TP MP2I : Rencontre au milieu

Présentation

Dans ce TP on implante l'algorithme de « rencontre au milieu » vu en cours.
Un `Makefile` et un fichier d'en-tpete `tools.h` sont notamment fournis dans l'archive.

1 Outils

On écrit deux fichiers `tools.c, test_tools.c`. Le fichier d'entête `tools.h` mentionne les fonctions `affiche, triFusion, dichot` et le type `ll` qui est un alias du type `long long`. Les tests et le `main` se trouvent dans `test_tools.c`.

La fonction

```
void triFusion(int i, int j, ll tab[], ll tmp[]);
```

implante le tri fusion d'un tableau `tab` d'entiers entre les indices `i,j`. Le tableau `temp` est un tableau auxiliaire qui sert à la fusion des sous-parties triées de `tab`.

Q.1 Dans `tools.c` écrire

```
int dichot(ll T[], ll x, int n);
```

Cette fonction effectue la recherche dichotomique du plus petit nombre $e \in T[:n]$ tel que $e \geq x$. Elle retourne la position correspondante. Le tableau `T` est supposé trié dans l'ordre croissant.

Contrairement à la dichotomie classique, on ne renvoie pas -1 si on ne trouve pas x. Il faudra donc interpréter par la suite le résultat obtenu.

Q.2 Ecrire des tests dans le fichier `test_tools.c`.

Indication. Chercher dans le tableau un élément plus grand que tous ceux du tableau ; un élément plus petit ; un élément dans le tableau ; un élément entre les bornes du tableau etc.

2 Meet in the middle

On écrit le code dans un fichier `meet.c`. Le projet utilise les fonctions outils présentées plus haut. On crée deux tableau globaux :

```
ll X[2000005],Y[2000005];
```

Ces tableaux contiendront les sommes des sous-ensembles constitués respectivement avec la première partie puis la seconde de l'ensemble de nombres étudié.

Les tests dans les `bench` se font avec

```
ll a[] = {3, 34, 4, 12, 5, 2};
```

Q.3 Ecrire la procédure

```
void tabsommes(ll a[], ll x[], int n, int c)

```

Cette procédure met dans le tableau `x` toutes les sommes calculées avec les éléments de `a`

— de l'indice 0 à l'indice $n-1$ 1i $c=0$;

— de l'indice n à l'indice $c+n-1$ si $c \neq 0$.

Indication. — Utiliser l'opérateur bitwise `<<` pour calculer des puissances de deux.

— Le codage binaire d'un nombre $i \leq N$ écrit sur N bits représente de façon bijective un sous-ensemble de $[\![0, N-1]\!]$.

— Utiliser l'opérateur bitwise `&` et le précédent pour obtenir la valeur du bit j de l'entier i.

Q.4 Ecrire une fonction de tests `void testtabsomme()` dont le comportement est le suivant :

```
--------- testtabsomme ----------
a=[3, 34, 4, 12, 5, 2]
Sommes 1er moitié X=[0, 3, 34, 37, 4, 7, 38, 41]
Sommes 2nd moitié Y=[0, 12, 5, 17, 2, 14, 7, 19]
--------- FIN testtabsomme ----------
```

Q.5 Ecrire la fonction

```
ll meetInTheMiddle(ll a[], int n, ll S)
```

qui implante l'algorithme de rencontre au milieu.
Compléter par une fonction de tests :

```
a=[3, 34, 4, 12, 5, 2]
partie triée Y : [0, 12, 5, 17, 2, 14, 7, 19]
Plus grande valeur inférieure ou égale à 10 :  10
```