Bolha

```
for (i=n-1; i>=1; i--)
  for (j=0; j<i; j++)

      if (v[j]>v[j+1])
      { /* troca */
        temp = v[j];
        v[j] = v[j+1];
        v[j+1] = temp;
       }
```

2	5	1	3	10
2	5	1	3	10
2	1	5	3	10
2	1	3	5	10
1	2	3	5	10

Método extremamente lento: só faz comparações entre posições adjacentes

• É o método mais ineficiente entre os simples

– Melhor caso: vetor já ordenado

– Pior caso: vetor de entrada em ordem reversa

• Cada passo aproveita muito pouco do que foi "descoberto" em relação à ordem das chaves no passo anterior (exibe informações redundantes)

Inserção

```
j=1;
while (j<tam)
{
   aux = vetor[j];
   i = j - 1;
   while (  i >= 0 &&  vetor[i] > aux )
   {
      vetor[i+1] = vetor[i];
       i = i - 1;
   }
   vetor[i+1] = aux;
   j = j + 1;
}
```

Aux=

2	5	1	3
2	5	1	3
1	2	5	3
1	2	3	5

Vantagens:

– O número mínimo de comparações e movimentos ocorre quando os itens já estão originalmente ordenados

– O número máximo ocorre quando os itens estão originalmente em ordem reversa, o que indica um comportamento natural para o algoritmo

Seleção

```
for (i=0; i<tam;i++){
   minimo = i;
   // pega indice do menor
   for (j=i+1;j<tam;j++){
       if ( vetor[j] < vetor[minimo] )
           {
               minimo = j;
           }
       }
      aux = vetor[minimo];
       vetor[minimo] = vetor[i];
       vetor[i] = aux;
   }
```

2	5	1	0
0	5	1	2
0	1	5	2
0	1	2	5

Um dos algoritmos mais simples

• Mais recomendado para conjuntos pequenos

• Procedimento:

– Selecione o menor item do conjunto e troque-o com o item que está na posição i

– Repita essas operações com os demais itens até que reste apenas um elemento