PROJETO E ANÁLISE DE ALGORITMOS - LABORATÓRIO II

Nome: João Victor de Mesquita Cândido dos Santos **RA:** 102028

Para a resolução do problema proposta foi usando o método de *backtracking* para definir um subconjunto de possíveis permutações a partir de cada índice de entrada do vetor de massas 'V[20]'. Para cada elemento da entrada é gerado um subconjunto e esse subconjunto é a representação de todas as possíveis permutações existentes, sendo que para cada valor N de elementos é possível se obter um conjunto de 2^N de permutações pois todo elemento pertencente ao conjunto **(1,...,N-1,N)** pode ser escrito como uma combinação dos demais.

Primeiro se inicia com a leitura dos possíveis elementos e após isso é feita a leitura das massas de cada elemento considerado, para se gerar ao subconjunto de permutações de cada elemento do conjunto de massas é realizado um processo chamado de *bitmask* que consiste em um deslocamento de bits para poder encontrar as possíveis combinações que formam cada elemento do conjunto. Todas as possíveis permutações são salvas em um novo vetor que pode ter um tamanho de 10.485.77 no pior caso, ou seja, se pode gerar um novo subconjunto com até 2^20 possíveis valores de massa, sendo 20 o número máximo de elementos que podem ser lidos.

Ao final se obtém um vetor com todos os valores de massa do primeiro conjunto e todas suas possíveis permutações, após isso é feita uma comparação de todos os elementos do subconjunto gerado com os possíveis compostos, se o valor presente no segundo vetor for igual ao do subconjunto uma flag é acionada e então é feito o *print* da palavra "yes" caso contrário os valores são diferentes e então é feito o *print* da palavra "no", finalizando assim o processo e resolvendo o problema proposto

```
#define MAXTAM 1048577
int main()
{
  int M,N,V[20],V2[200],flag=0,V3[MAXTAM],i,j;
  scanf("%d",&N);
  int max=pow(2,N);
  for(i=0;i<N;i++){
     scanf("%d",&V[i]);
  }
  scanf("%d",&M);
  for(i=0;i<M;i++){
```

```
        scanf("%d",&V2[i]);
    }

     for(i=0;i<(1<<N);i++){ //deslocamento de bits para formar o
vetor de permutações
            for(j=0;j<N;j++){
                  if((i & (1<<j))!= 0){
                        V3[i]+=V[j];
      }
     }
    }

   for(i=0;i<M;i++){
      for(j=0;j<max;j++)
         if(V2[i]==V3[j]){  //verifica se cada elemento do segundo
vetor é igual a algum elemento do vetor com todas as combinações
possíveis
            flag=1;//variavel para verificar que encontrou
         }
         if(flag==1){
            printf("yes\n"); //caso en
            flag=0;
         }
         else
         printf("no\n");
   }
return 0;
}
```

Através da análise do algoritmo foi possível analisar que o mesmo apresenta uma complexidade de exponencial de 2^N, isso se dá devido ao fato do laço crescer sempre através do valor de N de entrada para gerar as permutações.