CS353 - AlgorithmiqueGrille de rendu du TP 1

Owen Rougé et Adrian Bonnet

*struct Client *createClient(int numero, int prixAppel);*

Code source de la fonction:

```
{
        client* liste;
        if ( (liste = malloc(sizeof(client)) ) == NULL) return NULL;
        liste->numero = numero;
        liste->prixAppel = prixAppel;
        liste->nbAppel = 1;
        liste->suivant = NULL;
        return liste;
}
```

****

n'ayant pas d'utilité pour le nombre d'appel lors de la création d'un nouvel élément dans la chaîne celui ci a été retiré.

*int addLogLine(struct Client** list ,int numero, int prixAppel) ;*

Code source de la fonction:

```
{
        client* maliste = *liste;
        client* nouvelelem;
        if  (maliste == NULL) {
                if ( ( *liste = createClient(numero,prixAppel) ) == NULL){
                        printf("echec du a un manque de place");
                        return 0;
                }
                else {
                        return 1;
                }
        }
        else {
                if( maliste->numero == numero) {
                                maliste->prixAppel += prixAppel;
                                maliste->nbAppel ++;
```

	}
	else{
	if(maliste->numero > numero){
	if((nouvelelem = createClient(numero,prixAppel)) == NULL) return 0;
	nouvelelem->suivant = *liste;
	*liste = nouvelelem;
	}
	else{
	while(maliste->suivant != NULL && (maliste->suivant)->numero <= numero) maliste = maliste->suivant;
	if(maliste->numero == numero) {
	(*liste)->prixAppel += prixAppel;
	(*liste)->nbAppel ++;
	}
	else{
	if ((nouvelelem = createClient(numero,prixAppel)) == NULL) return 0;
	nouvelelem->suivant = maliste->suivant;
	maliste->suivant = nouvelelem;
	}
	}
	}
	}
	return 1;
	}

void dumpList(struct Client list) ;*	

Code source de la fonction:	{
	while (liste != NULL) {
	printf("numero : %d\tprix des appels : %d\tnombre d'appels : %d\n",liste->numero,liste->prixAppel,liste->nbAppel);
	liste = liste->suivant;
	}

Temps d'exécution pour différentes valeurs et conclusion (gardez un rapport 100 entre les 2 constantes)	
NBCLIENT 2000 NBLOGLINE 200000	**0.828 sec**
NBCLIENT 20000 NBLOGLINE 2000000	**214.453003 sec**
NBCLIENT 200000 NBLOGLINE 20000000	**temps d'exécution trop long**

complexité	**o (NBLOGLINE * NBCLIENT)**
Conclusion sur le TP:	**Bien que le problème posé soit en apparence simple, sa résolution par un algorithme est d'une complexité élevée** **Ce temps d'exécution si important est peut-être dû en partie à l'utilisation d'une liste chaînée, mais une gestion de données à accès plus rapide ne diminuerait pas la complexité de l'algorithme, seulement le temps d'exécution de chaque instruction.** **Pour réduire drastiquement le temps d'exécution, il faut penser un nouvel algorithme à complexité moindre**