Université de Savoie, UFR SceM
INFO601_CMI, L3 INFO

27 novembre 2025

TD 2, Analyse des tables dynamiques

Nous nous intéressons aux tables dynamiques, c'est-à-dire aux structures stockant des éléments de manières consécutives, mais qui ne connaissent pas *a priori* le nombre d'éléments qu'elles auront à stocker. En général, et ce quel que soit le type de table considéré, le principe suivi est de considérer un facteur de remplissage de la table ($\alpha(U)$ si U est la table) qui est la proportion des cellules réellement utilisées dans la table. Lorsque le facteur de remplissage s'approche trop de 1, on alloue une nouvelle table deux fois plus grande dans laquelle on recopie tous les éléments, et l'ancienne table est désallouée. Inversement, si la table se vide d'éléments, on peut aussi allouer une nouvelle table plus petite dans laquelle on remet les éléments et on désalloue l'ancienne table pour gagner de la place mémoire.

La question est donc de savoir si nos fonctions Insérer-Table et Supprimer-Table ont un coût amorti faible ou non. On note $num(U)$ le nombre d'éléments utiles stockés dans U et $taille(U)$ la taille réelle allouée de U.

Exercice 1. Fonction Insérer-Table

Voilà le pseudo-code de Insérer-Table, dans le cas où on double la taille lorsque $\alpha(U) = 1$.

```
typedef ... Element;
struct STable {
  Element* T;
  int      num;
  int      taille;
};
typedef struct STable Table;

int num( Table* U ) { return U->num; }
int taille( Table* U ) { return U->taille; }

void CreerTable( Table* U ) {
  U->T      = (Element*) malloc( sizeof(Element) );
  U->num    = 0;
  U->taille = 1;
}
void InsererTable( Table* U, Element e ) {
  if ( num(U) == taille(U) ) {
    Element* T2 = (Element*) malloc( 2*taille(U)*sizeof(Element) );
    for ( int i = 0; i < num(U); i++ )
      T2[ i ] = U->T[ i ];
    free( U->T );
    U->T = T2;
  }
  U->T[ U->num ] = e;
  U->num        += 1;
}
```

Quelle est la complexité en pire de cas de cette fonction, si $n = num(U)$?

La complexité est $O(n)$ lorsque $\alpha(U)=1$, sinon $O(1)$.

Exercice 2. Coût amorti de Insérer-Table

On considère une séquence de n appels à Insérer-Table. Montrer que le coût total $T(n)$ est inférieur à $3n$ par la méthode de l'agrégat.

```
La borne supérieure est le nombre total d'écriture mémoire.
Le pire cas est lorsqu'on vient de doubler la taille, donc n=2^k+1.
Alors le total des écritures mémoires est:
2^0+2^1+...+2^k + (2^k+1)= 3n
```

Exercice 3. Pourquoi $3n$?

Utilisez maintenant la méthode comptable pour calculer le coût amorti. Dans le coût amorti placé sur chaque élément, à quoi correspond chaque crédit mis sur l'élément ?

```
%% 1 pour insérer l'élément lui-même.
%% 1 pour se déplacer au prochain agrandissement
%% 1 pour déplacer un élément de l'autre moitié.
```

Exercice 4. Méthode du potentiel pour calculer le coût amorti de Insérer-Table

Déterminer maintenant une fonction Φ simple telle que Φ est toujours positive, vaut 0 juste après un agrandissement, et vaut $num(T)$ juste avant (ce qui paiera le coût du déplacement). Déterminer le coût amorti de Insérer-Table en sommant coût réel plus $\Phi(U_i) - \Phi(U_{i-1})$.

```
%% Phi(U)=2*num(U)-taille(U)
%% si pas extension:
%%   ca_i = 1 + 2*num(U_i)-taille(U_i)-(2*num(U_{i-1})-taille(U_{i-1})
%%   ca_i = 1 + 2 = 3
%% si extension num(U_i-1)=taille(U_i-1)=2^m
%%   ca_i = 1 + 2^m + 2*(2^m+1) - 2^(m+1) - (2*2^m - 2^m)
%%   ca_i = 1 + 2^m + 2 - 2^m = 3
```

Exercice 5. Contraction d'une table

On souhaite minorer le facteur de remplissage par une constante. On souhaite bien sûr garder le coût amorti de n opérations Insérer-Table ou Supprimer-Table en $\mathcal{O}(1)$.

Est-ce une bonne idée de contracter la table dès lors que $num(U) < taille(U)/2$? Donnez alors une séquence d'opérations qui prendrait un temps $\Theta(n^2)$.

Exercice 6. Choix du facteur de remplissage pour contracter

On propose de prendre un facteur de remplissage de $\frac{1}{4}$ pour décider de contracter. Cela donnera le temps à $n/4$ suppressions de payer pour la contraction. Il faut redéfinir la fonction potentiel Φ lorsque le facteur de remplissage est plus petit que $1/2$. Faites en sorte là aussi que $\Phi(U) = num(U)$ lorsque le facteur de remplissage est $1/4$. Du coup, on verra que le potentiel pourra payer la contraction.

Déterminer alors les coûts amortis de Insérer-Table ou Supprimer-Table avec ce nouveau Φ.