受験者(Examinee): 長谷川 匠 Sign Out (/signout)

注意: 現在回答を受け付けていません。

1. ((a),(b)各3点、(c)4点)

講義で扱ったリンクリストとその実装(下記のNodeクラスとそれを用いた関数)に関する問題である。以下の条件を満たすように下記のプログラムの空欄 __(a)__ から __(c)__ を正しく埋めよ。なお、各空欄内に改行を入れてはならない。

(1) 関数 swap(node1, node2) は、長さ2以上のリンクリストの任意の2個のノード node1 および node2 に対して、その value の値を交換する。 (2) for文の中では、以下の処理を行う。

- まず、 create_list 関数によってリンクリストを生成する。この長さを、forループ1回ごとに10000から100000まで10000刻みで増やすこと。
- 次に、生成されたリストに対して、 swap を用いて先頭のノードと、その次のノードの value の値を交換するのにかかる平均実行時間をJupyterLabと %timeit を用いて計測する。

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next = None
    def __str__(self):
        return str(self.value)

def print_list(top):
    print('<', end='')
    node = top
    while node is not None:
        print(node, end=", ")
        node = node.next
    print('>')

def create_list(n):
    top = Node(0)
    for i in range(1, n):
        insert(top, n - i)
    return top

def swap(node1, node2):
    temp = node1.value
    node1.value = __(a)__
    __(a)__ = temp

for i in range(1, 11):
    top = create_list(__(b)__)
    %timeit  __(c)__
```

ヒント: swap 実行例

```
>>> node1 = Node(3)
>>> node2 = Node(5)
>>> node3 = Node(6)
>>> node1.next = node2
>>> node2.next = node3
>>> print_list(node1)
<3, 5, 6, >
>>> swap(node1, node3)
>>> print_list(node1)
<6, 5, 3, >
```

- **(a)**
  `node2`
- **(b)**
  `i+10000`
- **(c)**
  `swap(top,value)`

2. (5点)

各面に1から8までのそれぞれの数値が書かれた正8面体をサイコロ同様に振って上になった面の数値を出た目とする。各数値の出る確率がすべて等しいとき、この正8面体を1回振った出目の持つ情報エントロピーを計算せよ。単位はビットとし、必要なら小数第三位を四捨五入すること。解答欄には、単位を書かず、数値だけを記入すること。

`3.0`

3. (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。

a)

```
def func1(lst):
    result = 0
    for i in range(100):
        for j in lst:
            result += g(lst, i * g(lst, j))
    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 = []
    for i in lst:
        result.append(f(lst, i))
    for j in range(5000000):
        result.append(j)
    return result
```

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

4. (各5点)

4-1

リストを処理するあるアルゴリズムがあり、その平均計算量はリストの長さ(サイズ)を$n > 0$としたとき $O(\log n)$ であるものとする。長さ 5000 の整数のリストに対して、このアルゴリズムを実行したところ、3ms かかったとする。 この時、長さ 25000000 (2500万) の整数のリストに対して、このアルゴリズムを実行したら、どの程度の時間がかかると考えられるか。実行時間が平均計算量に比例すると仮定し、最も近い値を選ぶこと。

- ◯ –
- ◯ 3ms
- ◉ 6ms
- ◯ 9ms
- ◯ 30ms
- ◯ 60ms
- ◯ 900ms
- ◯ 3000ms
- ◯ 6000ms
- ◯ 9000ms
- ◯ 30000ms

4-2

長さ 100000 の整数のリストをマージソートでソートしたところ、 50ms かかったとする。 この時、長さ 10000 の整数のリストを同じマージソートでソートすると、どの程度の時間がかかると考えられるか。実行時間が最悪計算量に比例すると仮定し、最も近い値を選ぶこと。

- ◯ –
- ◯ 1ms
- ◯ 2ms
- ◉ 4ms
- ◯ 5ms
- ◯ 10ms
- ◯ 20ms
- ◯ 40ms
- ◯ 50ms
- ◯ 100ms
- ◯ 200ms
- ◯ 400ms

5.

5-1 (5点)

マージソートで、以下のリストを昇順にソートすることを考える。この時、5回目のマージ完了後のリストの中身を答えよ。なお、講義で扱ったように、前半のソート完了後、後半のソートに取り掛かるアルゴリズムを使用するものとする。

リストは、各値を、カンマと空白で区切って回答すること。（例: `1, 2, 3, 4, 5, 6, 7, 8` ）

```
9, 4, 7, 3, 5, 2, 8, 1
```

```
3, 4, 7, 9, 2, 5, 1, 8
```

5-2 (5点)

講義で扱った「簡易版のクイックソート」を使用して、以下のリストを昇順にソートすることを考える。なお、中央(リストの長さが偶数の場合は、リストの後ろ半分の最初)の数値をピボットとして用いること。また、分割の際に、グループ内の順序は入れ替わらないと仮定すること。

この時、以下の問いに答えなさい。

```
9, 4, 7, 3, 5, 2, 8, 1
```

- 2回目の分割時に使用したピボットの数値を答えよ。

  ```
  2
  ```

6. (各5点)

`7, 8, 1` が既に入っているスタック(先頭が7)に対して、以下の操作を順に適用する。この時、以下の問いに答えなさい。

```
Pop
Pop
Push 5
Push 9
Push 4
Pop
Push 6
Pop
Push 2
Pop
```

- 最後の操作で、どのような値が取り出されるか。数値を答えよ。

  2

- 最後の操作の後、スタック内にはいくつの要素が残されているか。数値を答えよ。

  3

7.

7-1 (完答5点)

6種類の文字 A , D , H , N , P , R からなる文字列がある。それぞれの文字の出現頻度は、 A が40%, D が 15%、 H が5%、 N が5%、 P が 10%、 R が25% 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、 A , P , N のコードを書け。ただし、 H には 00000 を割り当てるものとする。

- A

  1

- P

  0001

- N

  00001

7-2 (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)。解答欄には数値のみ書くこと(%記号は不要)。

75

8. (各5点)

注: 適切に空白を用いること。例えば + 5 5 と +55 は異なる。

a)

以下の中置記法の数式を前置記法に直しなさい。

```
9 / 3 + (8 - 5) * (6 - 2)
```

```
+ / 9 3 * - 8 5 - 6 2
```

b)

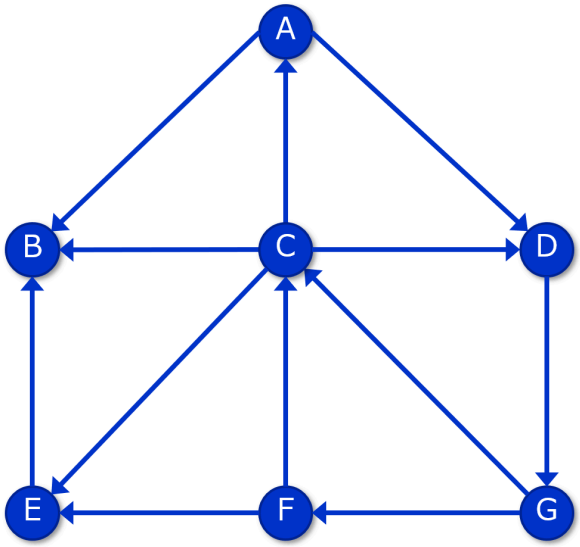
以下の後置記法の数式を中置記法に直しなさい。

```
3 1 - 7 2 + * 8 2 - /
```

```
(((3 - 1) * (7 + 2)) / (8 - 2))
```

9. (完答10点)

以下の有向グラフをノード(点)Aから深さ優先探索で探索するとき、訪れる順にノードを書くこと。なお、1番目は `A` とし、2番目以降を答えること。なお、各ノードに隣接しているノードのいずれかを選ぶ際には、アルファベット順に選択すること。



- 2番目

  ```
  B
  ```

- 3番目

  ```
  D
  ```

- 4番目

  ```
  G
  ```

- 5番目

  ```
  C
  ```

- 6番目

  ```
  E
  ```

10. (5点)

講義で扱った、pythonで整数を格納する以下のハッシュテーブルについて考える。

- 配列のサイズ `N` は10である。
- ハッシュ関数は下記の `h(x)` を用いる。
- ある整数を格納しようとして衝突が起こった際には、その整数は格納しない。

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

このとき、以下の整数を順にハッシュテーブルに格納しようとした際、格納されなかった整数をすべてカンマと空白で区切って書け。(例: 25, 62, 83)

整数列

10, 25, 62, 83, 71, 93, 45, 77, 14, 100

```
93, 45, 100
```

11. (10点)

空の二分探索木(注意: 平衡二分探索木ではない)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。

- 11, 12, 8, 9, 15, 1, 3, 18, 2, 14

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
  ```
  9
  ```
- 3番目に小さい値
  ```
  14
  ```
- 4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。

- 1, 11, 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。

```
14
```

---

* 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

90/100

コメント