Cours

- Définition d'une liste. Liste doublement chainée. D'un tableau. Cout des opérations de création, ajout, supression en temps et mémoire.
- Différence entre struture immuable et mutable.
- Définition d'une pile. Implémentation d'une pile par tableau dynamique.
- Définition d'une file. Implémentation par une liste doublement chainé, et par un tableau dynamique.
- Table de Hachage, tableau associatif.
- Hachage: Collision, densité, structure hachable (HP)
- Sérialisation d'un tableau ou d'une table de hashage (ou d'un arbre).

Complexité des opérations de bases

Pour chacune des structure entre tableau, liste, liste doublement chainée, tableau dynamique, pile, file, que l'on considèrent toutes de longeur n indiquer les complexitées au pire des cas, meilleurs des cas et amortis des opérations suivantes :

1. Lire le premier élément.
2. Lire le dernier élément.
3. Ajouter un élément à la fin.
4. Ajouter un élément au début.
5. Retirer le premier élément.
6. Retirer le dernier élément.
7. Inverser l'ordre.
8. Tester si il y a un 42.
9. Insérer un élément au milleu.

Merge de liste

Soit L_1, L_2 deux listes triées. Proposer un algoritme en OCaml qui calcule la liste trié des éléments contenue dans L_1 et L_2.

Calcul des différences

Proposer un algorithme qui prend un tableau d'entiers T de longueur N et qui en $O(n)$ renvoie $\sum_{0 \leq j < i < n} T[i] - T[j]$

On peut facilement transformer cet exercice en un autre en changeant la formule

Permutation suivante

On représente une permutation σ comme un tableau T de longueur n tel que $\forall i < N, T[i] = \sigma(i)$

Donner le code d'une fonction qui à une permutation de $[\![0; N]\!]$ représenté par T retourne la prochaine permutation de $[\![0; N]\!]$ dans l'ordre lexicographique.

Tableaux binaire

On considère le problème suivant:

Minimum change binary matrix
Entrée: Une matrice $M \in \mathcal{M}_{n,m}(\{0,1\})$
Sortie: Le nombre minimum de coefficients à changer pour que chaque ligne et chaque collone ait un nombre pair de 1.

Donner un algorithme résolvant ce problème en $O(NM)$ avec N, M les dimensions du tableau.
Montrer sa correction.

Sous-tableau connexe

Soit A un tableau de n entiers relatifs, on cherche un algorithme en $O(n)$ qui calcule le sous-tableau connexe qui maximise la somme de ses éléments, i.e. le couple (i, j) avec $0 \leq i \leq j \leq |T|$ tel que $\sum_{k \in [\![i;j]\!]} A[k]$ soit maximal.

Le glouton par défault[1]

Étant donné un ensemble $\{x_1, ..., x_n\}$ de n points sur une droite, décrire un algorithme qui détermine le plus petit ensemble d'intervalles fermés de longueur 1 qui contient tous les points donnés.

Prouver la correction de votre algorithme et donner sa complexité.

[1] Algo 1 ENS Lyon

Recherche dans une matrice

Question 1 Donner le code C d'une fonction `int find(int *arr, int n, int key);` qui prend en argument un tableau `arr` trié de longueur `n` et qui trouve un indice `i` tel que `arr[i] == key`.

On cherche à faire un algorithme efficace aussi dans le cadre de matrice trié. On dit qu'une matrice M de taille $n \times m$ est *linéairement trié* si chaque ligne est trié de haut en bas et que chaque colone est trié de gauche à droite.

Question 2 Donner le code d'un fonction C `bool is_ordered(int **mat, int n, int m)` qui à une matrice `mat` de taille $n \times m$ teste si elle est linéairement trié.

Question 3 Donner le code d'une fonction `is_in(int **mat, int n, int m, int key)` qui teste si `key` est présent dans la matrice `mat` linéairement trié de taille $n \times m$ par diviser pour reigner.

Question 4 Meme question en $O(n + m)$

Question 5 Donner le code d'une fonction C qui en $O(n)$ trouve un minimum local dans une matrice $n \times n$

Tableau cumulatif

Soit T un tableau de n entiers relatifs, on note C_T le tableau de longueur N tel que

$$C_T[i] = \sum_{j=0}^{i} T[j]$$

Question 1 Donner le code de la fonction `int* get_cumulatif(int* T, int n);` qui, à un tableau T de longueur N associe son tableau cumulatif.

Question 2 Montrer par récurrence sur la longueur de T que si $\max C_T > N$ avec N la longueur du tableau alors il existe un élément i avec $T[i] > 1$.

On cherche maintenant $s \leq t \leq N$ tel que $\sum_{i=s}^{t} T[i]$ soit maximale

Question 3 Proposer un algorithme qui, étant donné un tableau T, renvoie $i < j$ tel que $T[j] - T[i]$ soit maximal. En déduire un code C qui répond au problème en utilisant un tableau cumulatif.

Question 4 On cherche maintenant à calculer le nombre de couples (i, j) avec $i < j$ tel que $\sum_{i=s}^{t} T[i]$ soit maximale. Proposer un algorithme répondant au problème.

Multiplication rapide de polynome[2]

Ici on considère des polynômes d'entiers $\mathbb{Z}[X]$. Soient $P, Q \in \mathbb{Z}_n[X]$, leur produit $R = PQ \in \mathbb{Z}_{2d}[X]$

On représente un polynome par un tableau d'entiers T, tel que pour $P \in \mathbb{Z}_d[X]$, si on écrit $P = \sum_{i \leq d} a_i X^i$ on a $T[i] = a_i$.

1. Donner une fonction C `int* multiply(int* p, int* q, int n)` qui prend deux polynômes de degré $\leq n$ représenter par deux tableaux de longueur n et qui renvoie le polynome produit.
2. Soit $n = 2m$, pour $P \in \mathbb{Z}_n[X]$, montrer que on peut décomposer $P = P_1 + X^m P_2$ avec $P_1, P_2 \in \mathbb{Z}_m[X]$.

Soient $P, Q \in \mathbb{Z}_n[X]$, on décompose $P = P_1 + X^m P_2$ et $Q = P_1 + X^m P_2$. On définit alors $R_1 = P_1 Q_1$, $R_2 = P_2 Q_2$ et $R_3 = (P_1 + P_2) \times (Q_1 + Q_2)$.

3. Exprimer $P \times Q$ en fonctions de R_1, R_2, R_3.
4. En déduire un algorithme récursif pour calculer le produit de polynome.
5. Quelle est la complexité de cet algorithme ?

[2] Algo 1 ENS Lyon

Structure efficace pour représenter des ensembles

On cherche à implémenter une structure de données pour représenter un sous-ensemble de $[\![0; N]\!]$ (N sera fixé par la fonction `create`). Pour cela, on chercherait à définir une structure avec 3 opérations:

- `val create: int -> set` telle que `create N` renvoie $\emptyset$ (un sous-ensemble de $[\![0; N]\!]$)
- `val add: set -> int -> unit` telle que `add x i` renvoie $X \cup \{i\}$ (si $0 \leq i \leq N$)
- `val del: set -> int -> unit` telle que `del x i` renvoie $X \setminus \{i\}$ (si $0 \leq i \leq N$)

1. Proposer une implémentation de cette structure en utilisant des listes. Quelles sont les complexités des différentes opérations en mémoire et en espace ?

On cherche à avoir une complexité spatiale aussi petite que possible. Attention, à partir de maintenant, on prendra en compte la taille des entiers. On rappelle qu'un entier N à une taille de mémoire $\log_2(N)$.

2. On considère la représentation qui à $1 \leq a_1 \leq ... \leq a_m \leq n$ associe la liste

$$[a_1, a_2 - a_1, a_3 - a_2, ..., a_m - a_{m-1}]$$

Montrer que cette représentation est en $O(n)$ de mémoire.

3. Implémenter les fonctions `add` et `del` pour cette représentation. Quels sont les complexités ?
4. Proposer une implementation tel que `add` et `del` soient en $O(1)$ en complexité, mais que la structure soit toujours en $O(N)$ d'espace. La fonction `create` peut-être en $O(N)$.
5. Peut-on faire mieux en complexité spatiale que $O(N)$?

Liste cyclique sans lièvre et torture

1. Proposer une structure pour implémenter une liste simplement chainé en C.

Remarquer que le pointeur "next" pourrait être n'importe quel maillon de la liste, y compris un des éléments précédemment vu. On dit qu'une liste est cyclique si un pointeur "revient" sur un des maillons précédent. Dans ce cas, la liste représenté est infinie et, à partir d'un certain rang, devient périodique.

2. Proposer un code C pour définir la liste cyclique qui représente $[1, 2, 3, 2, 3, \ldots]$ en n'utilisant que 3 maillons.
3. Proposer un algorithme pour tester si une liste est cyclique. On fera bien attention à ce que l'algorithme termine.
4. Donner une fonction `liste_t* reverse(liste_t* list);` qui inverse une liste simplement chainé en C. Que ce passe-t'il si on inverse une liste cyclique ?
5. On cherche maintenant à calculer le nombre de maillons d'une liste cyclique. Proposer un algorithme en $O(n)$ de temps et $O(1)$ de mémoire.

Info bonus: Sachez que l'on peut définir de telle listes en OCaml ! Le code suivant définit la liste de la question 2 : `let q2 = 1 :: (let rec l = 2 :: 3 :: l in l)`

Tableaux auto-référents

Pour T un tableau de longueur n, on définit l'histogramme de T noté H_T comme le tableau de longueur n tel que $H_T[i]$ correspond au nombre de i dans T. Formellement, $H_T[i] = |\{j : T[j] = i\}|$

On dit que T est auto-référent si $H_T = T$. Par exemple, $[1, 2, 1, 0]$ est un tableau autoréférent.

Question 1 Donner le code d'une fonction `int* histogramme(int *t, int n);` qui, à un tableau t de longueur n renvoie H_t son histogramme.

Question 2 Donner tous les tableaux auto-référent de longueur 4 et 5

Question 3 Donner le code d'une fonction `bool is_autoref(int *t, int n);` qui teste si un tableau t est auto-référent.

Question 4 Donner le code d'une fonction de recherche exhaustive `int count_autoref(int n);` qui donne le nombre de tableaux auto-référents de longueur n.

Question 5 Montrer que pour tout $n > 6$ il existe un tableau autoréférent de longueur n

Question 6 $(*)$ Montrer qu'il est unique