elemento (A, B, C, ....)
        for (int elem = 0; elem < n; ++elem){
            int mask = 1 << elem;
            if ((scon&mask) != 0) // elemento presente neste bit
                soma += vet[elem];
        }
        if (soma == valor){
            achou = true;
            break;
        }
    }
    return achou;
}
```