I21 - Algorithmique élémentaire
TP 6

Semestre 2 2018/2019

Exercice 1. Recherche dichotomique

Le but est de comparer l'efficacité de deux approches pour la recherche dans un tableau trié: la recherche binaire (ou recherche par dichotomie) et la recherche ternaire, qui consiste à découper le tableau non pas en deux mais en trois parties pour sélectionner le sous tableau dans lequel se trouve l'élément recherché.

1. Écrire une fonction `gen_tableau(n)` qui génère un tableau trié de taille `n` de nombre aléatoires entre `1` et `2n`.
2. Écrire deux fonctions `recherche_binaire(T,x)` et `recherche_ternaire(T,x)` retournant l'indice de l'élément `x` dans le tableau `T` s'il s'y trouve et -1 sinon, ainsi que le nombre de comparaisons effectuées.
3. Écrire deux fonctions `complexité_binaire(T)` et `complexité_ternaire(T)` qui retourne la moyenne du nombre de comparaisons effectuées pour rechercher chacun des nombres chacun des nombres du talbeau `T`.
4. Tracer à l'aide de `matplotlib` des courbes de complexité moyenne des deux algorithmes en fonction de la taille des tableaux. Quel est la méthode la plus efficace ?

Exercice 2. Zéro d'une fonction

Le but de cet exercice est de programmer un algorithme classique d'analyse numérique: la recherche du zéro d'une fonction continue s'annulant une seule fois sur un intervalle. Plus précisément, on recherche un intervalle dans lequel ce trouve le zéro d'une taille fixée à l'avance.

1. Écrire une fonction `recherche_zero(a,b,prec)` qui recherche par dichotomie un intervalle de taille `prec` contenant le zero de la fonction `sin` définie sur l'intervalle `[a,b]`.
2. Tester votre programme sur l'intervalle $[-2.3, 0.5]$ avec une précision de 10^{-2}.
3. Tracer avec `matplotlib` la courbe précédente en faisant apparaître en rouge l'intervalle où se trouve le zéro.

Exercice 3. Maximum d'une fonction

Cette exercice repose sur un problème similaire au précédent. Ici on considère une fonction continu strictement croissante puis strictement décroissante sur un intervalle donnée. Le but est de trouvé un intervalle où se trouve le maximum de la fonction.

1. Écrire une fonction `recherche_max(a,b,prec)` qui recherche un intervalle de taille `prec` contenant le maximum de la fonction `sin` définie sur l'intervalle `[a,b]`.

   (Indication: étant donnés deux point x_1 et x_2, quelle est la position du maximum en fonction de l'ordre relatifs de $f(x_1)$ et $f(x_2)$?)

2. Tester votre programme sur l'intervalle $[0.8, 3]$ avec une précision de 10^{-2}.

3. Tracer avec `matplotlib` la courbe précédente en faisant apparaître en rouge l'intervalle où se trouve le maximum.

Exercice 4. Challenge: jeu du pendu

Le but de ce challenge est d'écrire un programme d'aide au jeu du pendu. Le programme `pendu.py` permet de jouer contre la machine au jeu du pendu en utilisant l'ensemble des mots contenus dans le fichier `dico-fr.txt` avec des règles légèrement modifiées: le but est de trouver le mot secret en un minimum d'essaies et le joueur peut proposer un mot après chaque proposition de lettre. Par exemple:

```
Choisir la longueur du mot a trouve: 10
Faire un proposition de mot: (o)ui/(n)on ?n
Proposer une lettre: o
****o*****
Faire un proposition de mot: (o)ui/(n)on ?n
Proposer une lettre: n
****o*****
Faire un proposition de mot: (o)ui/(n)on ?n
Proposer une lettre: t
****o*****
Faire un proposition de mot: (o)ui/(n)on ?n
Proposer une lettre: u
****ou****
Faire un proposition de mot: (o)ui/(n)on ?n
Proposer une lettre: i
****ou**i*
Faire un proposition de mot: (o)ui/(n)on ?n
Proposer une lettre: e
****ou**ie
Faire un proposition de mot: (o)ui/(n)on ?b
Proposer une lettre: b
*b**ou**ie
Faire un proposition de mot: (o)ui/(n)on ?o
Proposition: abasourdie
Gagne
```

Votre programme doit donc vous aider à choisir à chaque tour quelle lettre proposer et s'il ne reste qu'un mot possible, vous donner le mot secret. Le programme `AI_pendu.pyc` est un exemple du résultat attendu, il peut être exécuté comme un script Python standard avec la commande `python3 AI_pendu.pyc`.