プログラミング基礎演習レポート 2016-2

電気電子工学科2年　541046J　 高瀬健吾

導入

Plus[Times[Sin[13.4],3],2]

Plus[Power[x,3],Sin[x]]

上のような表式で与えられる式もしくは一変数関数を評価するプログラムkadai.cを実装した。実行時引数で与えられた表式を木構造化し、木に対して微分操作を行って新たに微分後の木を作り結果を二階微分まで同様の形式で表示する。式が与えられた場合は具体的な値を、関数が与えられた場合その局所的な極値を画面に出力する。認識可能な基本関数は四則演算、三角関数、指数対数であり、定義域、値域は共に実数である。

手法・結果

課題１

表式を木構造化するにあたって、int operation、double number、struct node \*left、struct node \*rightを要素に持つ構造体nodeを定義した。表式と(int) operation、(double) numberの対応は以下のとおりである。

|  |  |  |
| --- | --- | --- |
| 式 | operation | number |
| Plus[] | 1 | 0 |
| Times[] | 2 | 0 |
| Subtract[] | 3 | 0 |
| Divide[] | 4 | 0 |
| Sin[] | 5 | 0 |
| Cos[] | 6 | 0 |
| Exp[] | 7 | 0 |
| Log[] | 8 | 0 |
| Power[] | 9 | 0 |
| x(変数) | 10 | 0 |
| n(実数) | 0 | n(実数) |

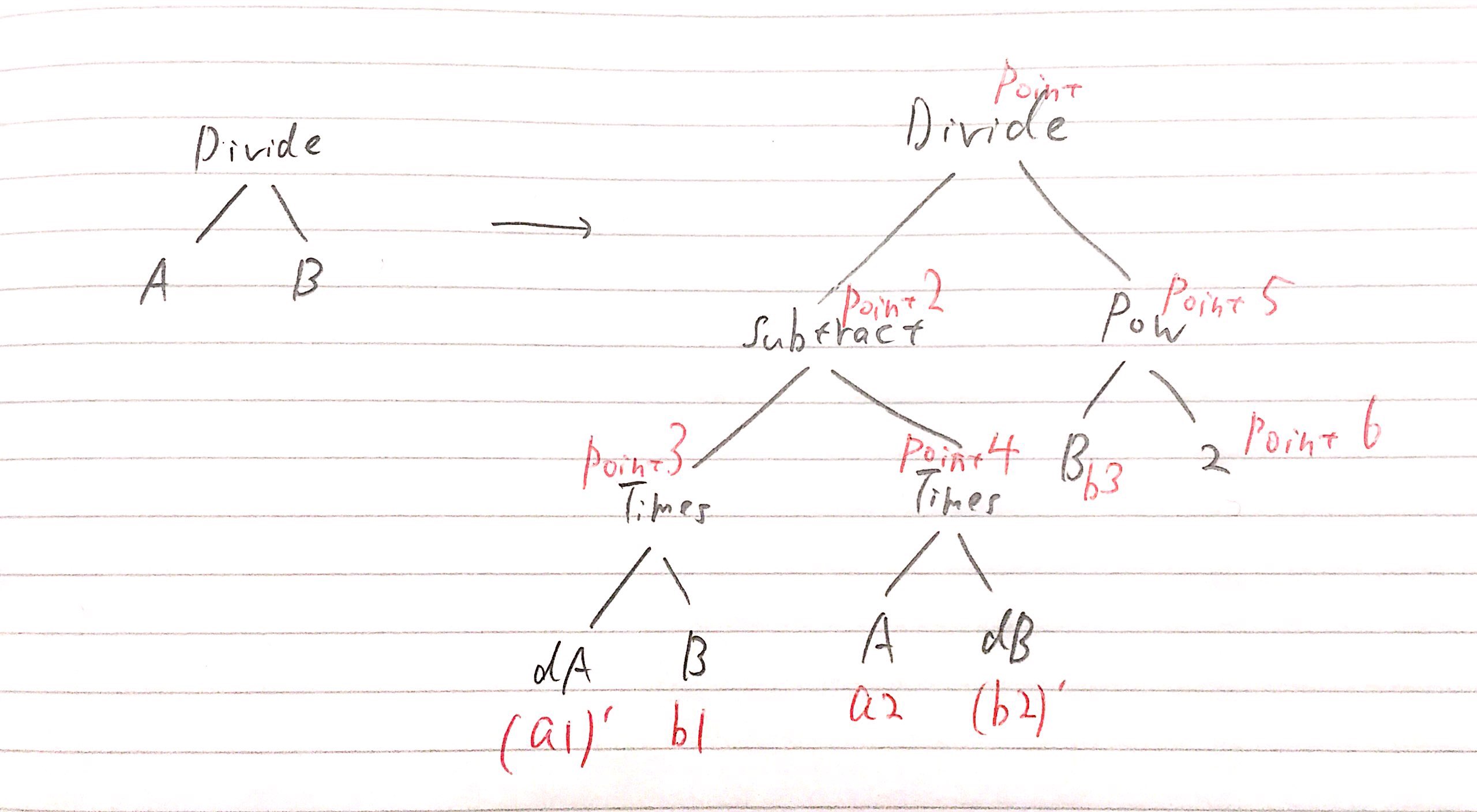
まず木構造と表式の変換を行う関数struct node \*create\_node(int \*pos,char \*s)とvoid traverse(struct node \*p)を実装した。第11回で実装したものとほぼ同じだが、今回は演算、数字ともに1文字ではないため、double read\_number(int \*pos,char \*str)、int read\_operation(int \*pos,char \*str)を用いて文字列の読み込みと読み込み位置posの移動を行っている。

さらに、int iffunc(struct node \*p)で木構造化された式がxの関数であるかどうか判定する。返り値は木構造内に格納された変数xの個数とした。ある木構造pに対してiffunc(p)が0のとき、つまりpがxによらない定数であるときはdouble calculate\_number(struct node \*p)で具体的な値を返す。

課題２

木構造化した式に対して微分操作を行い、新たに作った微分後の木構造のポインタを返す関数struct node \*differentiate\_node(struct node \*p)を実装した。ここでp->operationごとの変換規則は以下のとおりとした。。p->operationごとの対応は以下の通りとした。

微分して新たにできる基本関数それぞれに対して構造体を割り当てる必要があるため、コードの可読性が著しく下がってしまった。例としてp->operationが4の時のコードの対応を挙げる。



引数となる元の木構造を破壊しないために木構造を複製するstruct node\* copy\_node(struct node \*p)をあわせて実装している。

考察

参考文献