Preferred Networks インターン選考2019 コーディング課題 (性能最適化分野)

変更履歴

- 2019年4月19日 : 初版

回答に当たっての注意

- 実装に用いる言語・ライブラリは自由です。レポートには使用した処理系・ライブラリのバージョンを明記してください
- 添付のPythonコードはPython 3.7.2, ply 3.11, numpy 1.16.2, scipy 1.2.1で動作確認しています
- インタープリタは低速なので作業上必要なら同等のものを再実装しても構いません
- 課題には自分だけで取り組んでください。この課題を他の応募者を含めた他人と共有・相談することを禁止します。課題期間中に GitHub の公開リポジトリ等にソースコードや問題をアップロードする行為も禁止します。漏洩の証拠が見つかった場合、その応募者は失格となります。ある応募者が別の応募者に回答をコピーさせた場合、双方の応募者が失格となります
- 想定所要時間は最大2日です。全課題が解けていなくても提出は可能ですので、学業に支障の無い範囲で取り組んで下さい

提出物

- 課題1、2 それぞれのレポート
- 課題1、2 それぞれのソースコード

評価基準

提出物は以下を満たしていることが望ましいです(必須ではありません)。

- ソースコードが他人が読んで理解しやすいものになっていること
- レポートが要点についてわかりやすくまとまっていること
- ソースコードが正しいものであることを検証するためにある程度の単体テストや検証のためのコードが書かれていること

一次選考後の面接で提出されたコードについて質問する可能性があります。

提出方法

上記の提出物を単一のパスワード無しzipファイルにまとめ、こちらの専用フォームより応募してください。締切は日本時間2019年5月7日（火）12:00です。

問い合わせ

問題文に関して質問がある場合はintern2019-admin@preferred.jpまでご連絡ください。問題文に訂正が行われた場合は応募者全員にアナウンスいたします。なお、アプローチや解法に関する問い合わせにはお答えできません。

問題文

課題1

離散フーリエ変換 $x \in \mathbb{C}^N \mapsto y \in \mathbb{C}^N$ は $y(l) = \sum_{k=0}^{N-1} x(k) \exp\left(-2\pi i \frac{kl}{N}\right)\ (l = 0, \ldots, N-1)$ で表される操作です。これを計算するプログラムを記述するための簡易言語を次で定義します。

```
<statement> ::= <ident> = <expression>  (スカラー変数への代入)
              | <ident>[<int>] = <expression>  (配列要素への代入)
<expression> ::= <primitive> + <primitive> (和)
               | <primitive> - <primitive> (差)
               | <primitive> * <primitive> (積)
               | <primitive>
<primitive> ::= f(<int>, <int>)  (入力 m, n に対してexp(-2πi m / n))
              | <ident>  (スカラー変数の読み出し)
              | <ident>[<int>]  (配列要素の読み出し)
              | <const>  (定数)
<ident> ::= [a-zA-Z_][a-zA-Z0-9_]*  (変数名)
<int> ::= -?\d+  (整数)
<const> ::= \([\djeE.+-]+\)  (複素数の定数、Pythonのcomplex()で解釈されます)
```

添付の interpreter.py は簡易言語のインタープリタです(Pythonパッケージとしてply、numpy、scipyを要求します)。入力初期化方法 -i {const,sin,sawtooth,chirp} と 次元 N を与えて起動すると、入力配列 x を指定した方法で初期化、出力配列 y をゼロ初期化し、標準入力から簡易言語で記述されたプログラムを読み込んで実行し、最後に y の内容と、正しい出力とyとの誤差を出力します。`<primitive>` や `<expression>` の値はすべて複素数です。配列を新しく定義することはできません。代わりに変数は無制限に宣言できます。

添付の naive.py は離散フーリエ変換の簡易言語での素朴な実装です。Nを与えて起動すると、標準出力に実装を出力します。 以上ふたつのスクリプトの使用例を示します。

```
$ python naive.py 2 > 2.txt
$ cat 2.txt
t = x[0] * f(0, 2)
y[0] = y[0] + t
t = x[1] * f(0, 2)
y[0] = y[0] + t
t = x[0] * f(0, 2)
y[1] = y[1] + t
t = x[1] * f(1, 2)
y[1] = y[1] + t
$ cat 2.txt | python interpreter.py -i const 2
(1+1j)
(5.551115123125783e-17-1.1102230246251565e-16j)
max_error: 1.2412670766236366e-16
$ cat 2.txt | python interpreter.py -i sin 2
(-1.8369701987210297e-16+0.8290816487408506j)
(2.2699482268387765e-16+0.12197486755430303j)
max_error: 4.3297802811774677e-17
```

また、簡易言語で記述されたプログラムの標準コストを、和または差1回につき1、積1回につき2とした総和で定めます。添付の cost.py は標準コストを算出します。

```
$ cat 2.txt | python cost.py
12
```

離散フーリエ変換は $\exp(a+b) = \exp(a)\exp(b)$ が成立することなどを利用して計算量を大きく下げることができます。N = 2 ~ 256について、標準コストがなるべく小さくなるように最適化された、簡易言語で記述された離散フーリエ変換プログラムを出力するプログラムを作成してください。また、標準コストの N = 2 ~ 256 での総和の値と合わせて最適化の内容を報告してください。その際すべての入力初期化方法(const, sin, sawtooth, chirp)とすべてのNで、interpreter.py が `max_error:` で報告する誤差が 10^{-8} 未満になるようにしてください。

課題2

現実の計算機での実行時間は標準コストで見積もられるものよりもずっと複雑です。そこで、現実の計算機における何らかの効果を反映してコストを自由に定義しなおし、そのもとで課題1と同様に、最適化された離散フーリエ変換プログラムを出力するプログラムを作成し、定義したコストの N = 2 ~ 256 での総和の値と合わせて最適化の内容を報告してください。誤差に関する条件も同様です。導入した効果を表現しやすいように簡易言語の文法を変更しても構いません。

ヒント：独自定義コストに含まれうる要素は、たとえばある引数でのf()の値との積のコストは低いとする、依存関係についての特定の条件のもとで複数操作を並列に実行できるとする、レジスタの構成を定めてレジスタ割り付けを行いその利用効率を勘案する、などが挙げられます。

以上