TD d'algorithmique INF220 – TD2&3 – Tris

Exercice 1 – Implémentation du tri à bulles

• Q1. La fonction **etapeTriBulles** fait une étape du tri à bulles. Comme nous l'avons vu en cours, elle parcourt le tableau d'entiers fourni en entrée en vérifiant pour chaque case si elle est bien inférieure ou égale à la suivante, et échange leurs valeurs si ce n'est pas le cas. Elle renvoie le tableau ainsi obtenu. Que renvoie **etapeTriBulles**({4,5,1,2,2,9,8}) ?

• Q2. Écrivez la fonction **etapeTriBulles**.

• Q3. Pour savoir si on arrête l'algorithme à la fin d'une étape du tri à bulles, il faut savoir si le tableau est trié. Pour cela, écrivez un algorithme récursif **testTrieAvantCase**, qui prend en entrée un tableau d'entier *tab* et un entier *i*, et renvoie VRAI si tab est trié jusqu'à la *i*-ième case incluse, et renvoie FAUX sinon.

Indication pour l'initialisation : que renvoie **testTrieAvantCase** sur n'importe quel tableau *tab* en entrée si l'entrée *i* vaut 1 ?

Indication pour l'hérédité : si le tableau *tab* en entrée de **testTrieAvantCase** est trié jusqu'à la (*i*-1)-ième case, comment savoir s'il est trié jusqu'à la *i*-ième ? Et s'il n'est pas trié jusqu'à la *i*-ième case ?

Q4. En utilisant **etapeTriBulles** et **testTrieAvantCase**, écrivez l'algorithme **triBulles** qui prend en entrée un tableau d'entiers et renvoie ce tableau trié par le tri à bulles.

TD d'algorithmique INF220 – TD2&3 – Tris

Exercice 1 – Implémentation du tri à bulles

• Q1. La fonction **etapeTriBulles** fait une étape du tri à bulles. Comme nous l'avons vu en cours, elle parcourt le tableau d'entiers fourni en entrée en vérifiant pour chaque case si elle est bien inférieure ou égale à la suivante, et échange leurs valeurs si ce n'est pas le cas. Elle renvoie le tableau ainsi obtenu. Que renvoie **etapeTriBulles**({4,5,1,2,2,9,8}) ?

• Q2. Écrivez la fonction **etapeTriBulles**.

• Q3. Pour savoir si on arrête l'algorithme à la fin d'une étape du tri à bulles, il faut savoir si le tableau est trié. Pour cela, écrivez un algorithme récursif **testTrieAvantCase**, qui prend en entrée un tableau d'entier *tab* et un entier *i*, et renvoie VRAI si tab est trié jusqu'à la *i*-ième case incluse, et renvoie FAUX sinon.

Indication pour l'initialisation : que renvoie **testTrieAvantCase** sur n'importe quel tableau *tab* en entrée si l'entrée *i* vaut 1 ?

Indication pour l'hérédité : si le tableau *tab* en entrée de **testTrieAvantCase** est trié jusqu'à la (*i*-1)-ième case, comment savoir s'il est trié jusqu'à la *i*-ième ? Et s'il n'est pas trié jusqu'à la *i*-ième case ?

Q4. En utilisant **etapeTriBulles** et **testTrieAvantCase**, écrivez l'algorithme **triBulles** qui prend en entrée un tableau d'entiers et renvoie ce tableau trié par le tri à bulles.

TD d'algorithmique INF220 – TD2&3 – Tris

Exercice 1 – Implémentation du tri à bulles

• Q1. La fonction **etapeTriBulles** fait une étape du tri à bulles. Comme nous l'avons vu en cours, elle parcourt le tableau d'entiers fourni en entrée en vérifiant pour chaque case si elle est bien inférieure ou égale à la suivante, et échange leurs valeurs si ce n'est pas le cas. Elle renvoie le tableau ainsi obtenu. Que renvoie **etapeTriBulles**({4,5,1,2,2,9,8}) ?

• Q2. Écrivez la fonction **etapeTriBulles**.

• Q3. Pour savoir si on arrête l'algorithme à la fin d'une étape du tri à bulles, il faut savoir si le tableau est trié. Pour cela, écrivez un algorithme récursif **testTrieAvantCase**, qui prend en entrée un tableau d'entier *tab* et un entier *i*, et renvoie VRAI si tab est trié jusqu'à la *i*-ième case incluse, et renvoie FAUX sinon.

Indication pour l'initialisation : que renvoie **testTrieAvantCase** sur n'importe quel tableau *tab* en entrée si l'entrée *i* vaut 1 ?

Indication pour l'hérédité : si le tableau *tab* en entrée de **testTrieAvantCase** est trié jusqu'à la (*i*-1)-ième case, comment savoir s'il est trié jusqu'à la *i*-ième ? Et s'il n'est pas trié jusqu'à la *i*-ième case ?

Q4. En utilisant **etapeTriBulles** et **testTrieAvantCase**, écrivez l'algorithme **triBulles** qui prend en entrée un tableau d'entiers et renvoie ce tableau trié par le tri à bulles.

TD d'algorithmique INF220 – TD2&3 – Tris

Exercice 1 – Implémentation du tri à bulles

• Q1. La fonction **etapeTriBulles** fait une étape du tri à bulles. Comme nous l'avons vu en cours, elle parcourt le tableau d'entiers fourni en entrée en vérifiant pour chaque case si elle est bien inférieure ou égale à la suivante, et échange leurs valeurs si ce n'est pas le cas. Elle renvoie le tableau ainsi obtenu. Que renvoie **etapeTriBulles**({4,5,1,2,2,9,8}) ?

• Q2. Écrivez la fonction **etapeTriBulles**.

• Q3. Pour savoir si on arrête l'algorithme à la fin d'une étape du tri à bulles, il faut savoir si le tableau est trié. Pour cela, écrivez un algorithme récursif **testTrieAvantCase**, qui prend en entrée un tableau d'entier *tab* et un entier *i*, et renvoie VRAI si tab est trié jusqu'à la *i*-ième case incluse, et renvoie FAUX sinon.

Indication pour l'initialisation : que renvoie **testTrieAvantCase** sur n'importe quel tableau *tab* en entrée si l'entrée *i* vaut 1 ?

Indication pour l'hérédité : si le tableau *tab* en entrée de **testTrieAvantCase** est trié jusqu'à la (*i*-1)-ième case, comment savoir s'il est trié jusqu'à la *i*-ième ? Et s'il n'est pas trié jusqu'à la *i*-ième case ?

Q4. En utilisant **etapeTriBulles** et **testTrieAvantCase**, écrivez l'algorithme **triBulles** qui prend en entrée un tableau d'entiers et renvoie ce tableau trié par le tri à bulles.

Exercice 2 – Tri par insertion
• Q1. Écrivez un algorithme **decaleApresCase** qui prend en entrée un tableau d'entiers *tab* et un numéro de case *i* et décale d'une case vers la droite toutes les valeurs situées après la *i*-ième case (non incluse). Que renvoie **decaleApresCase**({1,4,5,,,,},1) ?
• Q2. Écrivez un algorithme **trouvePosition** qui prend en entrée un tableau d'entiers trié *tab* (dont les dernières cases sont éventuellement vides) et un entier *k*, et qui renvoie le numéro de case de *tab* après laquelle il faut insérer *k* pour que *tab* reste trié. Par exemple, **trouvePosition**({1,4,5,,,,},2) renvoie 1 car il faudrait insérer l'entrée *k*=2 après la première case pour obtenir le tableau trié {1,2,4,5,,,}.
• Q3. En utilisant les algorithmes **decaleApresCase** et **trouvePosition**, écrivez un algorithme **insereDansTableauTrie** qui prend en entrée un tableau d'entiers trié *tab* et un entier *k*, et qui insère *k* dans *tab* à la bonne position pour que *tab* reste trié, et renvoie *tab*. Que renvoie **insereDansTableauTrie**({1,2,4,5,,,},2) ?
• Q4. En utilisant l'algorithme **insereDansTableauTrie**, écrivez l'algorithme **triParInsertion** qui prend en entrée un tableau d'entiers *tab*, crée un tableau vide *tab2*, et le remplit progressivement avec les éléments de *tab* de telle sorte que *tab2* reste toujours trié. Finalement, **triParInsertion** renvoie le tableau *tab2* qui est trié et a été entièrement rempli par les éléments de *tab*.

Exercice 2 – Tri par insertion
• Q1. Écrivez un algorithme **decaleApresCase** qui prend en entrée un tableau d'entiers *tab* et un numéro de case *i* et décale d'une case vers la droite toutes les valeurs situées après la *i*-ième case (non incluse). Que renvoie **decaleApresCase**({1,4,5,,,,},1) ?
• Q2. Écrivez un algorithme **trouvePosition** qui prend en entrée un tableau d'entiers trié *tab* (dont les dernières cases sont éventuellement vides) et un entier *k*, et qui renvoie le numéro de case de *tab* après laquelle il faut insérer *k* pour que *tab* reste trié. Par exemple, **trouvePosition**({1,4,5,,,,},2) renvoie 1 car il faudrait insérer l'entrée *k*=2 après la première case pour obtenir le tableau trié {1,2,4,5,,,}.
• Q3. En utilisant les algorithmes **decaleApresCase** et **trouvePosition**, écrivez un algorithme **insereDansTableauTrie** qui prend en entrée un tableau d'entiers trié *tab* et un entier *k*, et qui insère *k* dans *tab* à la bonne position pour que *tab* reste trié, et renvoie *tab*. Que renvoie **insereDansTableauTrie**({1,2,4,5,,,},2) ?
• Q4. En utilisant l'algorithme **insereDansTableauTrie**, écrivez l'algorithme **triParInsertion** qui prend en entrée un tableau d'entiers *tab*, crée un tableau vide *tab2*, et le remplit progressivement avec les éléments de *tab* de telle sorte que *tab2* reste toujours trié. Finalement, **triParInsertion** renvoie le tableau *tab2* qui est trié et a été entièrement rempli par les éléments de *tab*.

Exercice 2 – Tri par insertion
• Q1. Écrivez un algorithme **decaleApresCase** qui prend en entrée un tableau d'entiers *tab* et un numéro de case *i* et décale d'une case vers la droite toutes les valeurs situées après la *i*-ième case (non incluse). Que renvoie **decaleApresCase**({1,4,5,,,,},1) ?
• Q2. Écrivez un algorithme **trouvePosition** qui prend en entrée un tableau d'entiers trié *tab* (dont les dernières cases sont éventuellement vides) et un entier *k*, et qui renvoie le numéro de case de *tab* après laquelle il faut insérer *k* pour que *tab* reste trié. Par exemple, **trouvePosition**({1,4,5,,,,},2) renvoie 1 car il faudrait insérer l'entrée *k*=2 après la première case pour obtenir le tableau trié {1,2,4,5,,,}.
• Q3. En utilisant les algorithmes **decaleApresCase** et **trouvePosition**, écrivez un algorithme **insereDansTableauTrie** qui prend en entrée un tableau d'entiers trié *tab* et un entier *k*, et qui insère *k* dans *tab* à la bonne position pour que *tab* reste trié, et renvoie *tab*. Que renvoie **insereDansTableauTrie**({1,2,4,5,,,},2) ?
• Q4. En utilisant l'algorithme **insereDansTableauTrie**, écrivez l'algorithme **triParInsertion** qui prend en entrée un tableau d'entiers *tab*, crée un tableau vide *tab2*, et le remplit progressivement avec les éléments de *tab* de telle sorte que *tab2* reste toujours trié. Finalement, **triParInsertion** renvoie le tableau *tab2* qui est trié et a été entièrement rempli par les éléments de *tab*.

Exercice 2 – Tri par insertion
• Q1. Écrivez un algorithme **decaleApresCase** qui prend en entrée un tableau d'entiers *tab* et un numéro de case *i* et décale d'une case vers la droite toutes les valeurs situées après la *i*-ième case (non incluse). Que renvoie **decaleApresCase**({1,4,5,,,,},1) ?
• Q2. Écrivez un algorithme **trouvePosition** qui prend en entrée un tableau d'entiers trié *tab* (dont les dernières cases sont éventuellement vides) et un entier *k*, et qui renvoie le numéro de case de *tab* après laquelle il faut insérer *k* pour que *tab* reste trié. Par exemple, **trouvePosition**({1,4,5,,,,},2) renvoie 1 car il faudrait insérer l'entrée *k*=2 après la première case pour obtenir le tableau trié {1,2,4,5,,,}.
• Q3. En utilisant les algorithmes **decaleApresCase** et **trouvePosition**, écrivez un algorithme **insereDansTableauTrie** qui prend en entrée un tableau d'entiers trié *tab* et un entier *k*, et qui insère *k* dans *tab* à la bonne position pour que *tab* reste trié, et renvoie *tab*. Que renvoie **insereDansTableauTrie**({1,2,4,5,,,},2) ?
• Q4. En utilisant l'algorithme **insereDansTableauTrie**, écrivez l'algorithme **triParInsertion** qui prend en entrée un tableau d'entiers *tab*, crée un tableau vide *tab2*, et le remplit progressivement avec les éléments de *tab* de telle sorte que *tab2* reste toujours trié. Finalement, **triParInsertion** renvoie le tableau *tab2* qui est trié et a été entièrement rempli par les éléments de *tab*.