Test d'embauche technique

Introduction

Le but de ce test est de mettre à l'épreuve, au-delà des technologies existantes, votre capacité d'abstraction et vos connaissances sur les structures élémentaires utilisées en informatique.

L'intégralité des implémentations doivent être réalisées en `nodejs`.

Bon courage!

Questions

Structure élémentaires

Question 1.1 : Itération

Implémentez une fonction générant une suite de nombre allant de 0 à `N` par pas de `p`. Si le nombre généré est un multiple de `a`, vous afficherez `foo`, un multiple de `b` vous afficherez `mit`, un multiple de `a` et `b` vous afficherez `buzz`.

Question 1.2 : Produit matriciel

Une matrice réelle A à n lignes et p colonnes s'écrit:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1p} \\ a_{21} & a_{22} & \dots & a_{2p} \\ \dots & & & \dots \\ a_{n1} & \dots & \dots & a_{np} \end{pmatrix}$$

Le produit d'une matrice par un vecteur s'écrit :

$$y = Ax, y_i = \sum_{j=1}^{p} a_{ij} x_j$$

Implémentez ce produit matriciel et donnez sa complexité.

Question 1.3 : Table de hash

Implémentez un annuaire téléphone en utilisant une table de hachage.

Question 1.4 : Parcours de graphe

Soit le graphe suivant :

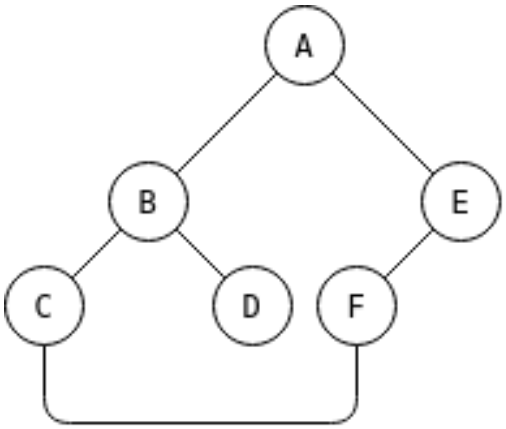

Figure 1: Graphe non oriente

Implémentez un parcours en profondeur du graphe ci-dessus puis donnez la complexité de votre algorithme.

Sécurité

Question 1.5.1

Soit le code `C` suivant :

```c
#include<stdio.h>

int main()
{
    for (int i=0; i<10000000; i++)
    {
       int *ptr = (int *)malloc(sizeof(int));
    }
}
```

Quel problème risque de poser l'exécution de ce dernier ?

Question 1.5.2

Qu'est-ce qu'une *rainbow table* et comment s'en protéger ?

Abstraction

Implémentez au choix la question `2.1` ou `2.2`.

Question 2.1 : Décorateur

Implémentez le *decorator design pattern*.

Question 2.2 : Observable

Implémentez l'*observer design pattern*.

Technologies

Question 3.1 : `Docker`

1. Citez les différences majeure entre `docker` et un `virtualbox`.
2. Faut il augmenter ou réduire au maximum le nombre de *layers* d'une image docker ?
3. Citez trois technique pour réduire la surface d'attaque du *docker deamon*.

Question 3.2 : `docker-compose`

Écrivez un `docker-compose.yml` fonctionnel mettant en œuvre une image `nodejs` et une base de donnée `mongoDB`. Vous devrez ouvrir les ports `8000` du `host` vers le port `80` de l'image `nodejs`.

Question 3.3 : Express, Graphql

En utilisant votre `docker-compose.yml` de la question précédente, mettant en place une `api` avec `expressjs` et `graphql`.

Le but ici est de développer une `api` permettant a des utilisateurs de communiquer via un chat. L'utilisation de `websocket` est un plus. Vous êtes libre de structurer votre application comme bon vous semble. Les messages doivent être sauvegardés dans la base de données `MongoDB`.

Une seule route est autorisée : `/graphql`.

Question 3.4 : ReactJs

En vous basant sur la question précédente, développez un client en `ReactJs` permettant aux utilisateurs d'utiliser votre messagerie.