Usando divisão e conquista forneça algoritmos para os itens a seguir e forneça a ordem de complexidade de tempo de execução:

a) Encontrar o maior valor em um vetor.

```
def buscaBinaria(vec,max):  #Custo das operações dentro da função, O(1), considerando que
  size=len(vec)
  middle=int(size/2)
  if(size==0):
      return max
  if(vec[middle]>max):
    max=vec[middle]
  aux1=buscaBinaria(vec[0:middle],max)
  aux2=buscaBinaria(vec[middle+1:size],max)
  if(aux1>aux2):
    return aux1
  else:
    return aux2
```

```
x=[18,2,3,4,25,6,92,11,9,10]
y= 0
y= buscaBinaria(x,y)
print(y)
```

```
92
```

b) Encontrar o maior e o menor elemento em um vetor.

```
def buscaMinMax(vec,max,min): #Custo das operações dentro da função, O(1), considerando qu
  size=len(vec)
  middle=int(size/2)
  if(size==0):
      return max,min
  if(vec[middle]>max):
    max=vec[middle]
  if(vec[middle]<min):
    min=vec[middle]
  auxMax1,auxMin1=buscaMinMax(vec[0:middle],max,min)
  auxMax2,auxMin2=buscaMinMax(vec[middle+1:size],max,min)
  if(auxMax1>auxMax2):
    max= auxMax1
  else:
    max= auxMax2
```

```
    max= auxMax2
  if(auxMin1<auxMin2):
    min= auxMin1
  else:
    min= auxMin2
  return max,min


x=[18,2,3,4,25,6,92,11,9,10]
max=0
min=9999999999
max,min = buscaMinMax(x,max,min)
print(max,min)
```

```
92 2
```

c) Exponenciação.

```
def exp(base,expoente):#Custo das operações dentro da função, O(1), considerando multiplic
  if(expoente==0):
    return 1
  if(expoente==1):
    return base
  if(expoente==2):
    return base*base
  part=int(expoente/2)
  aux1=exp(base,part)
  if(base%2==0):
    aux2=exp(base,part)
  else:
    aux2=exp(base,part+1)
  return aux1*aux2



result1=exp(2,8)
result2=exp(2,0)
result3=exp(2,1)
result4=exp(2,4)
print(result1)
print(result2)
print(result3)
print(result4)
```

```
256
1
2
16
```

Produtos pagos do Colab - Cancelar contratos

✓ 0s conclusão: 10:03