Practice 06. Arrays and Pointers

Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University

http://bi.snu.ac.kr

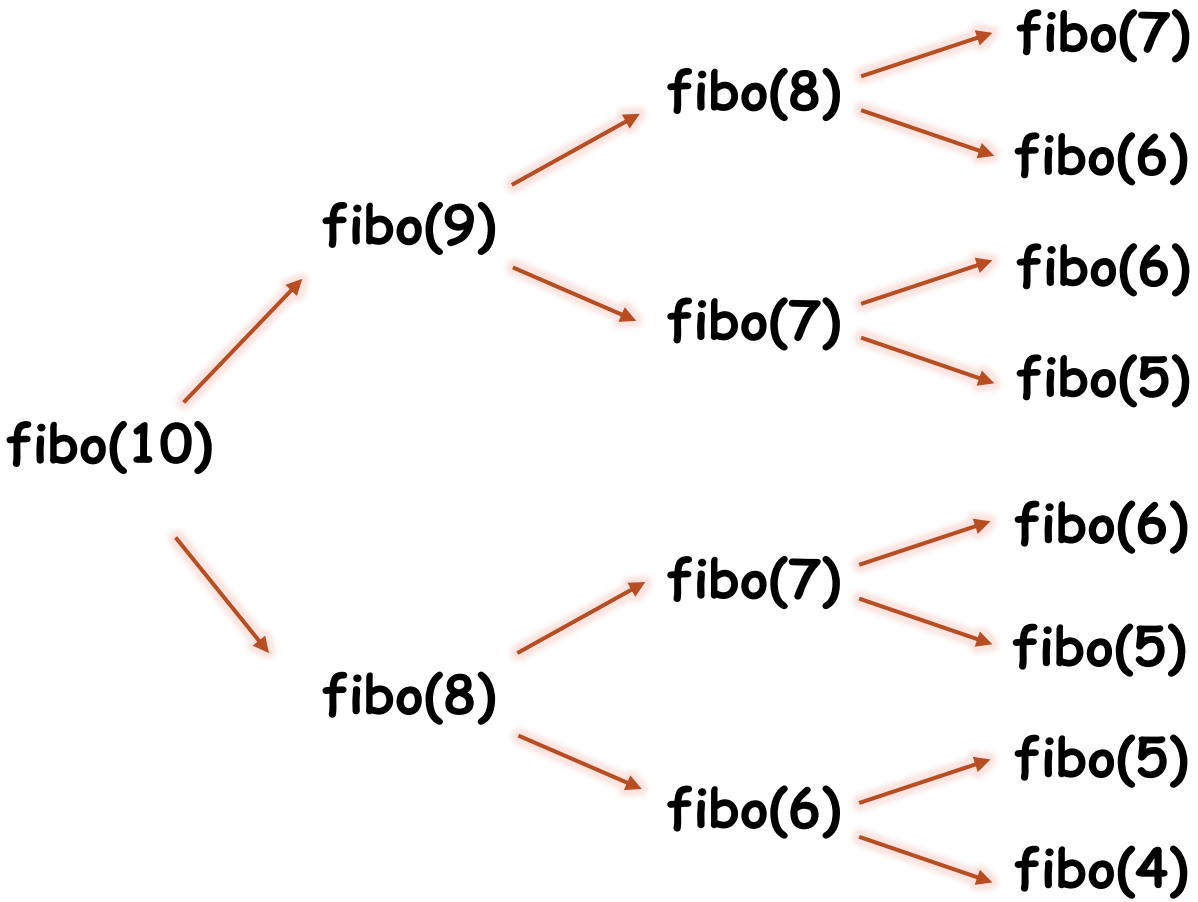
Dynamic Programming

- The dynamic programming is a technique that analyze the problem and see the order in which the sub-problems are solved and start solving from the trivial subproblem, up towards the given problem.
- Bottom-Up Process.

```
#define MAX 1000
...
int factorial[MAX];
...
factorial[0] = 1;
for (i = 1; i < MAX; i++)
   factorial[i] = factorial[i - 1] * i;
```

Dynamic Programming

- The recursive version of implementing the Fibonacci sequence is not efficient.



Dynamic Programming

- The dynamic programming for the Fibonacci sequence.

```
#define MAX 1000
...
int fibo[MAX];
...
fibo[0] = 1;
fibo[1] = 1;
for (i = 2; i < MAX; i++)
   fibo[i] = fibo[i - 1] + fibo[i - 2];
```

Practice Submission

- **Submit the practice problems if they are not checked in the class time.**
- Submit the solution code of **practice problem 01, 02, ...** by email.
- hnkwak@bi.snu.ac.kr
- Mail title: prg_[student number]_practice06
  - prg_2014-12345_practice06
- Submit two source files named **p01.c, p02.c, ...** for each problem.
- Due to : **4/15(Wed) 23:59 pm**

Assignment Submission

- Create a directory named assignment in you home directory.
- Create a directory named 06 in you assignment directory.
- Put your C files named p[# of problem].c for each problem.
  - p01.c
  - p02.c
  - ...
- Due to : **4/15(Wed) 23:59 pm**

Practice 01 – Inverse Matrices

- Complete the following code.
- **Do not modify the main function**

```
int main()
{
    int n; // the number of matrix
    int i;
    scanf("%d", &n);

    for (i = 0; i < n; i++)
    {
        double m[4];
        scanf("%lf%lf%lf%lf", m, m+1, m+2, m+3);
```

```
        if (no_inverse(m))
            printf("invalid matrix\n");
        else
        {
            invert(m);
            printf("%f %f %f %f\n", m[0], m[1], m[2], m[3]);
        }
    }
}
```

Practice 02 – Reverse

- The first line of the input gives a integer N. (0 < N < 30)
- The second line of the input contains N integers separated by a space.
- Output the reverted list of the integers.

```
[Input]
10
5 10 4 9 12 -4 0 -10 2 2

[Output]
2 2 -10 0 -4 12 9 4 10 5
```

Practice 03 – Sorting

- The first line of the input gives a integer N. (0 < N < 30)
- The second line of the input contains N integers separated by a space.
- Output the sorted list of the integers in ascending order.

[Input]

```
10
5 10 4 9 12 -4 0 -10 2 2
```

[Output]

```
-10 -4 0 2 2 4 5 9 10 12
```

Assignment 01 – Mean and Std.

- The first line of the input gives a integer N. (0 < N < 30)
- The second line of the input contains N integers separated by a space.
- Output the mean and standard deviation of those integers.

[Input]
10
5 10 4 9 12 -4 0 -10 2 2

[Output]
3.000000 6.324555

Assignment 02 – Chickens of Fibonacci

- It’s tricky to figure out how many fried chickens are needed for a dinner party.

Chickens of Fibonacci

- Assume there are k people. We want to order c_k fried chickens.
- Let $\{a_n\}$ be a Fibonacci sequence where $a_0 = 1, a_1 = 1, a_n = a_{n-1} + a_{n-2}$.
- If there exist n such that $a_n = k$, then $c_k = a_{n-1}$.
- Otherwise find the biggest value a_n that is smaller than k. Then $c_k = c_{k-a_n} + a_{n-1}$.
- For example, suppose there are 100 people.
  - $a_{10} = 89, a_9 = 55, a_5 = 8, a_4 = 5$
  - $c_{100} = a_9 + c_{11} = a_9 + (a_4 + c_3) = a_9 + (a_4 + a_2) = 55 + 5 + 2 = 62$

Chickens of Fibonacci

- The input contains the number (it's less than 300) of people.
- Output the number of fried chickens to order.

[Input]
100

[Output]
62