최적 알고리즘

최적 알고리즘은 가장 강력한 **최상**의 **알고리즘**을 찾아내는 것이다
최근, **인공지능** 분야에서도 최적화 알고리즘이 핵심적인 기술로 떠오르고 있다.**(당연)**
특히, 많은 양의 데이터를 빠른 속도로 처리하기 위해 최적 알고리즘은 매우 중요하다.

// 피보나치 수열에서 n번째 항을 차례로 구하는 프로그램

n 번째	0	1	2	3	4	5	6	7	8	9	10	…
피보나치 수	0	1	1	2	3	5	8	13	21	34	55	…

• fibo(n) = fibo(n-1) + fibo(n-2)

재귀적 방법(분할정복법)

```
int fibo(int n)
{
  if(n<=1) return n;
  else     return fibo(n-1) + fibo(n-2);
}
void main(){
  int i;
  for(i=0; i<10; i++)
    printf("%5d", fibo(i) );
}
```

[실행결과]

0 1 1 2 3 5 8 13 21 34

반복적 방법(동적계획법 – 배열을 이용)

```
#define MAX_SIZE 100
int f[MAX_SIZE + 1];
int fibo(int n){
    int i;
    f[0]=0;
    if(n>0){
      f[1]=1;
      for(i=2; i<=n; i++)
        f[i]=f[i-1]+f[i-2];
    }
    return f[n];
}
```

질문	피보나치 수열에서 **n**번째 항을 차례로 구하는 **최적**의 **알고리즘**은?

재귀적 방법(분할정복법)	반복적 방법(동적계획법 – 배열을 이용)

int fibo(int n)
{
  if(n<=1) return n;
  else return fibo(n-1) + fibo(n-2);
}
void main(){
  int i;
  for(i=0; i<10; i++)
  printf("%5d", fibo(i));
}
[실행결과]
0 1 1 2 3 5 8 13 21 34
	#define MAX_SIZE 100
	int f[MAX_SIZE + 1];
	int fibo(int n){
	  int i;
	  f[0]=0;
	  if(n>0){
	  f[1]=1;
	  for(i=2; i<=n; i++)
	  f[i]=f[i-1]+f[i-2];
	  }
	  return f[n];
	}

[재귀적 알고리즘 분석] ① 프로그램 작성 및 이해하기는 쉽다. ② 함수 fibo(i)는 같은 값을 반복해서 계산하기 때문에 매우 비효율적이다.**(재귀트리로 분석 가능)** → 재귀적 알고리즘의 문제점이다. ③ 계산시간(항수) : $2^{\frac{n}{2}}$ 개 이상**(지수시간)**	[반복적 알고리즘 분석] ① 계산된 값을 배열에 저장한다. → 나중에 필요할 때 다시 계산할 필요가 없다. ② 배열을 사용하지 않고 작성할 수도 있다. → 반복할 때마다 최근 2개항만 있으면 되므로 → 배열을 사용하면 알고리즘이 더 명확할 뿐이다. ③ 계산시간(항수) : n + 1개 → **좋다**

(1) 분할정복법

① 분할정복법은 **하향식** 접근 방법이다.
→ 상위 사례의 해답은 아래로 내려가서 작은 사례에 대한 해답을 구함으로써 해결한다.

② 분할정복법은 **재귀적** 루틴을 사용하는 방법이다.
→ 재귀적 루틴이 비효율적이면 이를 토대로 더 효율적인 반복적 알고리즘을 만들 수도 있다.

③ **예** : 빠른정렬(퀵정렬), 합병정렬, 큰 정수 계산법, 임계값 결정법

(2) 동적계획법

① 동적계획법은 **배열**을 이용한 **상향식** 접근 방법이다.
→ 먼저 작은 사례를 해결하고 그 결과를 저장한다. 나중에 저장된 결과가 필요하면 이용한다.

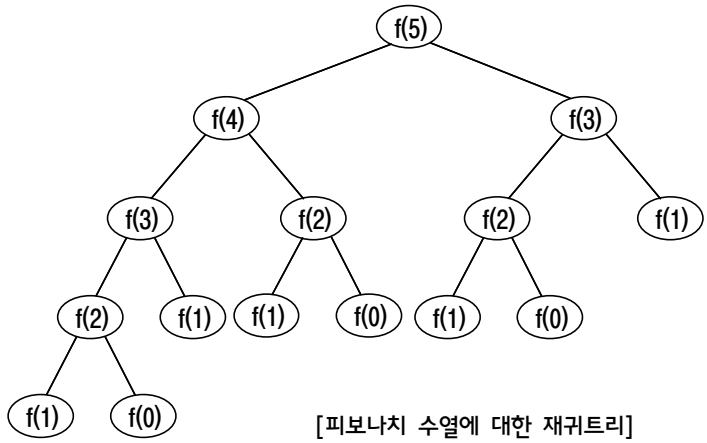
② 동적계획법은 배열을 이용한 **반복적** 알고리즘을 일컫는다.

③ **예** : 피보나치수 구하기, 이항계수 구하기, 최단경로, 최적 이진탐색트리 구하기, 외판원 문제

결론	• 어떤 문제를 해결하기 위한 가장 효율적인 알고리즘은 특별하게 정해지는 것은 아니다. • 여러 **조건**과 **주변 환경**에 따라 가장 효율적인 알고리즘은 다르다. • 가장 좋은 방법을 찾을 뿐이다.

// 피보나치 수열과 재귀트리

다음은 피보나치 수열에서 5번째 항을 구하는 재귀 알고리즘에 대한 재귀트리이다.



[피보나치 수열에 대한 재귀트리]

f(5) = f(4) + f(3)
f(4) = f(3) + f(2)
f(3) = f(2) + f(1)
f(2) = f(1) + f(0)
f(1) = 1
f(0) = 0

- 재귀트리에 의한 알고리즘은 이해 및 작성은 쉽다.
- 하지만 피보나치 수를 구하는 재귀 알고리즘 실행시간은 매우 비효율적이다. **지수시간**이다.
- f(5)를 구하기 위해서는 f(4)와 f(3)이 필요하고, f(4)는 f(3)과 f(2)를 필요로 한다.
- f(n)을 구하기 위해 계산해야 되는 항의 수는 다음처럼 증가한다.

n	0	1	**2**	3	**4**	5	**6**	…	→ 2씩 증가할 때
계산해야 되는 항의 수	1	1	**3**	5	**9**	15	**25**	…	→ **2배** 이상씩 증가함

n이 2씩 증가할 때 '계산해야 되는 항의 수'는 2배 이상씩 증가하고 있다.

여기서, T(n)을 n에 대한 '계산해야 되는 항의 수'라고 하면

```
T(n) > T(n-2) * 2  → n이 2씩 증가할 때, T(n)은 2배 이상씩 증가함으로
     > T(n-4) * 2 * 2
     > T(n-6) * 2 * 2 * 2
              :
     > T(0) * 2 * 2 * 2 * 2 * … * 2 → T(0) = 1이고, 2의 개수는 n/2개이다.
```

$\therefore T(n) > 2^{\frac{n}{2}}$	• 재귀 알고리즘으로 n번째 피보나치 수를 구하기 위해서 • '계산해야 되는 항의 수'는 2의 거듭제곱보다 크다는 것이다. • 매우 비효율적