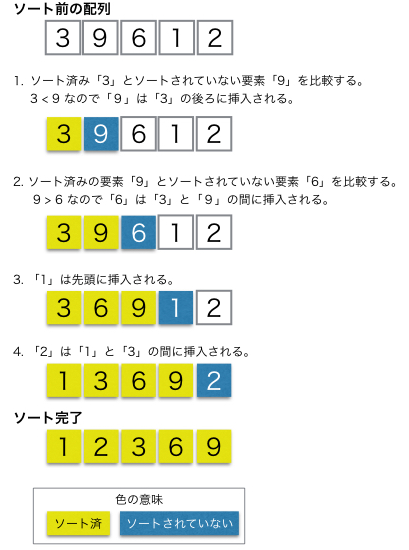
19\_基本情報アルゴリズム⑤

**課題1**

数値配列に対して挿入ソートを使用し、昇順に並び替えて表示するプログラムを作成しなさい。

並び替えの途中経過も表示すること。

◆挿入ソートの流れ



挿入ソートは選択ソートと同様に単純でアルゴリズムが容易だが全ての要素と比較する為、

時間計算量は最悪パターンでも最良パターンでもいずれも*О(n²)*となる。

挿入ソートを高速化したソート法として、シェルソートというものも存在する。

ファイル名：Kad19\_1.java

＜実行結果＞

\*\*\*\*\*\*\*\*\* 元データ \*\*\*\*\*\*\*\*\*

2, 5, 8, 1, 4, 7, 3, 6, 9, 0,

\*\*\*\*\*\* 挿入ソート開始 \*\*\*\*\*\*

2, 5, 8, 1, 4, 7, 3, 6, 9, 0,

2, 5, 8, 1, 4, 7, 3, 6, 9, 0,

1, 2, 5, 8, 4, 7, 3, 6, 9, 0,

1, 2, 4, 5, 8, 7, 3, 6, 9, 0,

1, 2, 4, 5, 7, 8, 3, 6, 9, 0,

1, 2, 3, 4, 5, 7, 8, 6, 9, 0,

1, 2, 3, 4, 5, 6, 7, 8, 9, 0,

1, 2, 3, 4, 5, 6, 7, 8, 9, 0,

0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

\*\*\*\*\*\* 挿入ソート完了 \*\*\*\*\*\*

|  |  |  |  |
| --- | --- | --- | --- |
|  | 戻り値 | 名前 | 引数 |
| メソッド | void | insertSort | int[] data |

◆insertSortメソッドの処理

引数で受け取った配列を挿入ソートを使用して昇順に並び替える。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 戻り値 | 名前 | 引数 |
| メソッド | void | printArray | int[] data |

◆printArrayメソッドの処理

引数で受け取った配列の値を表示する（過去の課題で作成した同じメソッドをコピーし流用する）

◆mainメソッドの処理

①以下の数値配列を宣言する。

int[] numbers = {2, 5, 8, 1, 4, 7, 3, 6, 9, 0};

②数値配列を引数にprintArrayメソッドを呼び出し、ソート前の配列の値を表示する。

System.out.println("\*\*\*\*\*\*\*\*\* 元データ \*\*\*\*\*\*\*\*\*");

printArray(～省略～);

③数値配列を引数にinsertSortメソッドを呼び出し、ソートする。

System.out.println("\*\*\*\*\*\* 挿入ソート開始 \*\*\*\*\*\*");

insertSort(～省略～);

System.out.println("\*\*\*\*\*\* 挿入ソート完了 \*\*\*\*\*\*");

**課題2**

数値配列に対してシェルソートを使用し、昇順に並び替えて表示するプログラムを作成しなさい。

並び替えの途中経過も表示すること。

◆挿入ソートの流れ



このようにシェルソートは等間隔ごとにある程度データを並び替えておくことによって、

最後の挿入ソートを高速化させることができる。

ただし、ほぼ整列されているデータには効果を発揮しない。

データの並びとソートを開始する間隔次第では無駄な比較が増えてしまう可能性もあるので、

時間計算量は最良パターンだと*О(n log n)*だが、最悪パターンだと*О(n²)*になる。

ファイル名：Kad19\_2.java

＜実行結果＞

\*\*\*\*\*\*\*\*\* 元データ \*\*\*\*\*\*\*\*\*

2, 5, 8, 1, 4, 7, 3, 6, 9, 0,

\*\*\*\*\* シェルソート開始 \*\*\*\*\*

2, 3, 6, 1, 0, 7, 5, 8, 9, 4,

0, 1, 2, 3, 5, 4, 6, 7, 9, 8,

0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

\*\*\*\*\* シェルソート完了 \*\*\*\*\*

|  |  |  |  |
| --- | --- | --- | --- |
|  | 戻り値 | 名前 | 引数 |
| メソッド | void | shellSort | int[] data |

◆shellSortメソッドの処理

引数で受け取った配列をシェルソートを使用して昇順に並び替える。

①int型変数hに配列の長さの半分を交換する間隔として初期化する。

int h = data.length / 2; //交換間隔(今課題は5→2→1→0と変化していく)

②シェルソートを行う、コメントを記述しながら省略部分を埋めなさい。

//間隔が0より大きい間ループする

while(～省略～){

//間隔番目の要素からスタートし、データ列の長さまでループする

for(int i = h; ～省略～; i++){

//jを比較初期位置とする

int j = i;

//比較する要素番号が間隔以上、かつ間隔分離れた左側の要素が大きい間ループする

while(j >= h && ～～){

//交換処理

int tmp = data[j];

data[j] = ～省略～;

～省略～ = tmp;

//比較要素を更新(5→0, 6→1...)

j -= h;

}

}

//間隔を更新5→2→1→0

～省略～;

//配列の値を表示

～省略～;

}

|  |  |  |  |
| --- | --- | --- | --- |
|  | 戻り値 | 名前 | 引数 |
| メソッド | void | printArray | int[] data |

◆printArrayメソッドの処理

引数で受け取った配列の値を表示する（過去の課題で作成した同じメソッドをコピーし流用する）

◆mainメソッドの処理

①以下の数値配列を宣言する。

int[] numbers = {2, 5, 8, 1, 4, 7, 3, 6, 9, 0};

②数値配列を引数にprintArrayメソッドを呼び出し、ソート前の配列の値を表示する。

System.out.println("\*\*\*\*\*\*\*\*\* 元データ \*\*\*\*\*\*\*\*\*");

printArray(～省略～);

③数値配列を引数にshellSortメソッドを呼び出し、ソートする。

System.out.println("\*\*\*\*\* シェルソート開始 \*\*\*\*\*");

shellSort(～省略～);

System.out.println("\*\*\*\*\* シェルソート完了 \*\*\*\*\*");