

Fundação Universidade Federal de Mato Grosso do Sul



Disciplina: Algoritmos e Programação II **Professora**: Yorah Bosse

Algoritmos de Ordenações Internas (Vetores)

1. SELEÇÃO DIRETA OU LINEAR (SELECTION SORT)

```
void direta (int *vet){
  int aux, i_menor, i_maior;
  for (i_menor = 0; i_menor < max-1; i_menor++)
     for (i_maior = i_menor+1; i_maior < max; i_maior++)
     if (vet[i_menor] > vet[i_maior]){
        aux = vet[i_menor];
        vet[i_menor] = vet[i_maior];
        vet[i_maior] = aux;
     }
}
```

2. SELEÇÃO DIRETA OTIMIZADA OU LINEAR OTIMIZADA (DELAYED SELECTION SORT)

```
void otimizada (int *vet){
   int aux, i_troca, i_menor, i_maior;
   for (i_menor = 0; i_menor < max-1; i_menor++){
      i_troca = i_menor;
      for (i_maior = i_menor+1; i_maior < max; i_maior++)
            if (vet[i_troca] > vet[i_maior])
            i_troca = i_maior;
      aux = vet[i_menor];
      vet[i_menor] = vet[i_troca];
      vet[i_troca] = aux;
   }
}
```



Fundação Universidade Federal de Mato Grosso do Sul



Disciplina: Algoritmos e Programação II **Professora**: Yorah Bosse

Algoritmos de Ordenações Internas (Vetores)

3. INSERÇÃO (INSERTION SORT)

```
void insercao (int *vet){
    int aux, anterior, atual;
    for (atual = 1; atual < max; atual++){
        aux = vet[atual];
        anterior = atual - 1;
        while (anterior > -1 && vet[anterior] > aux){
            vet[anterior+1] = vet[anterior];
            anterior--;
        }
        vet[anterior+1] = aux;
    }
}
```

4. MÉTODO BOLHA (BUBBLE SORT)



Fundação Universidade Federal de Mato Grosso do Sul



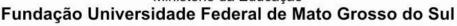
Disciplina: Algoritmos e Programação II **Professora**: Yorah Bosse

Algoritmos de Ordenações Internas (Vetores)

5. SHELL SORT

```
void shell(int *vet){
        int lsup = max,
        d = lsup / 2,
        i,
        х,
        aux;
        while (d > 0) {
                for (i = d; i < lsup; i++) {
                        x = i - d;
                        while (x \ge 0 \&\& vet[x] > vet[x+d]) \{
                                 aux = vet[x];
                                 vet[x] = vet[x+d];
                                 vet[x+d] = aux;
                                 x = x - d;
                        }
                }
                d = d / 2;
        }
}
```







Disciplina: Algoritmos e Programação II **Professora**: Yorah Bosse

Algoritmos de Ordenações Internas (Vetores)

6. MERGE SORT

```
void ordenacao_mergesort (int *vet, int esq, int dir){
     int temp [max];
    int i, j, k, metade;
     if (dir - esq > 0){
      metade = (dir + esq) / 2;
      ordenacao_mergesort (vet,esq,metade);
      ordenacao_mergesort (vet,metade+1,dir);
      for (i = esq; i <= metade; i++)
        temp[i] = vet[i];
      for (i = metade+1; i <= dir; i++)
        temp[dir+metade+1-i] = vet[i];
      i = esq;
      j = dir;
      for (k = esq; k \le dir; k++)
        if (temp[i] < temp[j]){</pre>
          vet[k] = temp[i];
          i++;
        }
           else{
          vet[k] = temp[j];
          j--;
    }
}
```



Serviço Público Federal Ministério da Educação Fundação Universidade Federal de Mato Grosso do Sul



Disciplina: Algoritmos e Programação II **Professora**: Yorah Bosse

Algoritmos de Ordenações Internas (Vetores)

7. MÉTODO DE TROCA E PARTIÇÕES (QUICK SORT)

```
void ordenacao_quicksort (int *vet, int prim, int ult){
    int i, j, x, y;
    i = prim;
    j = ult;
    x = vet[(prim+ult)/2];
    do{
        while (vet[i] < x)
            i++;
        while (x < vet[j])
           j--;
        if (i \le j)
            y = vet[i];
            vet[i] = vet[j];
            vet[j] = y;
            i++;
            j--;
    }while (i<=j);</pre>
    if (prim < j)
       ordenacao_quicksort(vet,prim,j);
    if (i < ult)
       ordenacao_quicksort(vet,i,ult);
}
```





Disciplina: Algoritmos e Programação II Professora: Yorah Bosse

Algoritmos de Ordenações Internas (Vetores)

8. SELEÇÃO EM ÁRVORE BINÁRIA (HEAP SORT)

```
void monta_heap (int *vet, int e, int d)
  int i, j, x;
  i = e;
 j = (i * 2) + 1;
  x = vet[i];
  while (j \le d)
     if (j < d \&\& vet[j] < vet[j+1])
       j++;
     if (x < vet[j]) {
       vet[i] = vet[j];
       i = j;
       j = (i * 2) + 1;
     else
       j = d + 1;
  }
  vet[i] = x;
}
```

```
void heap (int *v){
       int esq = max / 2,
             dir = max-1,
           aux;
       while (esq > 0) {
           esq--;
           monta heap (v,esq,dir);
       while (dir > 0)
          aux = v[0];
          v[0] = v[dir];
            v[dir] = aux;
            dir--;
          monta_heap (v,esq,dir);
       }
}
```