

- **Você deve justificar todas as suas respostas;**
- Não é permitida a consulta a livros ou anotações;
- **O plágio ou a tentativa de plágio implicará na nota zero.**

1. (3.5) Dado um conjunto  $E = \{1, 2, \dots, n\}$  de  $n$  pessoas, e um valor  $K$ . Deseja-se determinar se existe um grupo de  $K$  pessoas, subconjunto de  $E$  que se conhecem mutuamente, ou se existe um grupo de  $K$  pessoas, também um subconjunto de  $E$ , que se desconhecem mutuamente.

Considere o teorema abaixo:

**Teorema 1 :** *Sabe-se encontrar todos os subconjuntos de  $p$  pessoas de  $E$  tais que todas as pessoas em cada um desses subconjuntos ou se conhecem mutuamente, ou se desconhecem mutuamente.*

Assuma que  $n$  é uma constante (ou seja é um valor fixo, como 150, por exemplo).

- (a) Prove o teorema acima por indução matemática em  $p$
- (b) Apresente o algoritmo correspondente a sua prova no item anterior
- (c) (só para refletir) Considere  $n = 6$  e  $K = 3$ , existe algum conjunto de 6 pessoas onde nem 3 pessoas se conhecem mutuamente, nem 3 pessoas se desconhecem mutuamente ?

Obs.: A relação “conhece” é simétrica, i.e. se  $a$  conhece  $b$  então  $b$  conhece  $a$ .

2. (3.5) Considere o algoritmo abaixo, onde  $V$  é um vetor que representa um conjunto de  $num$  inteiros **dado**. Este algoritmo encontra o elemento de menor valor e o elemento de maior valor desse conjunto.

```
struct par {
    int i1;
    int i2;
};
struct par minmax (int , int );
int V[NMAX];

main( )
{
    struct par mima;
    mima = minmax(1, num); /* num: número de elementos no vetor V */
    printf("\n\n Mínimo e máximo do vetor: V[%d]=%d e V[%d]=%d \n",
        mima.i1, V[mima.i1], mima.i2, V[mima.i2]);
}
```

```

struct par minmax( inicio, fim )
int inicio, fim;
{
  struct par mm, mm1, mm2;          int meio;
  if (fim - inicio <= 1) {
    if( V[inicio] < V[fim] ) {
      mm.i1 = inicio;                mm.i2 = fim;
    }
    else {
      mm.i1 = fim;                   mm.i2 = inicio;
    }
  }
  else {
    meio = ((inicio + fim)/2);
    mm1 = minmax(inicio, meio);
    mm2 = minmax(meio+1, fim);
    if ( V[mm1.i1] > V[mm2.i1] ) {
      mm.i1 = mm2.i1;
    }
    else {
      mm.i1 = mm1.i1;
    }

    if ( V[mm1.i2] < V[mm2.i2] ) {
      mm.i2 = mm2.i2;
    }
    else {
      mm.i2 = mm1.i2;
    }
  }
  return( mm );
}

```

- (a) Enuncie o teorema cuja prova por indução matemática resulta no algoritmo acima.
- (b) Apresente a prova do teorema correspondente a este algoritmo.

Dica: considere indução forte.

3. (3.5) Seja  $G = (V, E)$  um grafo orientado onde  $l_e \geq 0$ ,  $e \in E$ , é o comprimento dos arcos de  $G$ . O **girth**( $G$ ) é definido como o comprimento do ciclo orientado de menor comprimento em  $G$ .
- (a) Enuncie um teorema cuja prova por indução matemática demonstra que sabe-se encontrar o **girth**( $G$ ) para um grafo  $G$  qualquer.
- (b) Apresente a prova deste teorema por indução matemática.
- (c) Escreva o algoritmo correspondente à sua prova.

Dica: utilize o teorema que leva ao algoritmo de caminho mais curto entre um vértice e todos os demais no grafo.