Extra Stuffs

It takes a really bad school to ruin a good student
and
a really fantastic school to rescue a bad student.

Dennis J. Frailey

General Thoughts

- **I always believe that the first impression in whatever course is important, because the first thing you learn may etch deep in your mind.**
- **As a result, you may unintentionally use something that you should not use in advance courses such as this.**
- **This set of slides try to illustrate some of my point. Feel free to disagree.**

Simple Examples: 1/12

- **Everyone knows how to compute the combinatorial coefficient $C(n,r)$ as follows:**

$$C(n,r) = \frac{n!}{r!(n-r)!}$$

- **How many of you know this is actually not a very good idea? It takes n-1 multiplications for n!, r-1 multiplication for r!, and (n-r)-1 multiplications for (n-r)!. So, the total number of multiplications is (n-1)+(r-1)+(n-r-1) = 2n-3.**

Simple Examples: 2/12

- **But, *n*!, *r*! and (*n*-*r*)! have some common part. That is, 1*2 *3*… *k* for some *k*.**
- **Suppose *n*-*r* > *r*. Then, the combinatorial coefficient can be simplified to the following:**

$$C(n,r) = \frac{((n-r)+1)\times((n-r)+2)\times\cdots\times n}{1\times 2\times\cdots\times r}$$

- **How many multiplications are needed?**
- **Simple. It is 2(*r*-1). Both the top part has *r* terms, and the lower part also have *r* terms. Both requires *r*-1 multiplications!**
- **Comparing 2(*r*-1) with 2*n*-3, what a huge difference!**

Simple Examples: 3/12

- Everyone knows how to compute x^n. It is:

```
product = 1;
for (i = 1; i < n; i++)
   product *= x;
```

- It takes n-1 multiplications! Is this good? Maybe.
- What if the x is a $k \times k$ matrix?
- Multiplying two $k \times k$ matrices requires k^3 multiplications.
- This means we need $(k^3)^{n-1} = k^{3(n-1)}$ multiplications!
- ***It is not very good!***

Simple Examples: 4/12

- **Do you still remember the divide-and-conquer technique?**
- **For x^n, if n is even then $x^n = (x^{n/2})^2$. For example, for x^{16}, we have $(x^8)^2$. Immediately, we cut the number of multiplications in half.**
- **If n is odd, than $x^n = (x^{n/2})^2 \times x$! For example, $x^{17} = (x^{16}) \times x = (x^8)^2 \times x$.**
- **The number multiplications is $O(\log_2 n)$.**
- **Therefore, if x is a $k \times k$ matrix, the number of multiplications immediately reduces to $(k^3)^{\log 2(n)} = k^{3(\log 2(n))}$, a significant reduction.**

Simple Examples: 5/12

- **You also know how to solve $ax^2+bx+c=0$ with the formula:**

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- **Is this a correct solution! Yes, it is correct, but it is an extremely poor one.**
- **Consider $a = 1$, $b = 100000$ and $c = 1$.**
- **Using the above formula, my simple program shows that a root is 0 and the other -100000!**
- **What went wrong?** ***We did not use floating point carefully.***

Simple Examples: 6/12

- **When b is large, b^2 is even larger!**
- **The $4*a*c$ part may be very insignificant!**
- **As a result, $b^2-4ac \approx b^2$ and $(-b)+(b^2-4ac)^{1/2} \approx (-b) + b \approx 0$. Thus, one of the two roots is 0!**
- **The formula is correct, but our programming method is *wrong*!**
- **Avoid the use of subtraction as much as possible.**

Simple Examples: 7/12

- Because the square root of b^2-4ac is always positive, depending on the sign of b, we are able to eliminate the subtraction (or the sum of a positive and negative) in the numerator part.

```
d = sqrt(b*b - 4*a*c);  // this one is always positive
if (b > 0)              // if b > 0 ..
   r1 = (-b) - d;       //   add 2 negative
Else                    // b < 0 here
   r1 = (-b) + d;       //   add 2 positive
root1 = r1/(2*a);       // first root
root2 = (c/a)/r1        // remember root1*root2 = c!!
```

- There are better methods, of course. But, this one is easy and effective!

Simple Examples: 8/12

- **Suppose we are given a sorted array and we want to find the longest section in the array such that the numbers are the same. This is referred to as a *plateau*!**
- **Suppose the array is 1, 2, 2, 2, *3, 3, 3, 3, 3*, 4, 5. The longest plateau is 5, because we see *3, 3, 3, 3, 3* of length 5.**
- **How do you write a program to find the length of the longest plateau? *We only need the length*.**

Simple Examples: 9/12

- Your program may very likely to be the following:

```
length = max_length = 1;      // plateau length at least 1
last = 1;                     // last position from 1
for (i = 2; i <= n; i++) {    // scan all elements
   if (x[i] == x[last] )      // if the current == last
      length++;               // length increases by 1
   else {                     // otherwise, a new plateau
      if (length >= max_length) {  // longer than max?
         max_length = length;  // YES, update length
         last = i;            // current is the last pos
         length = 1;          // length starts from 1
      }
   }
}
return max_length;
```

Simple Examples: 10/12

- This is an ugly program, too straightforward without much depth.
- Look at the following code? Do you understand it?
- If we know an existing plateau of `length`, we do not have to compare two elements whose distance is shorter than `length`.
- This is shorter and more elegant!

```
length = 1;                       // plateau length >= 1
for (i = 1; i < n; i++)           // scan the array
   if (x[i] == x[i-length])       // plateau of length?
      length++;                   // found 1 more
return length;
```

Simple Examples: 11/12

- My first challenge in programming: ***write a program to sort 10 distinct integers without using array!***
- How can you it?

```
Q = MAX(A,B,C,D,E,F,G,H,I,J);
P = MIN(A,B,C,D,E,F,G,H,I,J);
for (i = P; i <= Q; i++) {
   if (i == A) printf("%5d\n", i);
   if (i == B) printf("%5d\n", i);
   if (i == C) printf("%5d\n", i);
    ............
   if (i == J) printf("%5d\n", i);
}
```

Simple Examples: 12/12

- If you know C well, you certainly can do better!
- Assume no number is `MAX_INT`.
- This is a bubble sort-like program!

```
#define  MIN(x,y)  ((*x) <= (*y) ? (x) : (y))

int *p;

for (i = 0; i < 10; i++) {
   p = MIN(MIN(MIN(MIN(&a,&b), MIN(&c,&d)), \
           MIN(MIN(&e,&f), MIN(&g,&h))), MIN(&i,&j));
   printf("%5d", *p);
   *p = INT_MAX;
}
```

Parting Thoughts

- **Good programmers always find the best and nost efficient way to solve a problem rather than simply getting the job done.**
- **If you do that, you are a coder rather than even a programmer.**
- **No one but yourself can limit your imagination and creativity.**
- **Please think more and deeper, and be a good programmer and designer.**

The End