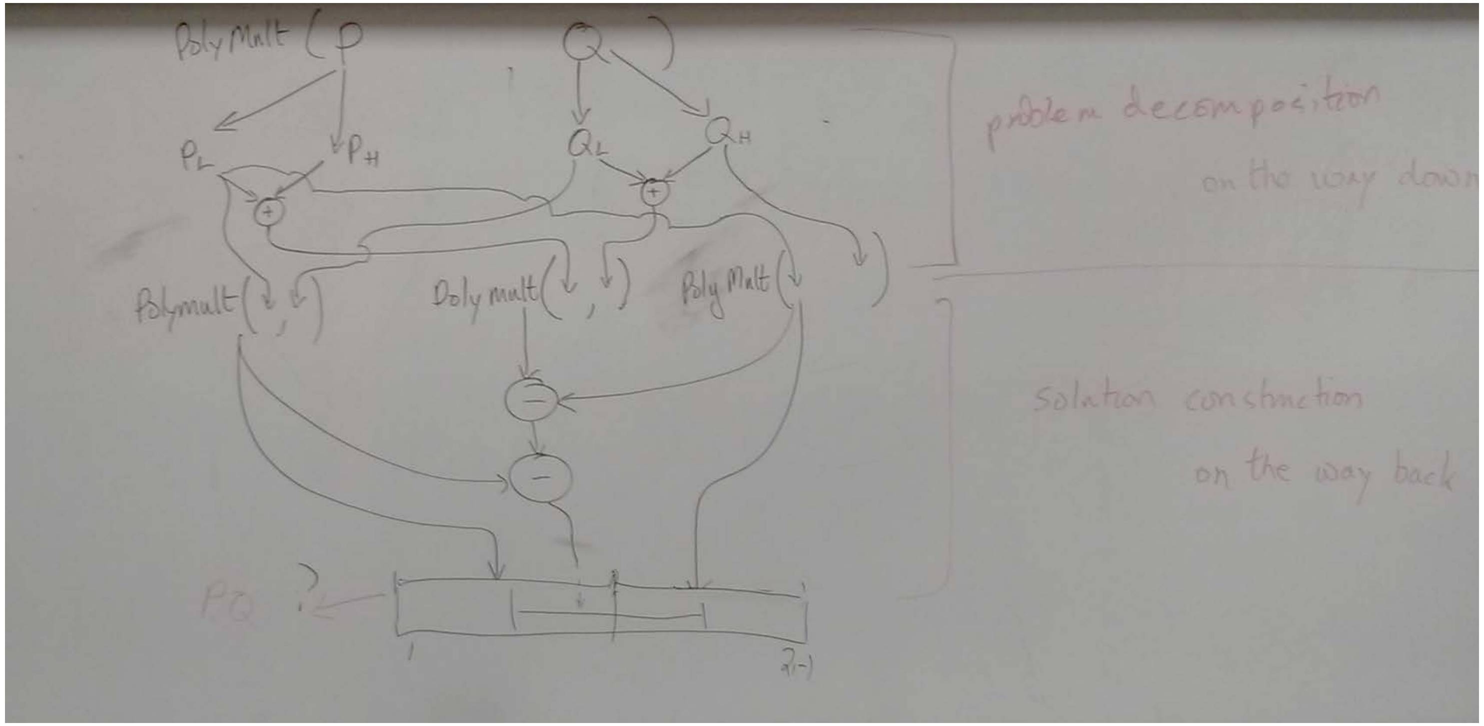
Cs 5050

02 13 14

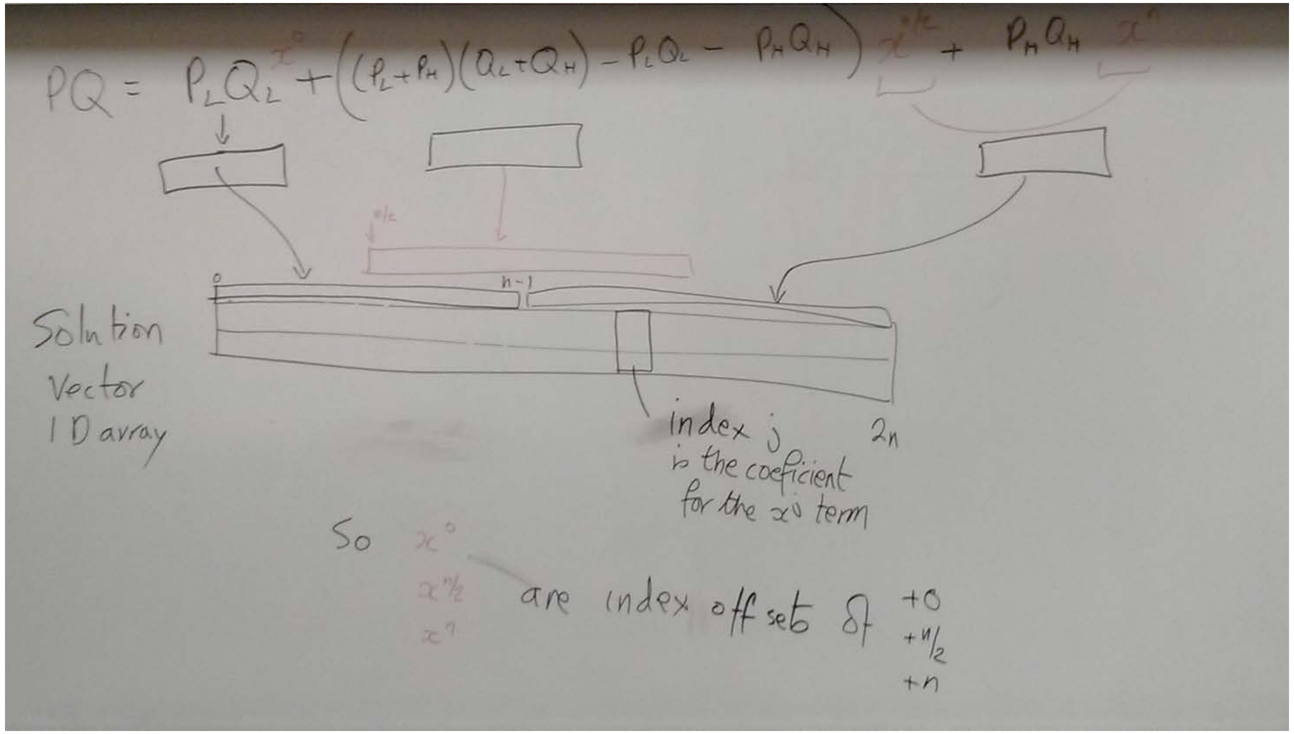



```
//Split P and Q into low and high
for i = 0 to n/2-1
   PL[i] = P[i]
   PH[i] = P[i+n/2]
end
```

```
//Generate PL plus PH
for i = 0 to n/2-1
      PLandPH = PL[i] + PH[i]
end
```

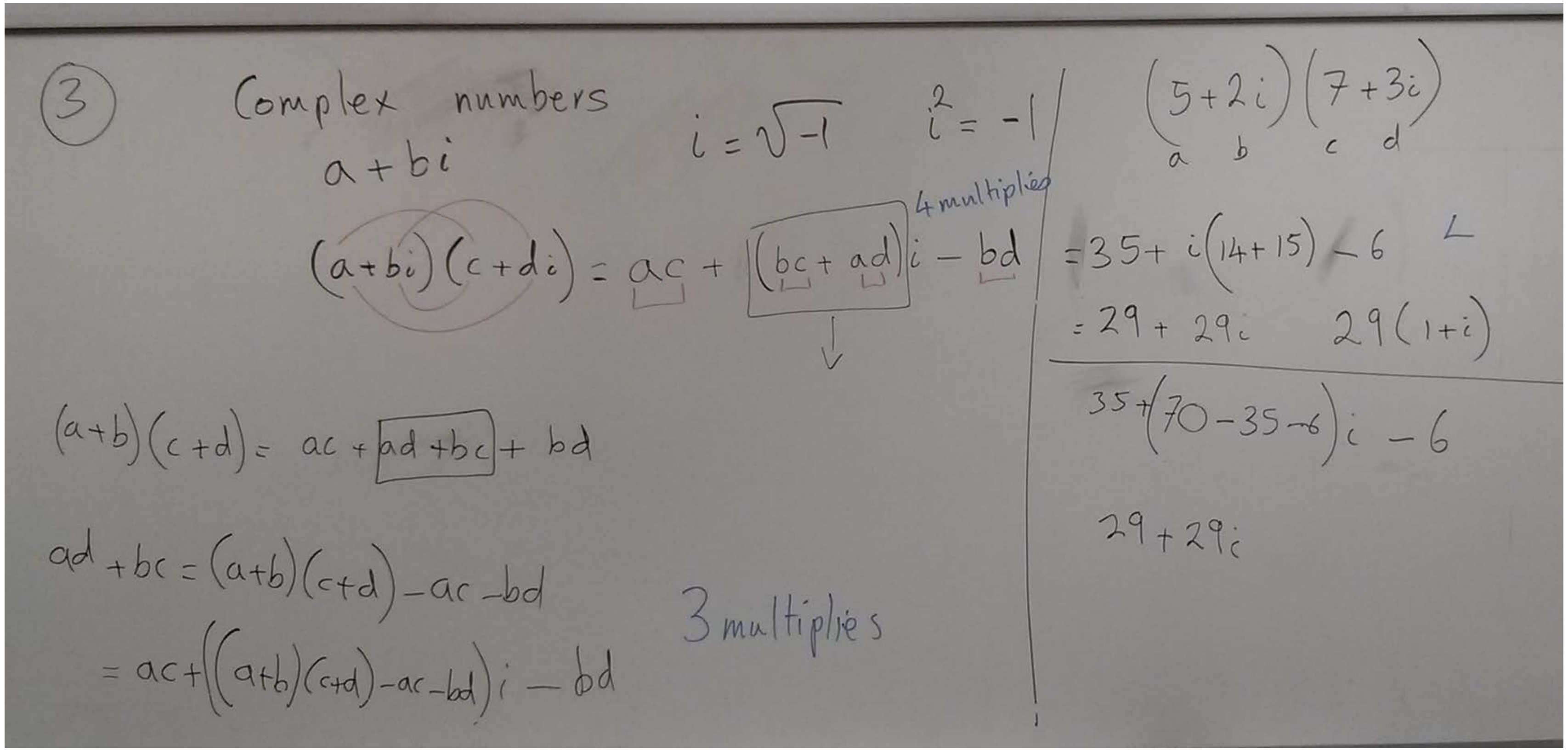
Generating the sub problems by scanning through the input arrays

Need to understand that the i^{th} coefficient of the polynomial P is stored at P[i]



PLQL = polyMult(PL, QL)
PHQH = polyMult(PH, QH)
PQSum = polyMult(PLandPH, QLandQH)
For i =0 to n-1
   PQ[i]+= PLQL[i]
   PQ[i+n/2] += PQSum[i]-PLQL[i]-PHQH[i]
   PQ[i+n] += PHQH[i]
end

PLQL, PHQH, PQSum are solutions to the subproblems, each is of size n
PQ is the whole answer of size 2n
We put the subsolutions into the whole solution array at the correct offsets based on $x^{n/2}$ (offset by n/2) and x^n (offset by n)



Complex numbers are two dimensional numbers, One real dimension and imaginary dimension

Multiplying two complex numbers using the “high school” algorithm requires 4 real multiplies

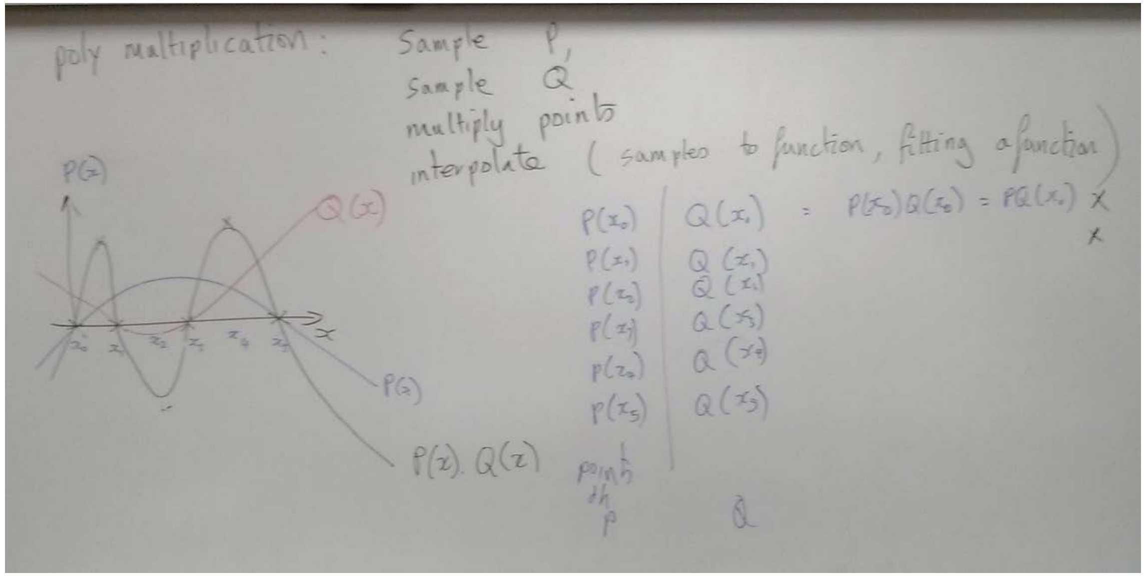
We can use the same trick that we use in polynomial multiplies:

(a+bi)(c+di) = a*c-b*d +((a+b)*(c+d) –a*c – b*d

Only three real multipies are needed

```
complexMult( C1, C2 )                I     R     local

ac = C1.R * C2.R
bd = C1.I * C2.I
mid = (C1.R + C1.I) * (C2.R + C2.I) - ac - bd

Cans <- new ....

Cans.R = ac - bd

Cans.I = mid
```

Simple method for multiplying two complex numbers using three real multiplies



Problem:
Given two polynomials $P=p_0+p_1x+p_2x^2$ …. $P_{n-1}x^{(n-1)}$ each represented as a 1D array of reals, so P[i] is the coefficient for the x to the i power
Find the polynomial P times Q as a 1D array of coefficients

New method that will turn out to be nlogn

1) Generate 2n real numbers and store in X[i]
2) Evaluate the polynomial P and Q at 2n values X[0], x[1]. … x[2n-1]
    1) Pvalues[i] = evalPolynomial(X[i])
    2) Qvalues[i] = evalPolynomial(X[i])
    3) PQvalues[i] = Pvalues[i]* Qvalues[i]
3) We know that the polynomial formed from P times Q will pass through all PQvalues.
4) Last step is to use a special algorithm called interpolation that takes a list of points of a polynomial and finds the coefficients of that function.