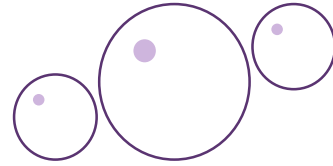
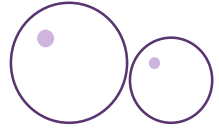


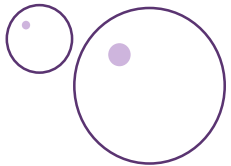
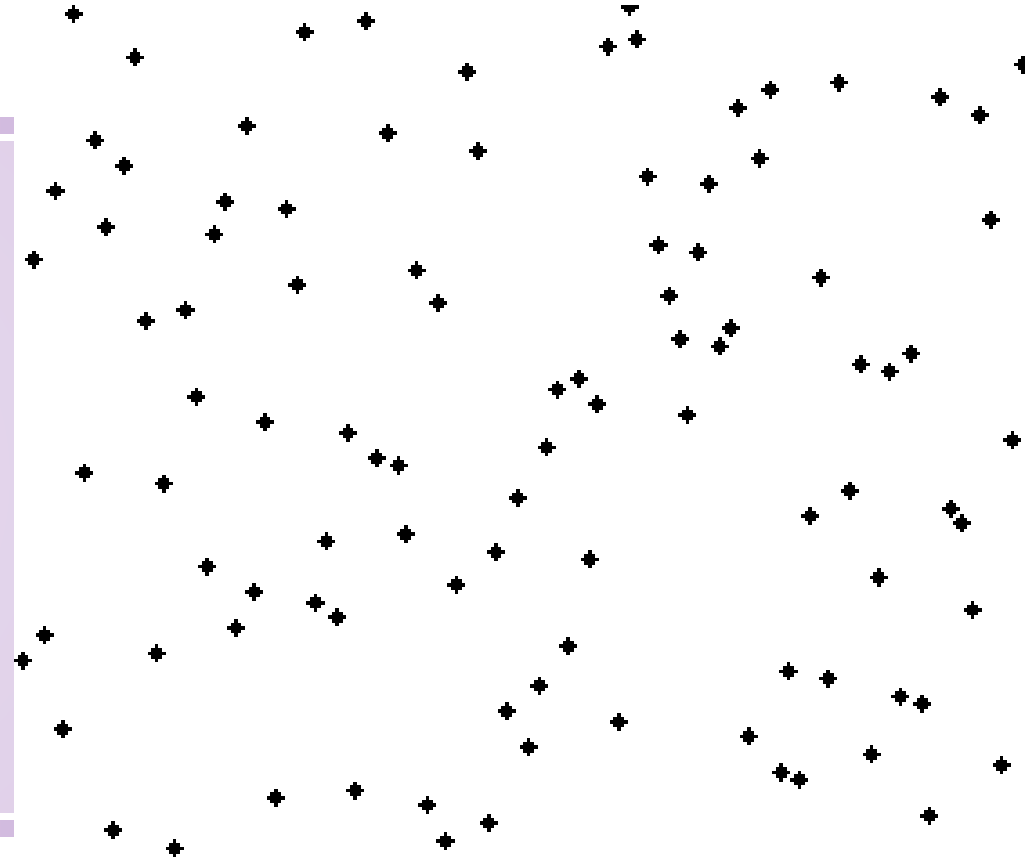
UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
CÂMPUS: PONTA PORÃ – MS

ILDERSON DOS SANTOS,
JOÃO VITOR SANTOS,
JÔNATA ROCHA.

PROF(A): YORAH BOSSE



Bubble Sort





Oque é ?

É um algoritmo de Ordenação que utiliza o método bolha para ordenar os dados.



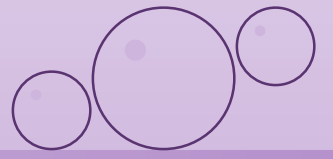

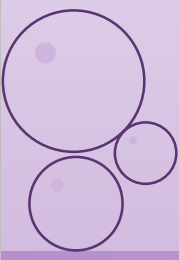
Oque ele faz?

O método bolha consiste em fazer flutuar o maior elemento da sequência para o topo.



Como ele faz?

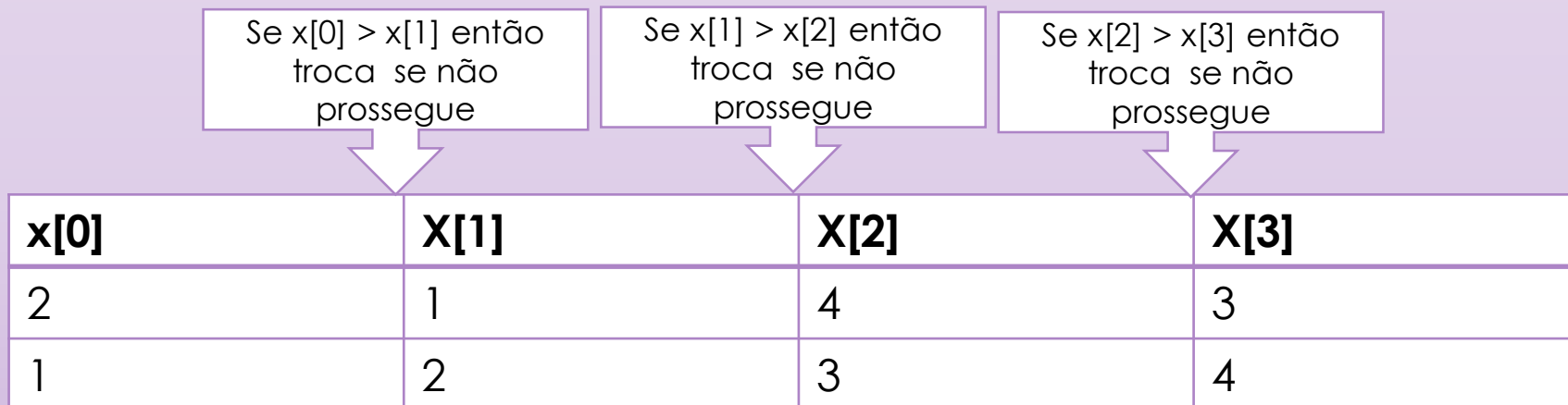
Percorre e compara o vetor várias e várias vezes até atingir o propósito de ordenação estabelecido.



Como ele funciona?

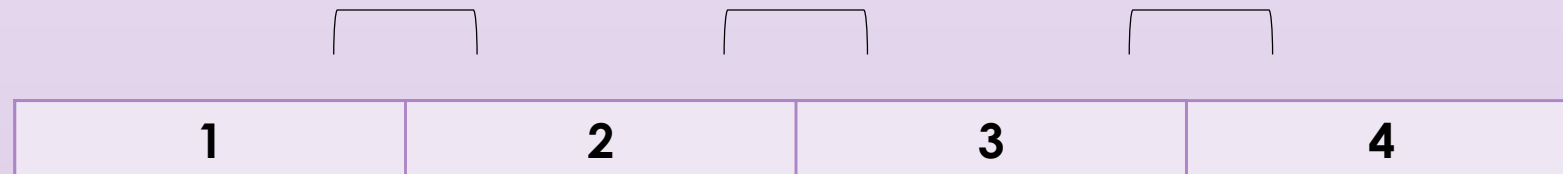
O algoritmo realiza comparação conjunta 2 à 2, comparando o elemento da posição (x) com o elemento da próxima posição do vetor (x+1).

Quando o elemento da posição x for maior que o elemento da posição x+1, o algoritmo realiza a troca de posição dos elementos, caso isso não aconteça o algoritmo continua realizando os testes até que a ordenação esteja correta.



Demonstração

A seguir será demonstrado como funciona o processo, através de um algoritmo escrito em linguagem C, por meio da função Bolha.



Legenda:

max = tamanho do vetor.

Isup = laço de repetição.

bolha = variável de controle do ponto flutuante.

aux = recebe valor de troca.

```
4
5 void bolha(int *vet) {
6
7     int Isup, bolha, aux, i;
8
9     Isup = max - 1;
10
11     while(Isup > 0) {
12
13         bolha = -1;
14
15         for (i = 0; i < Isup; i++)
16
17             if (vet[i] > vet[i+1]) {
18
19                 aux = vet[i];
20                 vet[i] = vet[i+1];
21                 vet[i+1] = aux;
22                 bolha = i;
23
24             }
25
26         Isup = bolha;
27
28     }
29 }
30
31
```

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int max;
5
6  void bolha(int *vet) {
7
8      int Isup, bolha, aux, i;
9
10     Isup = max - 1;
11
12     while(Isup > 0) {
13
14         bolha = -1;
15
16         for (i = 0; i < Isup; i++)
17
18             if (vet[i] > vet[i+1]) {
19
20                 aux = vet[i];
21                 vet[i] = vet[i+1];
22                 vet[i+1] = aux;
23                 bolha = i;
24
25             }
26
27         Isup = bolha;
28
29     }
30 }

```

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #define max 10
4
5  void bolha(int *vet) {
6
7      int Isup, bolha, aux, i;
8
9      Isup = max - 1;
10
11     while(Isup > 0) {
12
13         bolha = -1;
14
15         for (i = 0; i < Isup; i++)
16
17             if (vet[i] > vet[i+1]) {
18
19                 aux = vet[i];
20                 vet[i] = vet[i+1];
21                 vet[i+1] = aux;
22                 bolha = i;
23
24             }
25
26         Isup = bolha;
27
28     }
29 }

```



```

32
33 int main() {
34     int *vet, i;
35
36     do {
37         scanf("%d", &max);
38     } while(max <= 0);
39
40     vet = (int *) malloc(max * sizeof(int));
41
42     if (vet == NULL) {
43         printf("Não foi possível alocar memória.\n");
44         exit(1);
45     }
46
47     for (i = 0; i < max; i++) {
48         scanf("%d", &vet[i]);
49     }
50
51     bolha(vet);
52
53     for (i = 0; i < max; i++) {
54         printf("%d ", vet[i]);
55     }
56
57     printf("\n");
58
59     free(vet);
60
61     return 0;
62 }

```

```

31
32 int main() {
33     int *vet, i;
34
35     printf("Insira %d numeros reais:\n", max);
36
37     vet = (int *) malloc(max * sizeof(int));
38
39     if (vet == NULL) {
40         printf("Não foi possível alocar memória.\n");
41         exit(1);
42     }
43
44     for (i = 0; i < max; i++) {
45         scanf("%d", &vet[i]);
46     }
47
48     bolha(vet);
49
50     for (i = 0; i < max; i++) {
51         printf("%d ", vet[i]);
52     }
53
54     printf("\n");
55
56     free(vet);
57
58     return 0;
59 }

```



```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int max;
5
6  void bolha(int *vet) {
7
8      int Isup, bolha, aux, i;
9
10     Isup = max - 1;
11
12     while(Isup > 0) {
13
14         bolha = -1;
15
16         for (i = 0; i < Isup; i++)
17
18             if (vet[i] > vet[i+1]) {
19
20                 aux = vet[i];
21                 vet[i] = vet[i+1];
22                 vet[i+1] = aux;
23                 bolha = i;
24
25             }
26
27         Isup = bolha;
28
29     }
30 }
31
```

Resultado de testes realizados

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ gcc bubbleSort.c -o bubbleSort
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
4
4 3 2 1
1 2 3 4
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ gcc bubbleSort.c -o bubbleSort
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
4
4 4 4 3
3 4 4 4
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ gcc bubbleSort.c -o bubbleSort
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
4
4 4 3 2
2 3 4 4
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ gcc bubbleSort.c -o bubbleSort
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
4
4 4 4 4
4 4 4 4
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```


Resultado de testes realizados

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ gcc bubbleSort.c -o bubbleSort
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
5
5 4 3 2 1
1 2 3 4 5
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ gcc bubbleSort.c -o bubbleSort
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
5
5 5 4 3 2
2 3 4 5 5
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ gcc bubbleSort.c -o bubbleSort
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
5
5 5 5 4 3
3 4 5 5 5
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ gcc bubbleSort.c -o bubbleSort
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
5
5 5 5 5 4
4 5 5 5 5
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ gcc bubbleSort.c -o bubbleSort
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
5
5 5 5 5 5
5 5 5 5 5
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```

Resultado de testes realizados

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ gcc BubbleSort.c -o BubbleSort
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./BubbleSort
Insira 10 numeros reais:
10 -9 8 8 7 -6 1 5 4 1
-9 -6 1 1 4 5 7 8 8 10
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
2
10 2
2 10
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ gcc bubbleSort.c -o bubbleSort
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
9
90 -12 20 20 -56 -90 0 8 9
-90 -56 -12 0 8 9 20 20 90
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
1
1
1
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
-5
0
20
12 34 45 56 78 90 9 87 76 65 54 43 32 1 10 29 38 47 56 0
0 1 9 10 12 29 32 34 38 43 45 47 54 56 56 65 76 78 87 90
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```

```
jonatarocha@jonatarocha-HP-Notebook: ~/Documentos/FileinC
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$ ./bubbleSort
1
100
100
jonatarocha@jonatarocha-HP-Notebook:~/Documentos/FileinC$
```




VANTAGENS

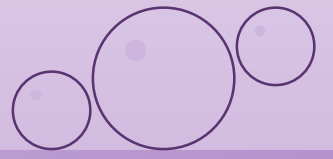
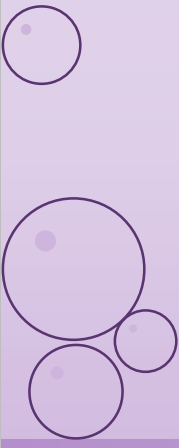
É um código relativamente simples, e fácil de ser implementado.

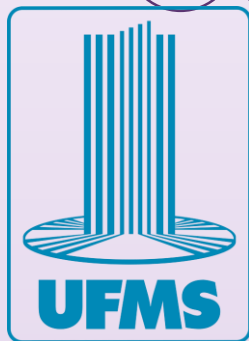
DESVANTAGENS



O nível de complexidade do algoritmo é $O(n^2)$ no pior e médio caso de teste.

A velocidade de tempo de resposta diminui exponencialmente em função do numero de elementos que ele tem que comparar.





UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL

Vídeo demonstrativo de encerramento

