注意: 現在回答を受け付けていません。

INIAD CS概論II 期末試験

I. アルゴリズム (計40点)

1. (各5点)

長さ$n > 0$の整数のリスト `lst` と整数xを引数とし、1個の整数を返す関数 `f(lst, x)` の計算量が $O(\log n)$ 、 `g(lst, x)` の計算量が $O(n)$ であるとするとき、以下のpythonの関数の計算量をオーダー記法で記述しなさい。

ただし、for と rangeを用いた繰り返しではfor文内の処理の合計の時間がかかるものとする。リスト `lst` の長さを返す関数 `len(lst)` は定数時間($O(1)$)かかるものとする。

a)

```
def func1(lst):
    result = []
    for i in range(10000):
        for j in lst:
            x = g(lst, i + j)
            result.append(x)
    for i in lst:
        x = f(lst, i)
        result.append(x)
    return result
```

- ◯ –
- ◯ $O(n)$
- ● $O(n^2)$
- ◯ $O(n^3)$
- ◯ $O(\log n)$
- ◯ $O(n \log n)$
- ◯ $O(n^2 \log n)$
- ◯ $O(n^3 \log n)$
- ◯ $O(2^n)$
- ◯ $O(1)$

b)

```
def func2(lst):
    result = 0
    n = len(lst)
    for i in lst:
        for j in lst:
            result += n * f(lst, i * g(lst, j))
    return result
```

- ○ –
- ○ $O(n)$
- ○ $O(n^2)$
- ● $O(n^3)$
- ○ $O(n^4)$
- ○ $O(n \log n)$
- ○ $O(n^2 \log n)$
- ○ $O(n^3 \log n)$
- ○ $O(n^4 \log n)$
- ○ $O(1)$

2.

a) (完答5点)

5種類の文字 `N`，`P`，`R`，`T`，`V` からなる文字列がある。それぞれの文字の出現頻度は、 `N` が30%，`P` が 12%、 `R` が19%、 `T` が21%、 `V` が 18% である。この文字列から生成できるハフマンコードのうち、 `N`，`R`，`V` のコードを書け。ただし、 `P` には `000`，`T` には `11` を割り当てるものとする。

- N
  ```
  01
  ```
- R
  ```
  10
  ```
- V
  ```
  001
  ```

b) (5点)

生成されたコードで上記の文字列をエンコードすると、各文字に3bitずつ割り当てた場合の何%の長さとなるか。整数で答えよ(必要ならば%の小数第一位を四捨五入すること)。

```
77
```

3.

a) (5点)

10000個の要素(整数)が格納されたハッシュテーブルから、ある値を検索したところ 0.12 ms かかった。 この時、要素の数が 1000000 になった場合、一つの値の検索に何 ms かかると考えられるか。 ハッシュテーブルのサイズには十分な余裕があり、実行時間が平均計算量に比例すると仮定して、見積れ。 最も近い値を以下から選ぶこと。

- ○ –
- ○ 0.01ms
- ● 0.12ms
- ○ 0.18ms
- ○ 0.24ms
- ○ 1.2ms
- ○ 1.8ms
- ○ 2.4ms
- ○ 12ms
- ○ 18ms
- ○ 24ms

b) (完答5点)

長さ10の配列を用いて実装されたハッシュテーブルによる集合に、以下の整数を順に追加することを考え、以下の問いに答えよ。衝突が起きた場合には、その整数をスキップするものとする。ハッシュ関数は以下を用いること。

```
def h(x):
    return x % 10
```

整数: 8, 17, 99, 3, 29, 80, 54, 65, 38, 11

1)

最終的に整数が格納されていないのは、配列の何番目か(ただし、配列の先頭は0番目とする)。小さい順にカンマで区切ってすべて答えよ。(例: 0, 1, 2)

```
2, 6
```

2)

2回目に衝突が起きる整数を答えよ。

```
38
```

4.

空の二分探索木に、以下の数値を順に挿入する。なお、値がxであるノードについて、その(存在するならば)左の子の値はxよりも小さく、(存在するならば)右の子の値はxよりも大きいものとする。

- 10, 5, 2, 12, 16, 8, 18, 7, 13, 9

a) (完答5点)

この時、最終的に得られる二分探索木の葉について、それらの値を小さい順に書くこと。

- 最も小さい値

```
2
```

- 2番目に小さい値

```
7
```

- 3番目に小さい値

```
9
```

- 4番目に小さい値

```
13
```

- 5番目に小さい値

```
18
```

b) (5点)

上の二分探索木から、次の値を順に削除する。

- 16, 10, 12

この時、最終的に得られる二分探索木の根の値を答えよ。

```
13
```

II. Web (計40点)

レシピ集を管理する Django のプログラムについて、以下の各問に答えよ。

1)

このプログラムには、 `cookbook` というアプリケーションが含まれており、以下に示すURLパターンが定義されている。

```
from django.contrib import admin
from django.urls import path
from cookbook import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('recipes/', views.list, name='list_recipe'),
    path('recipes/<int:recipe_id>', views.show, name='show_recipe'),
    path('recipes/<int:recipe_id>/review', views.review, name='review
_recipe'),
    path('recipes/<int:recipe_id>/yummy', views.yummy, name='yummy_re
cipe'),
    path('recipes/<int:recipe_id>/yucky', views.yucky, name='yucky_re
cipe'),
]
```

このプログラムをPC上で動かした時、以下の各URLにアクセスするときに呼び出される関数名を答えよ。

- モジュール名部分 (`views.` や `cookbook.` など) は省略して `detail` などの関数名のみ回答せよ
- 対応するURLパターンが存在せず、エラーとなる場合は「 `ERROR` 」と回答せよ

- `http://127.0.0.1:8000/recipes/1`

  ```
  show
  ```

- `http://127.0.0.1:8000/recipes/review`

  ```
  ERROR
  ```

- `http://127.0.0.1:8000/recipes/2/yummy`

  ```
  yummy
  ```

2)

このプログラムの `cookbook/models.py` に、レシピを表す以下のようなモデルが定義されている。

```
from django.db import models

class Recipe(models.Model):
    name = models.CharField(max_length=50)
    slogan = models.TextField()
    detail = models.TextField()
    yummy_count = models.IntegerField(default=0)
    yucky_count = models.IntegerField(default=0)
```

データベースにレコードが1件も登録されていない状況で、Djangoのシェル上でプログラムを入力したところ、以下のように表示された。

```
>>> from cookbook.models import Recipe
>>> recipe1 = Recipe(name='パスタ', slogan='イタリアの風', detail='パスタの作り方')
>>> recipe1.save()
>>> recipe2 = Recipe(name='カレー', slogan='[  D  ]', detail='カレーの作り方')
>>> recipe2.save()
>>> recipe3 = Recipe(name='コロッケ', slogan='キャベツはどうした', detail='コロッケの作り方')
>>> recipe3.save()
>>> for r in Recipe.objects.all():
...   print(r.id, r.slogan, r.yummy_count)
...
1 イタリアの風 0
2 洋食屋のカレー 0
3 キャベツはどうした 0
>>> r = Recipe.objects.get(pk=3)
>>> [  E  ]
'コロッケの作り方'
>>> r = Recipe.objects.get(pk=1)
>>> r.yummy_count = 1
>>> r.save()
>>> r = Recipe.objects.get(pk=2)
>>> r.yummy_count = 2
>>> r.save()
>>> r = Recipe.objects.order_by('yummy_count')[0]
>>> r.name
'[  F  ]'
```

空欄を埋めなさい。

- `[ D ]`

  洋食屋のカレー

- `[ E ]`

  `r.detail`

- `[ F ]`

  コロッケ

3)

このプログラムのレシピ一覧ページ (例: `http://127.0.0.1:8000/recipes/`) は、以下のような機能を持つ。

- このページにはレシピの投稿フォームとレシピ一覧が表示されている
- レシピの投稿フォームから、レシピの名前(name)、スローガン(slogan)、詳細(detail)を登録することができる
    - 投稿が完了すると、レシピの詳細情報を表示するページに遷移する

- 個々のレシピには、以下が表示される
    - レシピの名前
    - レシピのスローガン
    - レシピの「おいしい (Yummy)」の件数 (yummy_count)
    - レシピの「まずい (Yucky)」の件数 (yucky_count)
    - レシピの詳細
- レシピの名前をクリックすると、レシピの詳細情報を表示するページ に遷移する
- 「おいしい」「マズい」は、どちらもリンクになっており、クリックすると件数が1件増える
- クエリパラメータを指定すると、以下のように動作する
    - `?order=yummy` を指定すると、「おいしい」の件数が大きい順に表示される
    - `?order=yucky` を指定すると、「まずい」の件数が小さい順に表示される

このとき、`cookbook/views.py` の `list` 関数および、そこから利用される対応するテンプレートファイル `cookbook/list.html` の空欄を埋めよ。

```
from django.shortcuts import render
from django.shortcuts import render, redirect
from django.http import Http404
from cookbook.models import Recipe

def list(request):
    if request.method == 'POST':
        recipe = Recipe(name=request.POST['name'], slogan = request.P
OST['slogan'], detail=request.POST['detail'])
        recipe.save()
        return redirect(detail, recipe.id)

    if 'order' in request.GET:
        if [  G  ] == 'yummy':
            recipes = Recipe.objects.order_by('-yummy_count')
        else:
            recipes = Recipe.objects.order_by('yucky_count')
    else:
        recipes = Recipe.objects.all()

    context = {
        '[  H  ]': recipes
    }

    return render(request, 'cookbook/list.html', context)
```

```
<!DOCTYPE html>
<html lang="ja">

  <head>
    <meta charset="UTF-8">
    <title>Cookbook</title>
```

```
  </head>

  <body>
    <h1>Cookbook</h1>
    <form action="{% url 'list_recipe' %}" method="post">
      {% csrf_token %}
      <div>
        <label for="name">Recipe name</label><br>
        <input id="name" name="name" type="text" placeholder="Input r
ecipe name.">
      </div>
      <div>
        <label for="slogan">Slogan</label><br>
        <input id="slogan" name="slogan" type="text" placeholder="Inp
ut your slogan.">
      </div>
      <div>
        <label for="detail">Detail</label><br>
        <textarea id="detail" name="detail" rows="3"></textarea>
      </div>
      <div>
        <button type="submit">Submit</button>
      </div>
    </form>
    <br>

    <h2>Recipe list</h2>

    <div>
      <a href="{% url 'list_recipe' %}">Default</a> |
          <a href="{% url 'list_recipe' %}?order=yummy">Sort by Most
Yummy</a> |
      <a href="{% url 'list_recipe' %}?order=yucky">Sort by Least Yuc
ky</a>
          </div>

    {% for recipe in recipe_list %}
    <h3><a href="{% url 'show_recipe' recipe.id %}">{{ [  I  ] }}</a>
</h3>
    <div>{{ recipe.slogan }}</div>
    <div>
      <a href="{% url 'yummy_recipe' recipe.id %}">Yummy: <span>{{rec
ipe.yummy_count}}</span></a> |
      <a href="{% url 'yucky_recipe' recipe.id %}">Yucky: <span>{{rec
ipe.yucky_count}}</span></a>
    </div>
    <div>{{ recipe.detail | linebreaksbr }}</div>
    {% [  J  ] %}
  </body>
```

```
</html>
```

- [G]

```
request.GET['order']
```

- [H]

```
recipe_list
```

- [I]

```
recipe.name
```

- [J]

```
endfor
```

III. Database (計20点)

データベース `cdalbum.sqlite3` には、以下のような属性(列)を持つテーブル `albums` , `tracks` が作成されている。

- テーブル `albums`
  - アルバムID (列名: `id` , データ型: `integer` , 制約: 主キー, `autoincrement`)
  - CDアルバムの名前 (列名: `title` , データ型: `TEXT` , 制約: `not null`)
  - アーティスト名 (列名: `artist` , データ型: `TEXT` , 制約: なし)
- テーブル `tracks`
  - トラックID (列名: `id` , データ型: `integer` , 制約: 主キー, `autoincrement`)
  - 曲名 (列名: `name` , データ型: `text` , 制約: `not null`)
  - アルバムID (列名: `album_id` , データ型: `integer` , 制約: `albums` テーブルの `id` 属性を参照する外部キー)
  - 秒数 (列名: `seconds` , データ型: `integer` , 制約: なし)

また、それぞれのテーブルには以下のデータが登録されている。

- テーブル `albums`

id	title	artist
1	Greatest Hits II	Queen
2	Greatest Hits I	Queen
3	Emergency On Planet Earth	Jamiroquai
4	The Return Of The Space Cowboy	Jamiroquai

- テーブル `tracks`

id	name	album_id	seconds

1	A Kind Of Magic	1	262
2	Under Pressure	1	236
3	Bohemian Rhapsody	2	358
4	When You Gonna Learn	3	230
5	Too Young To Die	3	365
6	Hooked Up	3	275
7	Just Another Story	4	529
8	Stillness In Time	4	257
9	Half The Man	4	289
10	Light Years	4	354

上記のデータベースを操作することを想定し、以下の各問に答えよ。

- なお、この SQLite3 データベースファイルは https://moocs-files.iniad.org/2023/CS2/exam-cd-album/cdalbum.sqlite3 (https://moocs-files.iniad.org/2023/CS2/exam-cd-album/cdalbum.sqlite3) からダウンロードできる

(1) 以下の各操作に対応するSQL文の空欄を適切に埋めよ (各1点)

- なお、一つの空欄に複数語が入る場合もあることに注意すること

A. アルバムIDが3のトラックの名前と秒数を、秒数の長い順に表示する

```
[  (A)  ] name, seconds FROM [  (B)  ] WHERE [  (C)  ] ORDER BY [  (D)  ];
```

- [(A)]

  ```
  SELECT
  ```

- [(B)]

  ```
  tracks
  ```

- [(C)]

  ```
  album_id=3
  ```

- [(D)]

```
seconds DESC
```

B. テーブル `tracks` に、アルバムIDが `2`, 曲名が `We Will Rock You`, 秒数が `122` のデータを追加する

```
[  (E)  ] tracks [  (F)  ] VALUES (2, 'We Will Rock You', 122);
```

- `[ (E) ]`

```
INSERT INTO
```

- `[ (F) ]`

```
(album_id, name, seconds)
```

C. テーブル `tracks` からトラックIDが10のデータを削除する

```
[  (G)  ] tracks [  (H)  ] id = 10;
```

- `[ (G) ]`

```
DELETE FROM
```

- `[ (H) ]`

```
WHERE
```

(2) 以下の各操作を実現するSQL文を記述せよ (各4点)

A. トラックIDが `3` のトラックの秒数を `1000` に変更する

```
UPDATE tracks SET seconds=1000 WHERE id=3;
```

B. 秒数が200〜300秒の範囲にあるトラックの曲名と秒数を表示する

- 曲名と秒数以外の情報は表示しないようにすること

```
SELECT name, seconds FROM tracks WHERE seconds BETWEEN 200 and 3
```

C. アーティスト名が `Queen` であるCDに含まれるすべての曲名とそのアルバムの名前を表示する

- 曲名とアルバムの名前以外の情報は表示しないようにすること

```
SELECT tracks.name, albums.title FROM tracks JOIN albums ON albu
```

D. (Optional) すべてのアルバムについて、そのアルバムの名前とアルバムに含まれるトラック数を表示する

- アルバム名とその中に含まれるトラック数以外の情報は表示しないようにすること

- この問題を解かなくても100点をとることができます (解ければ4点が加算されます)

```
SELECT albums.title, count(tracks.album_id) FROM tracks JOIN alb
```

* 回答は自動的に記録されますが、最後に「提出」ボタンをクリックし提出を確認してください。

提出

あなたの得点

104/100

コメント