Time complexity analysis ordering algorithm

Code	Time
sortWishListByInsertion(ArrayList<Shelf> shelfsId) {	
String auxiliar;	1
String auxiliar2;	1
int i = 0;	1
int j = 0;	1
while (i < shelfsId.size()){	n+1
auxiliar = shelfsId.get(i).getId();	n
auxiliar2 = wishList.get(i);	n
for (j=i ; j>0 && shelfsId.get(j-1).getId().charAt(0) > auxiliar.charAt(0) ; --j) {	$\frac{n(n+1)}{2}+1$
shelfsId.set(j, shelfsId.get(j-1));	$\frac{n(n+1)}{2}$
wishList.set(j, wishList.get(j-1));}	$\frac{n(n+1)}{2}$
shelfsId.set(j,shelfsId.get(i));	n
wishList.set(j, auxiliar2);	n
++i;}}	n

$6 + 6n + 3\frac{n(n+1)}{2} \rightarrow 6 + 6n + \frac{3n^2}{2} + \frac{3n}{2} \rightarrow \frac{3n^2}{2} + \frac{15n}{2} + 6$ $O_{(n^2)}$

Code	Time
public void sortWishListBySelection(ArrayList<Shelf> shelfsId) {	
for (int i = 0; i < shelfsId.size()-1; i++){	n
int index = i;	n-1
for (int j = i + 1; j < shelfsId.size(); j++){	$\frac{n(n+1)}{2} - 1$
if (shelfsId.get(j).getId().charAt(0) < shelfsId.get(index).getId().charAt(0)){	$\frac{n(n+1)}{2}$
index = j;}}	$\frac{n(n+1)}{2}$
Shelf smallerNumber2 = shelfsId.get(index);	n-1
shelfsId.set(index,shelfsId.get(i));	n-1
shelfsId.set(i, smallerNumber2);	n-1
String smallerNumber = wishList.get(index);	n-1
wishList.set(index, wishList.get(i));	n-1
wishList.set(i, smallerNumber);}}	n-1

$$8n + \frac{3n(n+1)}{2} - 8 \rightarrow 8n + \frac{3n^2}{2} + \frac{3n}{2} - 8 \rightarrow \frac{3n^2}{2} + \frac{19n}{2} - 8 \qquad O_{(n^2)}$$